In [ ]:
```python
#Converting a decimal number into different basis
print("Enter a decimal number..")
x=int(input())
print("Binary is=",bin(x))
print("Oct is=",oct(x))
print("Hex is=",hex(x))
```

```
Enter a decimal number..
128
Binary is= 0b10000000
Oct is= 0o200
Hex is= 0x80
```

In [ ]:
```python
#python buildin functions
print(round(23.47))
print(abs(5-6))
print(max(2,4,6))
print(min(1,2,3,4))
print(divmod(24,3))#prints both quotient and remainder
print(bin(18))
print(oct(128))
print(eval('1+2'))# eval() function returns the value that results fro
m evaluating the input string
```

```
23
1
6
1
(8, 0)
0b10010
0o200
3
```

In [ ]:
```
#math module
import math
dir(math)
```

Out[ ]:
```
['__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'e',
 'erf',
 'erfc',
 'exp',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
 'gcd',
 'hypot',
 'inf',
 'isclose',
 'isfinite',
 'isinf',
 'isnan',
 'ldexp',
 'lgamma',
 'log',
 'log10',
 'log1p',
 'log2',
 'modf',
 'nan',
 'pi',
 'pow',
 'radians',
 'remainder',
 'sin',
 'sinh',
 'sqrt',
 'tan',
 'tanh',
 'tau',
 'trunc']
```

In [ ]:
```
import math
print (math.pow(5,2),math.sqrt(25))
print("value of 8^2 is and the value of 5^4 ",math.pow(8,2),math.pow(5
,4,))
```

```
25.0 5.0
value of 8^2 is and the value of 5^4  64.0 625.0
```

In [ ]:
```
#Python Program to find Area and Circumference of a Circle
#Standard formula to calculate the Area of a circle is: a=π r².
#Circumference c=2 π r.
import math
r=input("Enter radius :")
r=int(r)
a=math.pi * r * r
c=2* math.pi * r
print("Area of the circle",a)
print ("Circumference of the circle",c)
```

```
Enter radius :25
Area of the circle 1963.4954084936207
Circumference of the circle 157.07963267948966
```

In [ ]:
```
#program to convert time in sec to HH:MM:SS format
time=input("Enter time in seconds")
time=int(time)
timeinmin=time//60
timeinsec=time%60
timeinhr=timeinmin//60
timeinmin=timeinmin%60
print("HH:MM::SS----{}:{}:{}".format(timeinhr,timeinmin,timeinsec))
```

```
Enter time in seconds1600
HH:MM::SS----0:26:40
```

In [2]:
```
#largest of 2 numbers
x=int(input("enter the first no"))
y=int(input("enter the second no"))
if x>y:
    print(x,"is greater")

else:
    print(y,"is greater")
```

```
enter the first no4
enter the second no7
7 is greater
```

In [3]:
```
#largest  and smallest of 3 numbers (max and min)
x=int(input("enter the first no"))
y=int(input("enter the second no"))
z=int(input("enter the thirdno"))
newmin=min(x,y)
newmax=max(x,y,z)
print("maximum value",newmax)
print("minimum value",newmin)
```

```
enter the first no4
enter the second no8
enter the thirdno3
maximum value 8
minimum value 4
```

In [4]:
```
#grade of students
mark=int(input('enter the marks'))
if mark>89:
        print ("A grade")
elif mark>79 and mark<90:
        print ("b grade")
elif mark>69 and mark<80:
        print ("c grade")
else:
        print ("d grade")
```

```
enter the marks76
c grade
```

In [5]:
```
#quadrant of a given point
x=int(input("enter the x axis"))
y=int(input("enter the y axis"))
if x>0 and y>0:
    print("first quadrant")
if x<0 and y>0:
    print("second quadrant")
if x<0 and y<0:
    print("third quadrant")
if x>0 and y<0:
    print("fourth quadrant")
```

```
enter the x axis-6
enter the y axis4
second quadrant
```

In [6]:
```
#given 3 sides of a triangle.check whether it forms a triangle or not
a=int(input("enter the first side"))
b=int(input("enter the second side"))
c=int(input("enter the third side"))
if a+b>c or a+c>b :
    print("triangle")
elif b+c>a:
    print ("triangle")
else:
    print("not triangle")
```

```
enter the first side6
enter the second side3
enter the third side5
triangle
```

In [8]:
```
#sum of n numbers till you press enter
sum=0
data=input("enter the number")
while data !="":
    n1=float(data)
    sum=sum+n1
    data=input("enter the number")
print("sum is",sum)
```

```
enter the number5
enter the number4
enter the number2
enter the number6
enter the number
sum is 17.0
```

In [9]:
```
#sum of first 10 natural numbers
sum=0
count=1
while count<=10:
    sum=sum+count
    count+=1
print ("sum is",sum)
```

```
sum is 55
```

In [ ]:
```
#while with else
count=1
while count<=10:
    print(count)
    count+=1
else:
    print("reached limit")
```

```
1
2
3
4
5
6
7
8
9
10
reached limit
```

In [ ]:
```
#switch-dictionary
dict={1:"one",2:"two"}
print (dict.get(2,"fault"))
```

```
two
```

In [10]:
```
#switch
def sw(case):
        dict={1:"one",2:"two"}
        return dict.get(case,"invalid")
x=sw(1)
print (x)
print(sw(4))
```

```
one
invalid
```

In [11]:
```
#range operator
for i in range(6):
    print (i)
```

```
0
1
2
3
4
5
```

In [12]:
```
#in operator
for i in "python":
    print (i)
```

```
p
y
t
h
o
n
```

In [13]:
```
l1=["apple","orange","grapes",1]
for i in l1:
    print (i)
```

```
apple
orange
grapes
1
```

In [14]:
```
for i in [1,2,4]:
    print (i)
```

```
1
2
4
```

In [15]:
```
for i in range(6,20):
    print (i)
```

```
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

In [16]:
```
for i in range(6,20,2):
    print (i)
```

```
6
8
10
12
14
16
18
```

In [ ]:
```
for i in range(6):
    print (i)
else:
    print("iteration over")
```

```
0
1
2
3
4
5
iteration over
```

In [ ]:
```
#break
for x in range(6):
  if x == 3:
    break
  print(x)
else:
  print("Finally finished!")
```

```
0
1
2
```

In [17]:
```
#continue
for x in range(6):
  if x == 3:
    continue
  print(x)
else:
  print("Finally finished!")
```

```
0
1
2
4
5
Finally finished!
```

In [18]:
```
for count in range(5):
    print(count + 1, end = " ")
```

```
1 2 3 4 5
```

In [ ]:
```
for count in range(1, 4):
    print(count, end = " ")
```

```
1 2 3
```

In [ ]:
```
for count in range(1, 6, 2):
    print(count, end = " ")
```

```
1 3 5
```

In [19]:
```
for count in range(6, 1, -1):
    print(count, end = " ")
```

```
6 5 4 3 2
```

In [20]:
```
for letter in 'Python':
    if letter == 'h':
        break
    print(letter)
```

```
P
y
t
```

In [21]:
```
for i in range(10):
        pass
```

In [22]:
```
#reverse
rev=0
print("enter number")
num=int(input())
while num!=0:
    d=num%10
    rev=rev*10+d
    num=num//10
print(rev)
```

```
enter number
567
765
```

In [23]:
```
#fibonocci of 10 numbers
a=0
b=1
print(a,b,end=" ")
for i in range(10-2):
    c=a+b
    a=b
    b=c
    print(c,end=" ")
```

```
0 1 1 2 3 5 8 13 21 34
```

In [24]:
```
#prime numbers
print("prime numbers less than 100")
for n in range(2,100):
    i=2
    while i<=n/2:
        if n%i==0:
            break
        i=i+1
    else:
        print(n,end=" ")
```

```
prime numbers less than 100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

In [25]:
```
#pattern printing
for n in range(0,6):
    for i in range(1,n+1):
        print (i,end="   ")
    print("\n")
```

```
1

1   2

1   2   3

1   2   3   4

1   2   3   4   5
```

In [26]:
```
#format output
"%-10.3f" % 3.14
```

Out[26]:
```
'3.140     '
```

In [27]:
```
amount=24.325
print("Your salary is $%0.2f" % amount)
print("The area is %0.1f" % amount)
```

```
Your salary is $24.32
The area is 24.3
```

In [29]:
```
for exponent in range(7, 11):
  print("%-3d%12d" % (exponent, 10 ** exponent))
```

```
7       10000000
8      100000000
9     1000000000
10   10000000000
```

In [30]:
```
#factorial
n=int(input("enter the number"))
fact=1
while n>0:
    fact=fact*n
    n-=1
print(fact)
```

```
enter the number4
24
```

In [ ]:
```
#armstrong
n=int(input("enter the number"))
i=0
c=b=n
armstr=0
while(n>0):
    n=n//10
    i+=1

while(b>0):
    r=b%10
    armstr=armstr+r**i
    b=b//10
if(armstr==c):
    print("armstrong")
else:
    print("not")
```

```
enter the number121
not
```

In [ ]:
```
#armstrong series
n=int(input("enter the range"))
for s in range(n):
    i=0
    c=b=s
    armstr=0
    while(s>0):
        s=s//10
        i+=1

    while(b>0):
        r=b%10
        armstr=armstr+r**i
        b=b//10
    if(armstr==c):
        print(c,end=" ")
```

```
enter the range10000
0 1 2 3 4 5 6 7 8 9 153 370 371 407 1634 8208 9474
```

In [31]:
```
#biggest and largest among n numbers
n=int(input("enter the range"))
max=1
min=1
while(n>0):
    s=int(input("enter the number"))
    if(s>max):
        max=s
    elif(s<=min):
        min=s
    n-=1
print("max",max,"min",min)
```

```
enter the range4
enter the number3
enter the number6
enter the number8
enter the number2
max 8 min 1
```

In [32]:
```
#series 1 2 4 7 11 16...
s=int(input("enter the range"))
num=1
for i in range(s):
    num=num+i
    print(num,end=" ")
```

```
enter the range30
1 2 4 7 11 16 22 29 37 46 56 67 79 92 106 121 137 154 172 191 211 232 2
54 277 301 326 352 379 407 436
```

In [ ]:

```
#multiplication table of n numbers
s=int(input("enter the range"))
for i in range(1,s+1):
    for j in range(1,11):
        print(i ,"*",j,"=",i*j)
    print("\t")
```

```
enter the range10
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

6 * 1 = 6
```

6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60

7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70

8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90

10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100

In [ ]:

```
# simple function
def my_func():
  print("Hello! Hope you're doing well")
my_func()
```

```
Hello! Hope you're doing well
```

In [ ]:

```
#function call can be before function defintion
my_func()
def my_func():
  print("Hello! Hope you're doing well")
```

```
Hello! Hope you're doing well
```

In [ ]:

```
#function with argumets
def my_func(name,place):
  print(f"Hello {name}! Are you from {place}?")
my_func("john","chennai")
```

```
Hello john! Are you from chennai?
```

In [ ]:

```
#types of functions
#defining function no return value and no parameter
def sum():
  a=int(input("number1"))
  b=int(input("number2"))
  print(a + b)
sum()
```

```
number12
number23
5
```

In [ ]:

```
#defining function with parameter and  no return value
def sum(x,y):
    print(x + y)
a=int(input("number1"))
b=int(input("number2"))
sum(a,b)
```

```
number13
number22
5
```

In [ ]:

```
#defining function with  no parameter but with return value
def sum():
    a=int(input("number1="))
    b=int(input("number2="))
    return (a + b)

c=sum()
print("sum is {}".format(c))
```

```
number1=3
number2=2
sum is 5
```

In [ ]:

```

```

In [ ]:

```
#defining function with  parameter and return value
def sum(a, b):
  return a + b
result = sum(1, 2)
print(result)
```

```
3
```

In [ ]:

```
#Pass by Reference vs Value
#mutable data objects are passed by reference its value will change
def funct1(a):
  a[0]=100
a=[1,2,3]
funct1(a)
print(a)

#strings are immutable.so its value wont change
def f(s):
    s='cek'
    print (s)
s='mec'
f(s)
print(s)
```

```
[100, 2, 3]
cek
mec
```

In [ ]:

```
#function to find maximum of 2 numbers
def maximum(a,b):
  if(a>b):
    print("largest is ",a)
  else:
    print("largest is ",b)

x=int(input("number1="))
y=int(input("number2="))
maximum(x,y)
```

```
number1=4
number2=3
largest is  4
```

In [ ]:

```
#function to find  the absolute value of a number
def absval(a):
  if(a>0):
    print(a)
  else:
    print(a*-1)

x=int(input("number1="))
absval(x)
```

```
number1=-3
3
```

In [ ]:

```
#different types of formal arguments -Variable-length arguments"""
#1)Required arguments are the arguments passed to a function in correct positional orde
r.
#Here, the number of arguments in the function call should match exactly with the funct
ion definition.
def hello(s,msg):
  print( "the person is {} and message is{}" .format(s,msg))
hello("mark","how are you")
#arguments should be in correct positional order.

def hello2(msg,s):
  print("the person is {} and message is{}" .format(s,msg))
hello2("mark","how are you")
```

```
the person is mark and message ishow are you
the person is how are you and message ismark
```

In [ ]:

```
#positional arguments shows error if one value is missing
def hello3(s,msg):
  print( s,msg)
hello3("mark")
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-22-7b6427e53925> in <module>()
      2 def hello3(s,msg):
      3   print( s,msg)
----> 4 hello3("mark")

TypeError: hello3() missing 1 required positional argument: 'msg'
```

In [ ]:

```
#2)Default arguments
def hello(name="mark",msg="how are you"):
  print("the person is {} and message is {}" .format(name,msg))
hello("john","welcome")
hello('peter')
hello()
```

```
the person is john and message is welcome
the person is peter and message is how are you
the person is mark and message is how are you
```

In [ ]:

```
#3)Keyword arguments
def hello(name,msg):
  print(name,msg)
hello(name="john",msg="hai")
hello(msg="hello",name="edwin")
```

```
john hai
edwin hello
```

In [ ]:

```
#4)arbitary arguments
def hello(*names):
     for name in names:
         print("Hello", name)

hello("adam","mark","john")
```

```
Hello adam
Hello mark
Hello john
```

In [ ]:

```
#add sum of n numbers passed
def add(*x):
  sum=0
  for i in x:
    sum=sum+i
  print(sum)
add(1,2,3)
add(1,4,3,2,4,6,7)
```

```
6
27
```

In [ ]:

```
#map function
integer=["23","12",'11']
newlist=map(int,integer)
print(newlist)
print(list(newlist))
```

```
<map object at 0x7f6f8a198e10>
[23, 12, 11]
```

In [ ]:

```
#reduce function
from functools import reduce
def add(x,y):
  return(x+y)
def mul(x,y):
  return(x*y)
data=[8,4,2]
addval=reduce(add,data)
mulval=reduce(mul,data)
print(addval,"summation of the list")
print (mulval,"product of the list")
```

```
14 summation of the list
64 product of the list
```

In [ ]:

```
#lambda function
z=lambda x,y:x+y
print ((z(2,3)))
```

```
5
```

In [ ]:

```
#filter function:a function(predicate) is applied to each value in the list.
#if the value returns true its added  to the filter object.otherwised dropped
def odd(n):
  return n%2==1
f1=filter(odd,range(10))
print(list(f1))
```

```
[1, 3, 5, 7, 9]
```

In [ ]:

```
# variable declared outside of the function or in global scope is known as a global var
iable.
#This means that a global variable can be accessed inside or outside of the function.
x = 100
def fUNCTION1():
     x=1
     print("x inside VALUE:", x)

fUNCTION1()
print("x outside:", x)
```

```
x inside VALUE: 1
x outside: 100
```

In [ ]:

```
def fUNCTION1():
     x=1
     print("x inside VALUE:", x)

fUNCTION1()
print("x outside:", x)
```

```
x inside VALUE: 1

---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-1-8c850780b30a> in <module>()
      5
      6 fUNCTION1()
----> 7 print("x outside:", x)

NameError: name 'x' is not defined
```

In [ ]:

```
#Global Variables can be modified in functions with the keyword global
def fUNCTION1():
    global x
    x=1
    print("x inside VALUE:", x)

fUNCTION1()
print("x outside:", x)
```

```
x inside VALUE: 1
x outside: 1
```

In [ ]:

```
#factorial of a number
def fact(n):
  sum=0
  if(n==1):
    return 1
  else:
    return n*fact(n-1)

n=int(input("enter the number"))
sum=fact(n)
print("factorial of the number is ",sum)
```

```
enter the number5
factorial of the number is  120
```

In [ ]:

```
#nth finonocci number
def fib(n):
  if (n==1 or n==2):
    return 1
  else:
    return fib(n-1)+fib(n-2)
n=int(input("enter the number"))
sum=fib(n)
print("fibonocci number is ",sum)
```

```
enter the number6
fibonocci number is  8
```

In [ ]:
```
#create a list
l1=[1,2,3,4,5]
l2=list(range(1,5))
print(l1,l2)
```

```
[1, 2, 3, 4, 5] [1, 2, 3, 4]
```

In [ ]:
```
#len function
basket=[1,2,3,4]
len(basket)
```

Out[ ]:
```
4
```

In [ ]:
```
#slicing in list and creating a newlist
basket=[1,2,3,4,100]
newbasket=basket[2:5]
print(newbasket)
newbasket=basket[::-1]
print(newbasket)
```

```
[3, 4, 100]
[100, 4, 3, 2, 1]
```

In [ ]:
```
#combining list
l1=[1,2,3]
print(l1)
l2=["mon","tues","wed"]
print(l2)
print(l1+l2)
```

```
[1, 2, 3]
['mon', 'tues', 'wed']
[1, 2, 3, 'mon', 'tues', 'wed']
```

In [ ]:
```
#creating a list from a astring
print("hai")
print (list("hai"))
```

```
hai
['h', 'a', 'i']
```

In [ ]:
```
#slicing in list
listnew=[1,2,3]
print(listnew[0])
print(listnew[0:])
```

```
1
[1, 2, 3]
```

In [ ]:
```
#to print without bracket and commas
for number in [1, 2, 3, 4]:
    print(number, end = " ")
```

```
1 2 3 4
```

In [ ]:
```
#in and not in  operator in list to detect the presence or absence of
 element in list
print(3 in [1, 2, 3])
print(0 in [1, 2, 3])
print(0 not in  [1,2,3])
basket=[1,2,3,4,100]
print(1 in basket)
```

```
True
False
True
True
```

In [ ]:
```
#replace an element
example = [1, 2, 3, 4]
print(example)
example[3] = 0
print(example )
```

```
[1, 2, 3, 4]
[1, 2, 3, 0]
```

In [ ]:
```
#replace an element
numbers=[2,3,4,5]
for index in range(len(numbers)):
    numbers[index] = numbers[index] ** 2
print(numbers)
```

```
[4, 9, 16, 25]
```

In [ ]:
```
#converting a string to list and makin it upper
sentence = "This example has five words."
words = sentence.split()
print (words)
for index in range(len(words)):
   words[index] = words[index].upper()
print(words)
```

```
['This', 'example', 'has', 'five', 'words.']
['THIS', 'EXAMPLE', 'HAS', 'FIVE', 'WORDS.']
```

In [ ]:
```
#addig elements to list
#1)append
basket=[1,2,3,4]
print(basket.append(100))
print (basket)
#2)insert
basket=[1,2,3,4]
print(basket.insert(2,100))
print(basket)
#3)extend
basket=[1,2,3,4]
print(basket.extend([100,200]))
print (basket)
```

```
None
[1, 2, 3, 4, 100]
None
[1, 2, 100, 3, 4]
None
[1, 2, 3, 4, 100, 200]
```

In [ ]:
```
#index
basket=[1,2,3,4,100]
print(basket.index(3))
print (basket)
print(basket.index(3,0,4))
```

```
2
[1, 2, 3, 4, 100]
2
```

In [1]:
```
#index
basket=[1,2,3,4,100]
print(basket.index(3))
print (basket)
print(basket.index(3,0,4))
print(basket.index(4,0,2))
```

```
2
[1, 2, 3, 4, 100]
2

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call l
ast)
<ipython-input-1-1bd7d2deab60> in <module>()
      4 print (basket)
      5 print(basket.index(3,0,4))
----> 6 print(basket.index(4,0,2))

ValueError: 4 is not in list
```

In [ ]:
```
#remove methods
#1)pop-give index,if not specified last elemnt will be removed
basket=[1,2,3,4,100]
print(basket.pop())
print (basket)
print(basket.pop(2))
print(basket)
#2)remove-give element to remove
basket=[1,2,3,4]
print(basket.remove(4))
print(basket)
#3)clear-removes everythings from list
basket=[1,2,3,4]
print(basket.clear())
print(basket)
```

```
100
[1, 2, 3, 4]
3
[1, 2, 4]
None
[1, 2, 3]
None
[]
```

In [ ]:
```
#sorting a list
basket=[3,2,1,5]
print(basket.sort())
print (basket)
```

```
None
[1, 2, 3, 5]
```

In [ ]:
```
#copy and reverse a list
basket=[3,2,1,5]
newbasket=basket.copy()
print (newbasket)
print(newbasket.reverse())
print(newbasket)
```

```
[3, 2, 1, 5]
None
[5, 1, 2, 3]
```

In [ ]:
```
#count
basket=[1,2,3,2,1,1]
print(basket.count(1))
```

```
3
```

In [ ]:
```
#program to search an element and display its index
element=int(input("enter the element to search"))
n=int(input("enter the size of list"))
list1=[]
for i in range(0,n):
  l=int(input("enter the element"))
  list1.append(l)
print (list1)
if (element in list1):
  print(element,"is at position",list1.index(element))
else:
  print("couldnt find")
```

```
enter the element to search2
enter the size of list2
enter the element1
enter the element2
[1, 2]
2 is at position 1
```

In [ ]:
```
# program to find sum of even numbers from a group of numbers
n=int(input("enter the limit"))
list1=[]
 vsum=0
for i in range(0,n):
  l=int(input("enter the element"))
  list1.append(l)
for i in list1:
  if (i%2==0):
    sum+=i
print("sum",sum)
```

```
enter the limit5
enter the element2
enter the element7
enter the element3
enter the element2
enter the element1
sum 4
```

In [ ]:
```
#Program to remove all duplicate elements from a list
n=int(input("enter the limit"))
list1=[]
list2=[]
sum=0
for i in range(0,n):
  l=int(input("enter the element"))
  list1.append(l)
for i in 2
list1:
  if i not in list2:
    list2.append(i)
print("new list after removing duplicates",list2)
```

```
enter the limit5
enter the element1
enter the element2
enter the element3
enter the element1
enter the element2
new list after removing duplicates [1, 2, 3]
```

In [ ]:
```
#Read a string and print the words in alphabetical order
str1=input("enter the string")
l1=str1.split()
print (l1)
l1.sort()
print (l1)
```

```
enter the stringhello how are you
['hello', 'how', 'are', 'you']
['are', 'hello', 'how', 'you']
```

In [ ]:
```
#CREATE A DICTIONARY-1) curly braces, {}   2) built-in function  dict
() function.
#1)CURLY BRACES
mydict={}
print (type(mydict))

#2)using dict()
mydict=dict()
print (type(mydict))
```

```
<class 'dict'>
<class 'dict'>
```

In [ ]:
```
# keys in the dictionary are Boolean, integer, floating point number,
 and string data types, which are all acceptable.Dictionary keys canno
t be of a type that is mutable, such as sets, lists, or dictionaries.
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}

print(my_dictionary)
#key with list type are not supported
my_dictionary = {["Python"]: "languages"}

print(my_dictionary)
```

```
{True: 1, 1.1: 1.1, 'one': 1, 'languages': ['Python']}

---------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-1-7ed421803986> in <module>()
      4 print(my_dictionary)
      5 #key with list type are not supported
----> 6 my_dictionary = {["Python"]: "languages"}
      7
      8 print(my_dictionary)

TypeError: unhashable type: 'list'
```

In [ ]:
```
#creating a dictionary with items using {}
mydict={'name':"john",'age':45,'job':"doctor"}
print(mydict)
#create a dictionary with dict()
mydict = dict({'name': 'john' ,'age':45,'job':"doctor"})
print(mydict)
```

```
{'name': 'john', 'age': 45, 'job': 'doctor'}
{'name': 'john', 'age': 45, 'job': 'doctor'}
```

In [ ]:
```
#creating a dictionary using fromkeys() without setting a value for al
l the keys:
#create sequence of strings
cities = ('Paris','Athens', 'Madrid')
#create the dictionary, `my_dictionary`, using the fromkeys() method
my_dictionary = dict.fromkeys(cities)
print(my_dictionary)
```

```
{'Paris': None, 'Athens': None, 'Madrid': None}
```

In [ ]:
```
#create a sequence of strings
cities = ('Paris','Athens', 'Madrid')
#create a single value
continent = ('Europe')
my_dictionary = dict.fromkeys(cities,continent)

print(my_dictionary)
```

```
{'Paris': 'Europe', 'Athens': 'Europe', 'Madrid': 'Europe'}
```

In [ ]:
```
#len() function returns the total length of the object that is passed
 as an argument.(no of key value pair)
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (len(my_dictionary))
```

```
4
```

In [ ]:
```
#dictionaries can be created from list
L1=[1,2,3]
L2=[10,20,30]
dictionary1=dict(zip(L1,L2))
print (dictionary1)
```

```
{1: 10, 2: 20, 3: 30}
```

In [ ]:
```
#1)View All key-value Pairs Contained in a Dictionary in Python
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (my_dictionary.items())
#2)View All keys Contained in a Dictionary in Python
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (my_dictionary.keys())
#3)#View All values Contained in a Dictionary in Python
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (my_dictionary.values())
```

```
dict_items([(True, 1), (1.1, 1.1), ('one', 1), ('languages', ['Pytho
n'])])
dict_keys([True, 1.1, 'one', 'languages'])
dict_values([1, 1.1, 1, ['Python']])
```

In [ ]:
```
#how to access an item in a Python dictionary:
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (my_dictionary["one"])
#key not in dictionary
print (my_dictionary["two"])
```

```
1

---------------------------------------------------------------------------
----
KeyError                                  Traceback (most recent call l
ast)
<ipython-input-24-0c92bab679da> in <module>()
      3 print (my_dictionary["one"])
      4 #key not in dictionary
----> 5 print (my_dictionary["two"])

KeyError: 'two'
```

In [ ]:
```
#in keyword  returns True if the key is in the dictionary and False if
it isn't.
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print ("one" in my_dictionary)
print ("two" in my_dictionary)
```

```
True
False
```

In [ ]:
```
#Another way around this is to access items in the dictionary by using
the get() method.
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (my_dictionary.get("one"))
print (my_dictionary.get("two","no such key"))
```

```
1
no such key
```

In [ ]:
```
#Add New Items to A Dictionary in Python
my_dictionary = {}

#add a key-value pair to the empty dictionary
my_dictionary['name'] = "John "
# add another  key-value pair
my_dictionary['age'] = 34

#print dictionary
print(my_dictionary)
my_dictionary['age'] = 46

#the value of 'age' will now be updated

print(my_dictionary)
```

```
{'name': 'John ', 'age': 34}
{'name': 'John ', 'age': 46}
```

In [ ]:
```
#To update a dictionary, you can also use the dictionary method update
() .
my_dictionary={'name': 'John ', 'age': 34}
my_dictionary.update(name= 'Mike Green', age = 46, occupation = "softw
are developer")
print(my_dictionary)
#update method to combine two dictionaries
numbers = {'one': 1, 'two': 2, 'three': 3}
more_numbers = {'four': 4, 'five': 5, 'six': 6}

#update 'numbers' dictionary
#you update it by adding the contents of another dictionary, 'more_num
bers',
#to the end of it
numbers.update(more_numbers)

print(numbers)
```

```
{'name': 'Mike Green', 'age': 46, 'occupation': 'software developer'}
{'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5, 'six': 6}
```

In [ ]:
```
#copy method
my_dictionary={'name': 'John ', 'age': 34}
newdict=my_dictionary.copy()
print(newdict)
```

```
{'name': 'John ', 'age': 34}
```

In [ ]:
```
#delete a key-value pair
my_dictionary={'name': 'John ', 'age': 34}
del my_dictionary['age']

print(my_dictionary)
#remove a key value pair and store it using pop method
my_dictionary={'name': 'John ', 'age': 34}
value=my_dictionary.pop('age')
print(value)
#pop with a default value
my_dictionary={'name': 'John ', 'age': 34}
value=my_dictionary.pop('job',"not in dictionary")
print(value)
#remove last item from dictionary-popitem
my_dictionary={'name': 'John ', 'age': 34}
value=my_dictionary.popitem()
#delete an element from dictionary
my_dictionary={'name': 'John ', 'age': 34}
del(my_dictionary['name'])
print(my_dictionary)
print(value)
```

```
{'name': 'John '}
34
not in dictionary
('age', 34)
```

In [ ]:
```
#delete all items-clear method
my_dictionary={'name': 'John ', 'age': 34}
my_dictionary.clear()
print(my_dictionary)
```

```
{}
```

In [ ]:
```
#1)traversing a dictionary
my_dictionary={'name': 'John ', 'age': 34}
print(my_dictionary)
for i in my_dictionary:
  print(i,my_dictionary[i])
#using items
my_dictionary={'name': 'John ', 'age': 34}
print(my_dictionary.items())
for i,j in my_dictionary.items():
  print (i,j)
```

```
{'name': 'John ', 'age': 34}
name John
age 34
dict_items([('name', 'John '), ('age', 34)])
name John
age 34
```

In [ ]:
```
#sort-convert the dictionary to a list and use sort method
my_dictionary={'name': 'John ', 'age': 34}
l1=list(my_dictionary.keys())
l1.sort()
print (l1)
for i in l1:
  print(i,my_dictionary[i])

#2)using build in function called sorted return value is sorted list o
f keys
my_dictionary={'name': 'John ', 'age': 34}
newdict=sorted(my_dictionary)
print(newdict)
```

```
['age', 'name']
age 34
name John
['age', 'name']
```

In [ ]:
```
#dictionary membership function
my_dictionary={'name': 'John ', 'age': 34}
print('name'in my_dictionary)
print('job'in my_dictionary)
```

```
True
False
```

In [ ]:
```
#Dictionary Comprehension
 squares = {x: x*x for x in range(6)}
 print(squares)
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [ ]:
```
#entering key-value pair at runtime
dict1={}
n=int(input("enter number of elements in dictionary"))
for i in range(0,n):
  key=int(input("key"))
  dict1[key]=int(input("value"))
print(dict1)
```

```
enter number of elements in dictionary5
key1
value2
key1
value5
key2
value2
key4
value2
key6
value1
{1: 5, 2: 2, 4: 2, 6: 1}
```

In [ ]:
```
#key-value at runtime
str1=input("enter the string")
dict1={}
for i in str1:
  dict1[i]=(int(input("enter value")))
print (dict1)
```

```
enter the stringhello
enter value1
enter value3
enter value1
enter value5
enter value7
{'h': 1, 'e': 3, 'l': 5, 'o': 7}
```

In [ ]:
```
#Program to count the number of occurrence(frequency) of each letters
 in a given string( histogram)
str1=input("enter the string")
dict1={}
for i in str1:
  dict1[i]=(dict1.get(i,0)+1)
print (dict1)
```

```
enter the stringhello
{'h': 1, 'e': 1, 'l': 2, 'o': 1}
```

In [ ]:
```
#Program to display the frequency of each word in a given string
str1=input("enter the string")
dictnw={}
l1=str1.split()
for i in l1:
  dictnw[i]= dictnw.get(i,0)+1
print(dictnw)
```

```
enter the stringhello hai hello
{'hello': 2, 'hai': 1}
```

In [ ]:
```
#Write a Python program to create a dictionary of roll numbers and nam
es of five students.
#sort the roll nubers(sorting by converting to a list)
dict1={}
for i in range(0,3):
  roll=int(input("roll"))
  dict1[roll]=input("name")
print (dict1)
l2=list(dict1)
print(l2)
l2.sort()
for i in l2:
  print(i,dict1[i])
```

```
roll2
namefg
roll1
namesd
roll11
namef
{2: 'fg', 1: 'sd', 11: 'f'}
[2, 1, 11]
1 sd
2 fg
11 f
```

In [ ]:
```
#sortig a dictionary using buildin sorted method
dict1={}
for i in range(0,3):
  roll=input("roll")
  dict1[roll]=input("name")
print (dict1)
for i in sorted(dict1):
  print(i,dict1[i])
```

```
roll4
namedf
roll1
namehj
roll4
namedd
{'4': 'dd', '1': 'hj'}
1 hj
4 dd
```

In [ ]:

```
#hex to binary conversion
hextobin={'0':'0000','1':'0001','2':'0010','3':'0011','4':'0100','5':
'0101','6':'0110','7':'0111','8':'1000','9':'1001','A':'1010','B':'101
1','C':'1100','D':'1101','E':'1110','F':'1111'}
n=input('Enter the hexadecimal number….')
bn=''
n=n.upper()
for i in n:
    h=hextobin.get(i)
    if h==None:
       print('Invalid Number')
       break
    print("binary equilavent of the heaxdecimal is {} is {}".format(n,
h))
```

```
Enter the hexadecimal number….A
binary equilavent of the heaxdecimal is A is 1010
```

In [ ]:

```
#tuples
x=(1,2,3)
x[2]=7
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-4-87f45717a5a3> in <module>()
      1 #tuples
      2 x=(1,2,3)
----> 3 x[2]=7

TypeError: 'tuple' object does not support item assignment
```

In [ ]:

```
t = ("tuples", "are", "immutable")
#To create a tuple with a single element, we have to include the final comma:
t1 = ('a',)
print(type(t1))
t2 = ('a')
print(type(t2))
```

```
<class 'tuple'>
<class 'str'>
```

In [ ]:

```
#tuple indexing
a=(1,2,3,4)
print (a[0])
```

```
1
```

In [ ]:

```
#A tuple can also be created without using parentheses. This is known as tuple packing.
t = 3, 4.6, "cat"
print(t)
```

```
(3, 4.6, 'cat')
```

In [ ]:

```
#We can swap values of two tuples
t1=(10,20,30)
t2=(100,200)
t2,t1=t1,t2
print (t2)
```

```
(10, 20, 30)
```

In [ ]:

```
#tuples-concatination adding new elements to tuple
x=(1,2,3)
y=('a','b','c')
print (x+y)
z=(3,)
print(x+z)
```

```
(1, 2, 3, 'a', 'b', 'c')
(1, 2, 3, 3)
```

In [ ]:

```
#list to tuples and tuples to list
l1=[1,2,3]
print (tuple(l1))
t1=(1,2,3)
print (list(t1))
```

```
(1, 2, 3)
[1, 2, 3]
```

In [ ]:

```
#TUPLE METHODS AND BUILDIN FUNCTIONS
#zip(list of tuples)
l1=(1,2,3)
l2=(3,4,5)
l3=zip(l1,l2)
print(type(l3))
print (tuple(l3))
```

```
<class 'zip'>
((1, 3), (2, 4), (3, 5))
```

In [ ]:

```
#count,index,max and min ,del
l1=(1,2,3,1,1,3)
print(l1.count(1))
print(l1.index(3))
print(max(l1))
print (min(l1))
del(l1)
```

```
3
2
3
1
```

In [ ]:

```
2#Program to read numbers and find minimum, maximum and sum using Tuple
n=int(input("Enter how many numbers...."))
print("Enter the numbers")
t=tuple()
for i in range(n):
    x=int(input())
    t=t+(x,)
print("minimum=",min(t))
print("maximum=",max(t))
print("sum=",sum(t))
```

```
Enter how many numbers....4
Enter the numbers
1
5
8
3
minimum= 1
maximum= 8
sum= 17
```

In [ ]:

```
#enter elements to tuple at runtime..convert tuple to list use append function and then
list to tuple
t=tuple()
l1=list(t)
print("enter no of elements")
n=int(input())
for i in range(0,n):
  l1.append(int(input()))
print (tuple(l1))
```

```
enter no of elements
4
2
7
1
2
(2, 7, 1, 2)
```

In [ ]:

```
#tuple unpacking
(a,b,c)=(1,2,3)
print(a)
(a,*b,c)=(1,2,3,4,5)
print (c)
print(a)
```

```
1
5
1
```

In [ ]:

```
a=(10,20,30,40)
b=(1,2)
(b,a)=(a,b)
print (a)
print(b)
```

```
(1, 2)
(10, 20, 30, 40)
```

In [ ]:

```
#A set is an unordered collection of items.
#Every set element is unique (no duplicates) and must be immutable (cannot be changed).
#However, a set itself is mutable. We can add or remove items from it.
#Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc.
```

In [ ]:

```
#declaring a set
S={1,2,3}
print(S)
```

```
{1, 2, 3}
```

In [ ]:

```
S = {1.0, "Hello", (1, 2, 3)}
print(S)
```

```
{1.0, 'Hello', (1, 2, 3)}
```

In [ ]:

```
#set will not have duplicates
 S = {1, 2, 3, 4, 3, 2}
 print(S)
```

```
{1, 2, 3, 4}
```

In [ ]:

```
#set can be created from a list
l=[1,2,3,4,4]
S=set(l)
print(S)
```

```
{1, 2, 3, 4}
```

In [ ]:

```
#set cannot have mutable item like list
S={1,2,[3,4]}
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-6-9977531ccb41> in <module>()
      1 #set cannot have mutable item
----> 2 S={1,2,[3,4]}

TypeError: unhashable type: 'list'
```

In [ ]:

```
#Creating an empty set
S=set()
print(S)
```

```
set()
```

In [ ]:

```
#We can add a single element using the add() method, and multiple elements using the up
date() method.
#The update() method can take tuples, lists, strings or other sets as its argument. In
 all cases, duplicates are avoided
```

In [ ]:

```
S={1,3}
print(S)
S.add(4)
print(S)
S.update([2,3,5])
print(S)
S.update([7,8],{9,10})
print(S)
```

```
{1, 3}
{1, 3, 4}
{1, 2, 3, 4, 5}
{1, 2, 3, 4, 5, 7, 8, 9, 10}
```

In [ ]:

```
#Removing elements from a set
#A particular item can be removed from a set using the methods discard() and remove().
#The only difference between the two is that the discard() function leaves a set unchan
ged
#if the element is not present in the set.
#On the other hand, the remove() function will raise an error in such a condition (if e
lement is not present in the set).
S={1,2,3,4}
S.discard(4)
print(S)
S.remove(3)
print(S)
S.discard(3)
print(S)
S.remove(5)
```

```
{1, 2, 3}
{1, 2}
{1, 2}

---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
<ipython-input-11-0110ff0be21e> in <module>()
     11 S.discard(3)
     12 print(S)
---> 13 S.remove(5)
     14

KeyError: 5
```

In [ ]:

```
#we can remove and return an item using the pop() method.
#Since set is an unordered data type, there is no way of determining which item will be
popped. It is completely arbitrary
S={1,5,2,3,4}
print(S.pop())
print(S.pop())
```

```
1
2
```

In [ ]:

```
#union operation
A = {1, 2, 3, 4}
B = {4, 5, 6, 7, 8}
print(A|B)
print(A.union(B))

print(B.union(A))
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [ ]:

```
#intersection
print(A&B)
print(A.intersection(B))
print(B.intersection(A))
```

```
{4}
{4}
{4}
```

In [ ]:

```
#Difference of the set B from set A(A - B) is a set of elements that are only in A but
 not in B.
#Similarly, B - A is a set of elements in B but not in A
print(A-B)
print(B-A)
print(A.difference(B))
print(B.difference(A))
```

```
{1, 2, 3}
{8, 5, 6, 7}
{1, 2, 3}
{8, 5, 6, 7}
```

In [ ]:

```
#Symmetric Difference of A and B is a set of elements in A and B but not in both (exclu
ding the intersection).
#Symmetric difference is performed using ^ operator.
print(A^B)
print(B^A)
print(A.symmetric_difference(B))
```

```
{1, 2, 3, 5, 6, 7, 8}
{1, 2, 3, 5, 6, 7, 8}
{1, 2, 3, 5, 6, 7, 8}
```

In [ ]:
```
#strings ae immutable
STR1="HELLO"
STR1[1]="J"
```

```
---------------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-3-015f153cd62a> in <module>()
      1 #strings ae immutable
      2 STR1="HELLO"
----> 3 STR1[1]="J"

TypeError: 'str' object does not support item assignment
```

In [ ]:
```
name = "Alan Turing"
name[0]
```

Out[ ]:
```
'A'
```

In [ ]:
```
name = "Alan Turing"
print(name[0])
print(len(name))
print (name[len(name)-1])
```

```
A
11
g
```

In [ ]:
```
print(name[-1])
print(name[-3])
print(name[-5])
```

```
g
i
u
```

In [ ]:
```
name="hello"
print(name[0:-2:1])
```

```
hel
```

In [ ]:
```
name="hello"
print(name[::1])
```

```
hello
```

In [ ]:
```
name="hello"
print(name[::-1])
```

```
olleh
```

In [ ]:
```
name="hello"
print(name[-2:0:-1])
```

```
lle
```

In [ ]:
```
name="hello"
print(name[-3:])
```

```
llo
```

In [ ]:
```
name="hello"
print(name[-3::-1])
```

```
leh
```

In [ ]:
```
fileList = ["myfile.txt", "myprogram.exe", "yourfile.txt"]
 for fileName in fileList:
    if ".txt" in fileName:
        print(fileName)
```

```
myfile.txt
yourfile.txt
```

In [ ]:
```
data="myprogram.exe"
print(data[2])
print(data[-1])
print(len(data))
print(data[0:8])
```

```
p
e
13
myprogra
```

In [1]:
```
#string methods
data="hello"
print(data.center(20),"*")
print(data)
print(data.endswith("lo"))
print(data.find("llo"))
```

```
       hello         *
hello
True
2
```

In [ ]:
```
data="hello"
print(data.center (20))
print(data.center (20,"*"))
print(data)
```

```
       hello
*******hello********
hello
```

In [ ]:
```
print(data.count("l"))
print(data.count("l",0,2))
print(data.count("l",0,3))
```

```
2
0
1
```

In [ ]:
```
#find returns index
print(data.find("o"))
print(data.find("o",0,2))
```

```
4
-1
```

In [ ]:
```
print(data.isalpha())
print(data.isdigit())
```

```
True
False
```

In [2]:
```
l1=["apple","orange","grapes"]
print("".join(l1))
print(l1)
print("and".join(["a","b","c"]))
```

```
appleorangegrapes
['apple', 'orange', 'grapes']
aandbandc
```

In [ ]:
```
#for join the list should be strings
l1=[1,2,3]
print("-".join(l1))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-21-61ec243e10f1> in <module>()
      1 l1=[1,2,3]
----> 2 print("-".join(l1))

TypeError: sequence item 0: expected str instance, int found
```

In [ ]:
```
str1="helloHOW"
print(str1.lower())
```

```
hellohow
```

In [ ]:
```
str1="helloHOWel"
print(str1.replace("el","me"))
print(str1)
```

```
hmeloHOWme
helloHOWel
```

In [ ]:
```
str1="helloHOWelel"
print(str1.replace("el","me",2))
```

```
hmeloHOWmeel
```

In [ ]:
```
str1="     hello    "
print(str1.split())
```

```
['hello']
```

In [ ]:
```
str1="*****hello*** "
print(str1.split("l"))
```

```
['*****he', '', 'o*** ']
```

In [ ]:
```
str1="helloHOW"
print(str1.startswith("hello"))
```

```
True
```

In [ ]:
```
str1="     hello    "
print(str1.lstrip())
```

```
hello
```

In [3]:
```
str1="   thello    "
print(str1.rstrip())
```

```
   thello
```

In [ ]:
```
str1="   thello    "
print(str1.strip())
```

```
thello
```

In [ ]:
```
str1="thello    "
print(str1.strip("th"))
```

```
ello
```

In [ ]:
```
str1="thello    "
print(str1.strip("el"))
```

```
thello
```

In [ ]:
```
str1="hello  how are you  "
print(str1.split(" "))
```

```
['hello', '', 'how', 'are', 'you', '', '']
```

In [ ]:
```
str1="hello ,how ,are, you  "
print(str1.split())
```

```
['hello', ',how', ',are,', 'you']
```

In [ ]:
```
#string is pallindrone
str1=input("enter the string")
rev=str1[-1::-1]

if (str1==rev):
    print("pallindrome")
else:
    print("not")
```

```
enter the stringhello
not
```

In [ ]:
```
str1="hello how are you  "
print(str1.split(" ",2))
```

```
['hello', 'how', 'are you  ']
```

In [ ]:
```
str1="hello.txt"
print(str1.split("."))
```

```
['hello', 'txt']
```

In [ ]:
```
str1=".hello.txt."
print(str1.split("."))
```

```
['', 'hello', 'txt', '']
```

In [ ]:
```
str1=".hello.txt."
print(str1.strip("."))
```

```
hello.txt
```

In [ ]:
```
str1="mon  tues wednes"
str2=str1.split()
print (str2)
print (" ".join(str2))
```

```
['mon', 'tues', 'wednes']
mon tues wednes
```

In [ ]:
```
#Remove all vowel characters from a string( university question)
vowels="AEIOUaeiou"
s=input("Enter the string...")
ns=""
for char in s:
    if char not in vowels:
        ns=ns+char
print("new string after removing vowels=",ns)
```

```
Enter the string...hello how are you
new string after removing vowels= hll hw r y
```

In [ ]:
```
#Remove characters at odd index positions from a string ( university q
uestion)
s=input("Enter the string..:")
i=0
ns=""
while i<len(s):
    if i%2==0:
        ns=ns+s[i]
    i=i+1
print("New string:",ns)
```

```
Enter the string..:good morning
New string: go onn
```

In [ ]:
```
#Palindrome checking using loop
s=input("Enter the string..")
beg=0
end=len(s)-1
while beg<end:
   if s[beg]!=s[end]:
         print("Not palindrome")
         break
   beg+=1
   end-=1
else:
  print("Palindrome")
```

```
Enter the string..malayalam
Palindrome
```

In [ ]:
```
#Replace all the spaces in the input string with * or if no spaces fou
nd,put $ at the start and end of the string.(university question)
s=input("Enter the string:")
s=s.replace(" ","*")
if "*" not in s:
   s="$"+s+"$ "
   print(s)
else:
  print(s)
```

```
Enter the string:hello
$hello$
```

In [ ]:
```
#Slice the string to two separate strings; one with all the characters in the odd indices and one with all characters in even indices.(univer sity question)
s=input("enter the string:")
eps=s[0:len(s):2]
print("slice with even position characters:",eps)
ops=s[1:len(s):2]
print("slice with odd position chracters:",ops)
```

```
enter the string:python programming language
slice with even position characters: pto rgamn agae
slice with odd position chracters: yhnpormiglnug
```

In [ ]:
```
#Remove all occurrence of a substring from a string
s=input("enter the string..")
ss=input("enter substring to remove..")
ls=len(s) # length of the string
lss=len(ss) # length of the substring
ns="" # new string
i=0
while i<ls:
  css=s[i:lss+i] #css is the substring to be compared extracted from m ain string
  if css==ss:
    i=i+lss
  else:
    ns=ns+s[i]
    i=i+1
print("new string",ns)
```

```
enter the string..hello how are you
enter substring to remove..are
new string hello how  you
```

In [ ]:

```
#Program to replace all occurrence of a substring with a new substring
(university question)
s=input("enter string..")
ss=input("enter substring to remove..")
nss=input("enter the substring to replace....")
ls=len(s)
lss=len(ss)
ns=""
i=0
while i<ls:
  css=s[i:lss+i]
  if css==ss:
     ns=ns+nss
     i=i+lss
  else:
     ns=ns+s[i]
     i=i+1
print("new string",ns)
```

```
enter string..hello python language
enter substring to remove..hello
enter the substring to replace....hai
new string hai python language
```

In [ ]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving patent.png to patent.png
```

In [ ]:
```
with open("/content/mee.txt","w") as f:
  f.write("hello")
```

In [ ]:
```
#open and read a file
f=open("/content/filenw1.txt","w")
```

In [ ]:
```
f=open("/content/filenw1.txt","w")
```

In [ ]:
```
#We don't need to explicitly call the close() method. It is done inter
nally.
with open("/content/f1.txt","w") as f:
  f.write("hello")
```

In [ ]:
```
#writing data to file
f=open("/content/f2.txt","w")
f.write("this is a new  message written to file")
f.close()
```

In [ ]:
```
#read and write to a same file
f=open("/content/f1.txt","r+")
f.read()
f.write("hello")
f.seek(0)
print(f.read())
f.close()
```

```
hi..welcome to file processinghello
```

In [ ]:
```
#Count the total number of upper case, lower case, and digits used in
 the text file
with open("f1.txt","r") as f1:
    data=f1.read()
    cnt_ucase =0
    cnt_lcase=0
    cnt_digits=0
    for ch in data:
        if ch.islower():
            cnt_lcase+=1
        if ch.isupper():
            cnt_ucase+=1
        if ch.isdigit():
            cnt_digits+=1
    print("Total Number of Upper Case letters are:",cnt_ucase)
    print("Total Number of Lower Case letters are:",cnt_lcase)
```

```
Total Number of Upper Case letters are: 0
Total Number of Lower Case letters are: 5
```

In [ ]:
```
#f.read() will read all content
with open("book.txt","r") as f1:
    data=f1.read()
    print (data)
```

```
reading and writing files, programs can exchange information with each
other and generate printable formats like PDF.
Working with files is a lot like working with books. To use a book, you
have to open it. When you’re done,
you have to close it. While the book is open, you can either write in i
t or read from it. In either case, you know where you are in the book.
Most of the time, you read the whole book in its natural order, but you
can also skip around.
All of this applies to files as well.
```

In [ ]:
```
#f.readlines() will read all lines into a list
with open("/content/book.txt" ,"r") as f1:
 data1=f1.readline(1)
 print(data1)
```

```
---------------------------------------------------------------------------
----
FileNotFoundError                         Traceback (most recent call l
ast)
<ipython-input-2-545561afae8f> in <module>()
      1 #f.readlines() will read all lines into a list
----> 2 with open("/content/book.txt" ,"r") as f1:
      3  data1=f1.readline(1)
      4  print(data1)
      5

FileNotFoundError: [Errno 2] No such file or directory: '/content/book.
txt'
```

In [ ]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving f1.txt to f1 (2).txt
```

In [ ]:
```
#f.readline() will read
#one line from file
with open("f1.txt","r") as f1:
    data=f1.readline()
    print (data)
```

```
hi..welcome to file processing
```

In [ ]:
```
# program to count a total number of lines and
#count the total number of lines starting with ‘A’, ‘B’, and
#‘C’
with open("f2.txt","r") as f1:
    data=f1.readlines()
    cnt_lines=0
    cnt_A=0
    cnt_B=0
    cnt_C=0
    for lines in data:
        cnt_lines+=1
        if lines[0]=='A':
            cnt_A+=1
        if lines[0]=='B':
            cnt_B+=1
        if lines[0]=='C':
            cnt_C+=1
    print("Total Number of lines are:",cnt_lines)
    print("Total Number of lines strating with A are:",cnt_A)
    print("Total Number of lines strating with B are:",cnt_B)
    print("Total Number of lines strating with C are:",cnt_C)
```

```
Total Number of lines are: 3
Total Number of lines strating with A are: 0
Total Number of lines strating with B are: 0
Total Number of lines strating with C are: 0
```

In [ ]:
```python
#word search in python
cnt = 0
word_search = input("Enter the words to search:")
with open("book.txt","r") as f1:
  data=f1.read()
  words = data.split()
  for word in words:
    if (word == word_search):
      cnt+=1
print(word_search, "found ", cnt, " times from the file")
```

```
Enter the words to search:hello
hello found  0  times from the file
```

In [ ]:
```python
#Replace all spaces from text with * (asterisk).
cnt = 0
with open("f2.txt","r") as f1:
    data = f1.read()
    data=data.replace(' ','*')
with open("merge.txt","w") as f1:
    f1.write(data)
```

In [ ]:
```python
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving f1.txt to f1 (2).txt
```

In [ ]:
```python
#read function
f=open('book.txt','r')
c=f.read(4)
print('first 4 characters')
print(c)
c=f.read(4)
print('next 4 characters')
print(c)
print('read the remaining contents')
c=f.read()
print(c)
f.close()
```

```
first 4 characters

next 4 characters

read the remaining contents
```

In [ ]:
```
f=open('book.txt','r')
for l in f:
     print(l,end=' ')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call l
ast)
<ipython-input-3-afb547fa62bf> in <module>()
----> 1 f=open('book.txt','r')
      2 for l in f:
      3      print(l,end=' ')

FileNotFoundError: [Errno 2] No such file or directory: 'book.txt'
```

In [ ]:
```
#Write a program to read the contents of
# both the files created in the above programs and
 #merge the contents into “merge.txt”. Avoid using the close() functio
n to close the files.

with open("book.txt","r") as f1:
    data=f1.read()
with open("/content/f1.txt","r") as f2:
    data1=f2.read()
with open("merge.txt","w") as f3:
    f3.write(data)
    f3.write(data1)
```

In [ ]:
```
f=open('book.txt','r')
for l in f:
     print(l,end='')
```

```
reading and writing files, programs can exchange information with each
other and generate printable formats like PDF.
Working with files is a lot like working with books. To use a book, you
have to open it. When you’re done,
you have to close it. While the book is open, you can either write in i
t or read from it. In either case, you know where you are in the book.
Most of the time, you read the whole book in its natural order, but you
can also skip around.
All of this applies to files as well.
```

In [ ]:
```
#writing numbers to file
f=open("num.txt","w")
for i in range(1,11):
    f.write(str(i)+"\n")
f.close()
```

In [ ]:
```
#To insert 10 numbers to a file and display their sum
f=open("num.txt","w")
for i in range(1,11):
    f.write(str(i)+"\n")
f.close()
f = open("num.txt", 'r')
for line in f:
 number1 = int(line)
 theSum1 += number1
print("The sum is", theSum1)
```

```
1

2

3

4

5

6

7

8

9

10

The sum is 55
```

In [ ]:
```
0#seek and tell mode
f=open('test.txt','w+')
f.write('this is the new content')
f.seek(0)
str=f.read()
print(str)
f.seek(3)
print(f.tell())
str2=f.read()
print(str2)
f.close()
```

```
this is the new content
3
23
s is the new content
```

In [ ]:

```
#binary files
f = open('/content/pexels-pixabay-45201.jpg', 'rb')
print(list(f.read(20000)))
f.close()
```

[255, 216, 255, 224, 0, 16, 74, 70, 73, 70, 0, 1, 1, 1, 0, 72, 0, 72,
0, 0, 255, 226, 12, 88, 73, 67, 67, 95, 80, 82, 79, 70, 73, 76, 69, 0,
1, 1, 0, 0, 12, 72, 76, 105, 110, 111, 2, 16, 0, 0, 109, 110, 116, 114,
82, 71, 66, 32, 88, 89, 90, 32, 7, 206, 0, 2, 0, 9, 0, 6, 0, 49, 0, 0,
97, 99, 115, 112, 77, 83, 70, 84, 0, 0, 0, 0, 73, 69, 67, 32, 115, 82,
71, 66, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 246, 214, 0, 1, 0, 0,
0, 0, 211, 45, 72, 80, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 17, 99, 112, 114, 116, 0, 0, 1, 80, 0, 0,
0, 51, 100, 101, 115, 99, 0, 0, 1, 132, 0, 0, 0, 108, 119, 116, 112, 11
6, 0, 0, 1, 240, 0, 0, 0, 20, 98, 107, 112, 116, 0, 0, 2, 4, 0, 0, 0, 2
0, 114, 88, 89, 90, 0, 0, 2, 24, 0, 0, 0, 20, 103, 88, 89, 90, 0, 0, 2,
44, 0, 0, 0, 20, 98, 88, 89, 90, 0, 0, 2, 64, 0, 0, 0, 20, 100, 109, 11
0, 100, 0, 0, 2, 84, 0, 0, 0, 112, 100, 109, 100, 100, 0, 0, 2, 196, 0,
0, 0, 136, 118, 117, 101, 100, 0, 0, 3, 76, 0, 0, 0, 134, 118, 105, 10
1, 119, 0, 0, 3, 212, 0, 0, 0, 36, 108, 117, 109, 105, 0, 0, 3, 248, 0,
0, 0, 20, 109, 101, 97, 115, 0, 0, 4, 12, 0, 0, 0, 36, 116, 101, 99, 10
4, 0, 0, 4, 48, 0, 0, 0, 12, 114, 84, 82, 67, 0, 0, 4, 60, 0, 0, 8, 12,
103, 84, 82, 67, 0, 0, 4, 60, 0, 0, 8, 12, 98, 84, 82, 67, 0, 0, 4, 60,
0, 0, 8, 12, 116, 101, 120, 116, 0, 0, 0, 0, 67, 111, 112, 121, 114, 10
5, 103, 104, 116, 32, 40, 99, 41, 32, 49, 57, 57, 56, 32, 72, 101, 119,
108, 101, 116, 116, 45, 80, 97, 99, 107, 97, 114, 100, 32, 67, 111, 10
9, 112, 97, 110, 121, 0, 0, 100, 101, 115, 99, 0, 0, 0, 0, 0, 0, 0, 18,
115, 82, 71, 66, 32, 73, 69, 67, 54, 49, 57, 54, 54, 45, 50, 46, 49, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 18, 115, 82, 71, 66, 32, 73, 69, 67, 54,
49, 57, 54, 54, 45, 50, 46, 49, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 88, 89, 90, 32, 0, 0, 0, 0, 0,
0, 243, 81, 0, 1, 0, 0, 0, 1, 22, 204, 88, 89, 90, 32, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 88, 89, 90, 32, 0, 0, 0, 0, 0, 0, 111,
162, 0, 0, 56, 245, 0, 0, 3, 144, 88, 89, 90, 32, 0, 0, 0, 0, 0, 0, 98,
153, 0, 0, 183, 133, 0, 0, 24, 218, 88, 89, 90, 32, 0, 0, 0, 0, 0, 0, 3
6, 160, 0, 0, 15, 132, 0, 0, 182, 207, 100, 101, 115, 99, 0, 0, 0, 0,
0, 0, 0, 22, 73, 69, 67, 32, 104, 116, 116, 112, 58, 47, 47, 119, 119,
119, 46, 105, 101, 99, 46, 99, 104, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2
2, 73, 69, 67, 32, 104, 116, 116, 112, 58, 47, 47, 119, 119, 119, 46, 1
05, 101, 99, 46, 99, 104, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 100, 101, 115, 99, 0, 0, 0, 0, 0, 0, 0, 46, 73, 6
9, 67, 32, 54, 49, 57, 54, 54, 45, 50, 46, 49, 32, 68, 101, 102, 97, 11
7, 108, 116, 32, 82, 71, 66, 32, 99, 111, 108, 111, 117, 114, 32, 115,
112, 97, 99, 101, 32, 45, 32, 115, 82, 71, 66, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 46, 73, 69, 67, 32, 54, 49, 57, 54, 54, 45, 50, 46, 49, 32, 6
8, 101, 102, 97, 117, 108, 116, 32, 82, 71, 66, 32, 99, 111, 108, 111,
117, 114, 32, 115, 112, 97, 99, 101, 32, 45, 32, 115, 82, 71, 66, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100, 101, 1
15, 99, 0, 0, 0, 0, 0, 0, 0, 0, 44, 82, 101, 102, 101, 114, 101, 110, 99,
101, 32, 86, 105, 101, 119, 105, 110, 103, 32, 67, 111, 110, 100, 105,
116, 105, 111, 110, 32, 105, 110, 32, 73, 69, 67, 54, 49, 57, 54, 54, 4
5, 50, 46, 49, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 44, 82, 101, 102, 101,
114, 101, 110, 99, 101, 32, 86, 105, 101, 119, 105, 110, 103, 32, 67, 1
11, 110, 100, 105, 116, 105, 111, 110, 32, 105, 110, 32, 73, 69, 67, 5
4, 49, 57, 54, 54, 45, 50, 46, 49, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 118, 105, 101, 119, 0, 0, 0,
0, 0, 19, 164, 254, 0, 20, 95, 46, 0, 16, 207, 20, 0, 3, 237, 204, 0,
4, 19, 11, 0, 3, 92, 158, 0, 0, 0, 1, 88, 89, 90, 32, 0, 0, 0, 0, 0, 7
6, 9, 86, 0, 80, 0, 0, 0, 87, 31, 231, 109, 101, 97, 115, 0, 0, 0, 0,

0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 14
3, 0, 0, 0, 2, 115, 105, 103, 32, 0, 0, 0, 0, 67, 82, 84, 32, 99, 117,
114, 118, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 5, 0, 10, 0, 15, 0, 20, 0, 2
5, 0, 30, 0, 35, 0, 40, 0, 45, 0, 50, 0, 55, 0, 59, 0, 64, 0, 69, 0, 7
4, 0, 79, 0, 84, 0, 89, 0, 94, 0, 99, 0, 104, 0, 109, 0, 114, 0, 119,
0, 124, 0, 129, 0, 134, 0, 139, 0, 144, 0, 149, 0, 154, 0, 159, 0, 164,
0, 169, 0, 174, 0, 178, 0, 183, 0, 188, 0, 193, 0, 198, 0, 203, 0, 208,
0, 213, 0, 219, 0, 224, 0, 229, 0, 235, 0, 240, 0, 246, 0, 251, 1, 1,
1, 7, 1, 13, 1, 19, 1, 25, 1, 31, 1, 37, 1, 43, 1, 50, 1, 56, 1, 62, 1,
69, 1, 76, 1, 82, 1, 89, 1, 96, 1, 103, 1, 110, 1, 117, 1, 124, 1, 131,
1, 139, 1, 146, 1, 154, 1, 161, 1, 169, 1, 177, 1, 185, 1, 193, 1, 201,
1, 209, 1, 217, 1, 225, 1, 233, 1, 242, 1, 250, 2, 3, 2, 12, 2, 20, 2,
29, 2, 38, 2, 47, 2, 56, 2, 65, 2, 75, 2, 84, 2, 93, 2, 103, 2, 113, 2,
122, 2, 132, 2, 142, 2, 152, 2, 162, 2, 172, 2, 182, 2, 193, 2, 203, 2,
213, 2, 224, 2, 235, 2, 245, 3, 0, 3, 11, 3, 22, 3, 33, 3, 45, 3, 56,
3, 67, 3, 79, 3, 90, 3, 102, 3, 114, 3, 126, 3, 138, 3, 150, 3, 162, 3,
174, 3, 186, 3, 199, 3, 211, 3, 224, 3, 236, 3, 249, 4, 6, 4, 19, 4, 3
2, 4, 45, 4, 59, 4, 72, 4, 85, 4, 99, 4, 113, 4, 126, 4, 140, 4, 154,
4, 168, 4, 182, 4, 196, 4, 211, 4, 225, 4, 240, 4, 254, 5, 13, 5, 28,
5, 43, 5, 58, 5, 73, 5, 88, 5, 103, 5, 119, 5, 134, 5, 150, 5, 166, 5,
181, 5, 197, 5, 213, 5, 229, 5, 246, 6, 6, 6, 22, 6, 39, 6, 55, 6, 72,
6, 89, 6, 106, 6, 123, 6, 140, 6, 157, 6, 175, 6, 192, 6, 209, 6, 227,
6, 245, 7, 7, 7, 25, 7, 43, 7, 61, 7, 79, 7, 97, 7, 116, 7, 134, 7, 15
3, 7, 172, 7, 191, 7, 210, 7, 229, 7, 248, 8, 11, 8, 31, 8, 50, 8, 70,
8, 90, 8, 110, 8, 130, 8, 150, 8, 170, 8, 190, 8, 210, 8, 231, 8, 251,
9, 16, 9, 37, 9, 58, 9, 79, 9, 100, 9, 121, 9, 143, 9, 164, 9, 186, 9,
207, 9, 229, 9, 251, 10, 17, 10, 39, 10, 61, 10, 84, 10, 106, 10, 129,
10, 152, 10, 174, 10, 197, 10, 220, 10, 243, 11, 11, 11, 34, 11, 57, 1
1, 81, 11, 105, 11, 128, 11, 152, 11, 176, 11, 200, 11, 225, 11, 249, 1
2, 18, 12, 42, 12, 67, 12, 92, 12, 117, 12, 142, 12, 167, 12, 192, 12,
217, 12, 243, 13, 13, 13, 38, 13, 64, 13, 90, 13, 116, 13, 142, 13, 16
9, 13, 195, 13, 222, 13, 248, 14, 19, 14, 46, 14, 73, 14, 100, 14, 127,
14, 155, 14, 182, 14, 210, 14, 238, 15, 9, 15, 37, 15, 65, 15, 94, 15,
122, 15, 150, 15, 179, 15, 207, 15, 236, 16, 9, 16, 38, 16, 67, 16, 97,
16, 126, 16, 155, 16, 185, 16, 215, 16, 245, 17, 19, 17, 49, 17, 79, 1
7, 109, 17, 140, 17, 170, 17, 201, 17, 232, 18, 7, 18, 38, 18, 69, 18,
100, 18, 132, 18, 163, 18, 195, 18, 227, 19, 3, 19, 35, 19, 67, 19, 99,
19, 131, 19, 164, 19, 197, 19, 229, 20, 6, 20, 39, 20, 73, 20, 106, 20,
139, 20, 173, 20, 206, 20, 240, 21, 18, 21, 52, 21, 86, 21, 120, 21, 15
5, 21, 189, 21, 224, 22, 3, 22, 38, 22, 73, 22, 108, 22, 143, 22, 178,
22, 214, 22, 250, 23, 29, 23, 65, 23, 101, 23, 137, 23, 174, 23, 210, 2
3, 247, 24, 27, 24, 64, 24, 101, 24, 138, 24, 175, 24, 213, 24, 250, 2
5, 32, 25, 69, 25, 107, 25, 145, 25, 183, 25, 221, 26, 4, 26, 42, 26, 8
1, 26, 119, 26, 158, 26, 197, 26, 236, 27, 20, 27, 59, 27, 99, 27, 138,
27, 178, 27, 218, 28, 2, 28, 42, 28, 82, 28, 123, 28, 163, 28, 204, 28,
245, 29, 30, 29, 71, 29, 112, 29, 153, 29, 195, 29, 236, 30, 22, 30, 6
4, 30, 106, 30, 148, 30, 190, 30, 233, 31, 19, 31, 62, 31, 105, 31, 14
8, 31, 191, 31, 234, 32, 21, 32, 65, 32, 108, 32, 152, 32, 196, 32, 24
0, 33, 28, 33, 72, 33, 117, 33, 161, 33, 206, 33, 251, 34, 39, 34, 85,
34, 130, 34, 175, 34, 221, 35, 10, 35, 56, 35, 102, 35, 148, 35, 194, 3
5, 240, 36, 31, 36, 77, 36, 124, 36, 171, 36, 218, 37, 9, 37, 56, 37, 1
04, 37, 151, 37, 199, 37, 247, 38, 39, 38, 87, 38, 135, 38, 183, 38, 23
2, 39, 24, 39, 73, 39, 122, 39, 171, 39, 220, 40, 13, 40, 63, 40, 113,
40, 162, 40, 212, 41, 6, 41, 56, 41, 107, 41, 157, 41, 208, 42, 2, 42,
53, 42, 104, 42, 155, 42, 207, 43, 2, 43, 54, 43, 105, 43, 157, 43, 20
9, 44, 5, 44, 57, 44, 110, 44, 162, 44, 215, 45, 12, 45, 65, 45, 118, 4
5, 171, 45, 225, 46, 22, 46, 76, 46, 130, 46, 183, 46, 238, 47, 36, 47,

90, 47, 145, 47, 199, 47, 254, 48, 53, 48, 108, 48, 164, 48, 219, 49, 1
8, 49, 74, 49, 130, 49, 186, 49, 242, 50, 42, 50, 99, 50, 155, 50, 212,
51, 13, 51, 70, 51, 127, 51, 184, 51, 241, 52, 43, 52, 101, 52, 158, 5
2, 216, 53, 19, 53, 77, 53, 135, 53, 194, 53, 253, 54, 55, 54, 114, 54,
174, 54, 233, 55, 36, 55, 96, 55, 156, 55, 215, 56, 20, 56, 80, 56, 14
0, 56, 200, 57, 5, 57, 66, 57, 127, 57, 188, 57, 249, 58, 54, 58, 116,
58, 178, 58, 239, 59, 45, 59, 107, 59, 170, 59, 232, 60, 39, 60, 101, 6
0, 164, 60, 227, 61, 34, 61, 97, 61, 161, 61, 224, 62, 32, 62, 96, 62,
160, 62, 224, 63, 33, 63, 97, 63, 162, 63, 226, 64, 35, 64, 100, 64, 16
6, 64, 231, 65, 41, 65, 106, 65, 172, 65, 238, 66, 48, 66, 114, 66, 18
1, 66, 247, 67, 58, 67, 125, 67, 192, 68, 3, 68, 71, 68, 138, 68, 206,
69, 18, 69, 85, 69, 154, 69, 222, 70, 34, 70, 103, 70, 171, 70, 240, 7
1, 53, 71, 123, 71, 192, 72, 5, 72, 75, 72, 145, 72, 215, 73, 29, 73, 9
9, 73, 169, 73, 240, 74, 55, 74, 125, 74, 196, 75, 12, 75, 83, 75, 154,
75, 226, 76, 42, 76, 114, 76, 186, 77, 2, 77, 74, 77, 147, 77, 220, 78,
37, 78, 110, 78, 183, 79, 0, 79, 73, 79, 147, 79, 221, 80, 39, 80, 113,
80, 187, 81, 6, 81, 80, 81, 155, 81, 230, 82, 49, 82, 124, 82, 199, 83,
19, 83, 95, 83, 170, 83, 246, 84, 66, 84, 143, 84, 219, 85, 40, 85, 11
7, 85, 194, 86, 15, 86, 92, 86, 169, 86, 247, 87, 68, 87, 146, 87, 224,
88, 47, 88, 125, 88, 203, 89, 26, 89, 105, 89, 184, 90, 7, 90, 86, 90,
166, 90, 245, 91, 69, 91, 149, 91, 229, 92, 53, 92, 134, 92, 214, 93, 3
9, 93, 120, 93, 201, 94, 26, 94, 108, 94, 189, 95, 15, 95, 97, 95, 179,
96, 5, 96, 87, 96, 170, 96, 252, 97, 79, 97, 162, 97, 245, 98, 73, 98,
156, 98, 240, 99, 67, 99, 151, 99, 235, 100, 64, 100, 148, 100, 233, 10
1, 61, 101, 146, 101, 231, 102, 61, 102, 146, 102, 232, 103, 61, 103, 1
47, 103, 233, 104, 63, 104, 150, 104, 236, 105, 67, 105, 154, 105, 241,
106, 72, 106, 159, 106, 247, 107, 79, 107, 167, 107, 255, 108, 87, 108,
175, 109, 8, 109, 96, 109, 185, 110, 18, 110, 107, 110, 196, 111, 30, 1
11, 120, 111, 209, 112, 43, 112, 134, 112, 224, 113, 58, 113, 149, 113,
240, 114, 75, 114, 166, 115, 1, 115, 93, 115, 184, 116, 20, 116, 112, 1
16, 204, 117, 40, 117, 133, 117, 225, 118, 62, 118, 155, 118, 248, 119,
86, 119, 179, 120, 17, 120, 110, 120, 204, 121, 42, 121, 137, 121, 231,
122, 70, 122, 165, 123, 4, 123, 99, 123, 194, 124, 33, 124, 129, 124, 2
25, 125, 65, 125, 161, 126, 1, 126, 98, 126, 194, 127, 35, 127, 132, 12
7, 229, 128, 71, 128, 168, 129, 10, 129, 107, 129, 205, 130, 48, 130, 1
46, 130, 244, 131, 87, 131, 186, 132, 29, 132, 128, 132, 227, 133, 71,
133, 171, 134, 14, 134, 114, 134, 215, 135, 59, 135, 159, 136, 4, 136,
105, 136, 206, 137, 51, 137, 153, 137, 254, 138, 100, 138, 202, 139, 4
8, 139, 150, 139, 252, 140, 99, 140, 202, 141, 49, 141, 152, 141, 255,
142, 102, 142, 206, 143, 54, 143, 158, 144, 6, 144, 110, 144, 214, 145,
63, 145, 168, 146, 17, 146, 122, 146, 227, 147, 77, 147, 182, 148, 32,
148, 138, 148, 244, 149, 95, 149, 201, 150, 52, 150, 159, 151, 10, 151,
117, 151, 224, 152, 76, 152, 184, 153, 36, 153, 144, 153, 252, 154, 10
4, 154, 213, 155, 66, 155, 175, 156, 28, 156, 137, 156, 247, 157, 100,
157, 210, 158, 64, 158, 174, 159, 29, 159, 139, 159, 250, 160, 105, 16
0, 216, 161, 71, 161, 182, 162, 38, 162, 150, 163, 6, 163, 118, 163, 23
0, 164, 86, 164, 199, 165, 56, 165, 169, 166, 26, 166, 139, 166, 253, 1
67, 110, 167, 224, 168, 82, 168, 196, 169, 55, 169, 169, 170, 28, 170,
143, 171, 2, 171, 117, 171, 233, 172, 92, 172, 208, 173, 68, 173, 184,
174, 45, 174, 161, 175, 22, 175, 139, 176, 0, 176, 117, 176, 234, 177,
96, 177, 214, 178, 75, 178, 194, 179, 56, 179, 174, 180, 37, 180, 156,
181, 19, 181, 138, 182, 1, 182, 121, 182, 240, 183, 104, 183, 224, 184,
89, 184, 209, 185, 74, 185, 194, 186, 59, 186, 181, 187, 46, 187, 167,
188, 33, 188, 155, 189, 21, 189, 143, 190, 10, 190, 132, 190, 255, 191,
122, 191, 245, 192, 112, 192, 236, 193, 103, 193, 227, 194, 95, 194, 21
9, 195, 88, 195, 212, 196, 81, 196, 206, 197, 75, 197, 200, 198, 70, 19
8, 195, 199, 65, 199, 191, 200, 61, 200, 188, 201, 58, 201, 185, 202, 5

6, 202, 183, 203, 54, 203, 182, 204, 53, 204, 181, 205, 53, 205, 181, 2
06, 54, 206, 182, 207, 55, 207, 184, 208, 57, 208, 186, 209, 60, 209, 1
90, 210, 63, 210, 193, 211, 68, 211, 198, 212, 73, 212, 203, 213, 78, 2
13, 209, 214, 85, 214, 216, 215, 92, 215, 224, 216, 100, 216, 232, 217,
108, 217, 241, 218, 118, 218, 251, 219, 128, 220, 5, 220, 138, 221, 16,
221, 150, 222, 28, 222, 162, 223, 41, 223, 175, 224, 54, 224, 189, 225,
68, 225, 204, 226, 83, 226, 219, 227, 99, 227, 235, 228, 115, 228, 252,
229, 132, 230, 13, 230, 150, 231, 31, 231, 169, 232, 50, 232, 188, 233,
70, 233, 208, 234, 91, 234, 229, 235, 112, 235, 251, 236, 134, 237, 17,
237, 156, 238, 40, 238, 180, 239, 64, 239, 204, 240, 88, 240, 229, 241,
114, 241, 255, 242, 140, 243, 25, 243, 167, 244, 52, 244, 194, 245, 80,
245, 222, 246, 109, 246, 251, 247, 138, 248, 25, 248, 168, 249, 56, 24
9, 199, 250, 87, 250, 231, 251, 119, 252, 7, 252, 152, 253, 41, 253, 18
6, 254, 75, 254, 220, 255, 109, 255, 255, 255, 219, 0, 132, 0, 2, 3, 3,
3, 4, 3, 4, 5, 5, 4, 6, 6, 6, 6, 6, 8, 8, 7, 7, 8, 8, 13, 9, 10, 9, 10,
9, 13, 19, 12, 14, 12, 12, 14, 12, 19, 17, 20, 17, 15, 17, 20, 17, 30,
24, 21, 21, 24, 30, 35, 29, 28, 29, 35, 42, 37, 37, 42, 53, 50, 53, 69,
69, 92, 1, 2, 3, 3, 3, 4, 3, 4, 5, 5, 4, 6, 6, 6, 6, 6, 8, 8, 7, 7, 8,
8, 13, 9, 10, 9, 10, 9, 13, 19, 12, 14, 12, 12, 14, 12, 19, 17, 20, 17,
15, 17, 20, 17, 30, 24, 21, 21, 24, 30, 35, 29, 28, 29, 35, 42, 37, 37,
42, 53, 50, 53, 69, 69, 92, 255, 194, 0, 17, 8, 9, 196, 9, 88, 3, 1, 3
4, 0, 2, 17, 1, 3, 17, 1, 255, 196, 0, 30, 0, 0, 1, 5, 1, 1, 1, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 4, 5, 6, 3, 7, 8, 9, 10, 255, 218, 0,
8, 1, 1, 0, 0, 0, 0, 254, 127, 220, 170, 138, 0, 0, 0, 0, 168, 3, 133,
26, 142, 68, 28, 173, 112, 162, 136, 0, 15, 80, 104, 225, 200, 138, 12
8, 130, 128, 15, 20, 21, 85, 17, 17, 80, 81, 206, 122, 53, 17, 80, 106,
185, 81, 5, 69, 68, 21, 84, 17, 21, 16, 16, 65, 81, 206, 120, 52, 28, 1
0, 168, 160, 196, 81, 69, 16, 68, 64, 80, 21, 168, 136, 8, 136, 162, 16
8, 8, 162, 2, 181, 173, 104, 0, 40, 130, 160, 136, 32, 142, 65, 21, 20
2, 130, 128, 141, 86, 171, 208, 5, 7, 10, 160, 196, 98, 15, 80, 5, 87,
8, 212, 1, 85, 17, 206, 0, 98, 2, 160, 136, 228, 20, 16, 82, 59, 149, 6
4, 0, 0, 0, 0, 0, 112, 168, 208, 81, 28, 168, 168, 56, 0, 5, 78, 136, 5
3, 30, 163, 154, 42, 0, 130, 138, 10, 170, 138, 229, 68, 26, 160, 42, 1
84, 122, 52, 68, 81, 4, 114, 162, 10, 168, 136, 170, 10, 2, 34, 32, 13
8, 34, 189, 85, 68, 71, 42, 40, 160, 57, 26, 34, 57, 5, 6, 0, 10, 130,
163, 68, 68, 20, 68, 85, 106, 170, 40, 42, 35, 90, 212, 80, 64, 0, 4, 8
4, 16, 65, 20, 5, 84, 112, 32, 32, 160, 142, 65, 65, 202, 136, 136, 21
3, 111, 64, 0, 115, 128, 70, 163, 133, 71, 0, 213, 26, 128, 0, 138, 21
3, 80, 17, 72, 238, 112, 0, 42, 42, 0, 0, 0, 2, 168, 208, 21, 90, 229,
69, 17, 84, 0, 1, 69, 68, 115, 133, 64, 0, 0, 28, 40, 162, 168, 136, 13
0, 128, 229, 81, 195, 68, 69, 81, 1, 90, 10, 161, 204, 87, 40, 42, 13,
64, 64, 28, 225, 202, 52, 87, 32, 160, 163, 129, 136, 130, 162, 170, 3
4, 42, 180, 4, 84, 4, 26, 138, 130, 40, 0, 2, 163, 70, 136, 0, 172, 64,
1, 170, 2, 181, 68, 104, 56, 112, 0, 8, 209, 202, 128, 10, 162, 2, 53,
85, 92, 130, 138, 131, 128, 0, 1, 81, 24, 162, 2, 40, 32, 213, 114, 40,
34, 175, 5, 80, 1, 68, 85, 65, 0, 0, 1, 69, 68, 1, 85, 162, 168, 160, 1
62, 160, 0, 43, 216, 43, 128, 0, 21, 1, 85, 85, 20, 20, 86, 128, 174, 1
06, 142, 5, 16, 17, 192, 128, 34, 10, 10, 196, 120, 162, 170, 32, 34, 1
76, 28, 160, 174, 4, 28, 0, 228, 112, 168, 193, 5, 65, 80, 65, 68, 16,
64, 1, 20, 107, 81, 64, 84, 21, 28, 212, 70, 136, 213, 85, 107, 64, 0,
69, 0, 65, 1, 163, 156, 2, 0, 212, 81, 84, 84, 20, 81, 4, 107, 149, 23
4, 136, 225, 26, 143, 85, 16, 0, 5, 107, 5, 64, 17, 64, 65, 68, 5, 115,
163, 40, 0, 10, 10, 32, 128, 2, 136, 57, 90, 128, 40, 128, 224, 114, 1
0, 0, 0, 10, 160, 224, 0, 28, 128, 57, 64, 5, 115, 81, 20, 114, 168, 8,
160, 0, 43, 145, 16, 4, 5, 1, 20, 1, 200, 32, 2, 42, 40, 42, 185, 68,
0, 85, 20, 17, 17, 80, 65, 68, 1, 80, 106, 0, 0, 2, 34, 57, 1, 16, 85,
68, 70, 167, 49, 85, 195, 16, 0, 0, 0, 16, 64, 28, 8, 2, 177, 7, 14, 1

7, 21, 64, 1, 174, 123, 154, 212, 122, 140, 71, 56, 0, 68, 85, 4, 96,
2, 49, 202, 40, 162, 13, 81, 94, 232, 192, 0, 42, 14, 106, 160, 2, 130,
160, 170, 131, 64, 84, 21, 69, 69, 84, 80, 0, 0, 21, 202, 2, 160, 10, 1
0, 174, 114, 34, 10, 229, 24, 128, 174, 80, 0, 4, 20, 85, 65, 0, 84, 1,
87, 163, 26, 131, 144, 16, 0, 69, 20, 28, 245, 84, 106, 10, 170, 168, 1
73, 17, 21, 0, 17, 81, 68, 17, 17, 80, 0, 21, 4, 84, 84, 84, 106, 131,
65, 173, 230, 14, 84, 104, 0, 160, 128, 0, 8, 8, 228, 84, 1, 162, 0, 22
9, 69, 80, 81, 0, 94, 138, 214, 15, 14, 96, 175, 1, 24, 14, 112, 115,
1, 168, 142, 85, 115, 196, 98, 53, 71, 61, 241, 1, 20, 0, 21, 0, 21, 6
4, 4, 5, 84, 68, 0, 28, 168, 162, 130, 40, 0, 0, 174, 80, 7, 34, 10, 3
5, 213, 234, 212, 65, 94, 168, 141, 71, 42, 128, 0, 2, 40, 162, 180, 6
9, 0, 85, 31, 38, 43, 81, 200, 130, 160, 128, 0, 170, 143, 30, 162, 34,
168, 0, 136, 136, 162, 0, 8, 0, 136, 130, 171, 64, 1, 80, 0, 68, 69, 1,
168, 198, 160, 170, 193, 68, 5, 4, 0, 0, 65, 81, 65, 21, 4, 68, 1, 94,
130, 128, 168, 3, 220, 230, 32, 160, 209, 170, 57, 65, 173, 7, 57, 26,
128, 212, 21, 87, 167, 65, 140, 104, 3, 221, 24, 4, 80, 0, 0, 28, 160,
173, 84, 69, 69, 65, 20, 64, 85, 80, 81, 170, 160, 2, 136, 15, 80, 5, 1
6, 114, 14, 115, 145, 4, 30, 170, 131, 17, 206, 0, 0, 0, 7, 53, 80, 5,
81, 21, 195, 245, 217, 62, 72, 34, 128, 128, 34, 10, 56, 21, 234, 2, 4
0, 0, 52, 65, 20, 64, 1, 21, 16, 104, 10, 32, 10, 32, 138, 8, 141, 114,
136, 55, 152, 128, 163, 21, 220, 197, 64, 0, 0, 17, 81, 64, 4, 17, 16,
112, 224, 69, 0, 1, 94, 162, 34, 128, 53, 162, 188, 1, 138, 136, 244, 1
04, 35, 5, 114, 143, 85, 107, 80, 1, 235, 24, 17, 84, 0, 1, 84, 85, 65,
20, 65, 65, 170, 138, 136, 10, 229, 4, 84, 20, 0, 0, 28, 224, 5, 1, 23
7, 5, 85, 16, 87, 185, 90, 214, 162, 171, 132, 80, 5, 64, 0, 17, 69, 11
4, 185, 173, 127, 91, 159, 126, 249, 254, 145, 168, 32, 162, 40, 8, 2,
128, 175, 81, 69, 86, 136, 8, 136, 138, 2, 0, 8, 162, 32, 8, 162, 0, 3
2, 0, 136, 138, 228, 83, 155, 90, 34, 141, 1, 160, 0, 42, 0, 128, 168,
160, 2, 32, 32, 229, 28, 136, 0, 0, 61, 64, 0, 6, 160, 240, 0, 98, 61,
90, 209, 26, 170, 160, 160, 128, 0, 174, 140, 2, 168, 0, 3, 156, 0, 12
8, 35, 144, 68, 17, 192, 130, 168, 160, 0, 0, 162, 0, 175, 0, 1, 85, 2
1, 81, 192, 57, 194, 136, 115, 85, 17, 84, 80, 0, 84, 1, 5, 21, 87, 17
1, 137, 62, 229, 214, 215, 143, 206, 145, 154, 0, 2, 136, 42, 40, 0, 17
5, 114, 138, 140, 4, 21, 26, 138, 138, 130, 136, 8, 160, 8, 168, 173, 8
4, 81, 170, 208, 4, 5, 1, 27, 205, 0, 6, 170, 52, 1, 68, 21, 1, 0, 80,
0, 6, 170, 128, 116, 70, 2, 40, 0, 160, 229, 0, 1, 17, 192, 0, 212, 12
0, 115, 65, 21, 65, 5, 0, 0, 85, 224, 35, 128, 0, 1, 234, 2, 43, 84, 2
0, 16, 106, 170, 10, 160, 0, 0, 34, 142, 85, 230, 14, 112, 0, 0, 40, 17
0, 163, 144, 114, 128, 192, 65, 206, 84, 69, 69, 5, 16, 16, 81, 71, 62,
204, 245, 63, 169, 179, 26, 31, 29, 249, 170, 11, 80, 69, 1, 80, 21, 0,
17, 92, 231, 40, 173, 104, 32, 163, 80, 71, 34, 0, 43, 84, 80, 16, 16,
69, 4, 68, 84, 0, 84, 4, 107, 26, 0, 130, 181, 16, 28, 225, 173, 21, 4,
5, 0, 17, 65, 5, 0, 114, 177, 16, 112, 42, 0, 42, 184, 5, 16, 4, 80, 1,
20, 57, 136, 40, 0, 34, 56, 0, 14, 35, 129, 68, 84, 21, 21, 224, 2, 10,
34, 128, 34, 160, 160, 0, 0, 0, 14, 84, 106, 185, 64, 0, 0, 85, 81, 81,
65, 202, 8, 136, 160, 229, 1, 20, 20, 4, 26, 174, 17, 93, 215, 77, 195,
218, 190, 188, 246, 39, 124, 115, 226, 126, 20, 209, 90, 254, 104, 8,
2, 168, 128, 174, 58, 32, 11, 204, 17, 68, 70, 142, 104, 0, 0, 2, 130,
2, 2, 8, 138, 136, 40, 168, 131, 90, 52, 64, 64, 68, 65, 94, 168, 32, 1
36, 130, 10, 13, 85, 107, 128, 20, 64, 81, 17, 20, 81, 80, 0, 87, 40, 1
60, 128, 128, 40, 162, 13, 104, 0, 0, 0, 0, 1, 205, 20, 21, 90, 2, 168,
170, 2, 10, 34, 128, 0, 138, 138, 2, 160, 168, 0, 29, 17, 81, 90, 224,
0, 0, 1, 84, 81, 80, 87, 43, 80, 5, 114, 170, 0, 3, 128, 69, 104, 168,
227, 164, 175, 82, 200, 253, 51, 246, 77, 255, 0, 168, 248, 159, 231, 2
23, 146, 100, 153, 117, 74, 238, 44, 114, 32, 10, 246, 160, 229, 20, 12
0, 8, 192, 1, 17, 160, 162, 0, 0, 168, 42, 40, 128, 3, 90, 0, 0, 173, 7
0, 163, 92, 131, 65, 130, 131, 94, 174, 68, 0, 98, 0, 8, 2, 128, 43, 15

2, 116, 107, 64, 69, 58, 115, 233, 204, 0, 114, 170, 136, 2, 34, 42, 13
8, 13, 104, 138, 0, 0, 32, 2, 128, 115, 0, 0, 21, 192, 160, 8, 160, 0,
0, 0, 0, 57, 21, 160, 2, 188, 4, 80, 0, 0, 69, 5, 84, 80, 7, 10, 34, 4
2, 162, 185, 64, 1, 84, 64, 114, 2, 142, 237, 111, 244, 135, 146, 125,
175, 238, 126, 183, 113, 164, 248, 115, 202, 62, 93, 243, 15, 98, 242,
14, 144, 90, 10, 128, 229, 65, 84, 20, 28, 162, 48, 1, 21, 163, 85, 20
5, 64, 21, 1, 81, 205, 20, 64, 16, 68, 69, 64, 84, 4, 106, 32, 8, 35, 9
0, 138, 228, 81, 84, 17, 205, 26, 208, 0, 65, 64, 1, 205, 114, 141, 64,
1, 202, 34, 8, 42, 57, 202, 32, 3, 90, 168, 42, 162, 34, 128, 0, 0, 13
6, 168, 162, 129, 204, 0, 0, 28, 228, 80, 16, 80, 0, 7, 52, 0, 1, 234,
12, 64, 81, 202, 34, 128, 0, 0, 34, 128, 2, 168, 225, 202, 53, 21, 200,
138, 224, 16, 85, 114, 8, 174, 65, 84, 235, 99, 119, 246, 207, 148, 25
4, 141, 125, 7, 177, 177, 216, 120, 111, 147, 124, 137, 249, 199, 245,
215, 201, 144, 106, 121, 162, 162, 160, 225, 85, 20, 3, 162, 40, 140,
0, 4, 104, 224, 106, 0, 2, 128, 32, 162, 2, 13, 0, 16, 4, 106, 0, 213,
26, 136, 131, 145, 28, 42, 130, 32, 35, 20, 16, 21, 80, 64, 0, 115, 14
5, 128, 14, 114, 40, 214, 128, 175, 69, 0, 17, 168, 32, 160, 136, 170,
0, 0, 138, 136, 230, 168, 162, 167, 48, 20, 64, 7, 42, 130, 8, 162, 12
8, 42, 160, 128, 0, 15, 80, 98, 10, 61, 170, 10, 0, 0, 0, 0, 0, 170, 17
0, 174, 80, 16, 20, 69, 6, 142, 20, 21, 71, 57, 174, 233, 35, 209, 57,
254, 187, 236, 181, 255, 0, 64, 242, 187, 225, 42, 94, 83, 240, 219, 23
5, 111, 143, 126, 88, 170, 136, 196, 84, 69, 21, 69, 0, 81, 92, 35, 0,
0, 69, 69, 17, 168, 0, 228, 17, 68, 5, 65, 16, 64, 21, 162, 160, 136, 2
09, 21, 170, 212, 4, 71, 53, 84, 81, 81, 0, 17, 26, 2, 188, 14, 96, 8,
116, 84, 98, 184, 69, 80, 65, 26, 170, 56, 0, 68, 68, 4, 80, 17, 17, 8
1, 202, 8, 40, 8, 32, 170, 116, 111, 16, 85, 86, 160, 14, 114, 40, 136,
162, 56, 1, 92, 53, 0, 4, 81, 94, 8, 193, 200, 138, 174, 0, 0, 1, 80,
1, 68, 81, 194, 138, 229, 0, 0, 0, 65, 69, 17, 92, 178, 39, 214, 44, 15
7, 47, 210, 190, 219, 250, 169, 215, 202, 189, 19, 47, 177, 149, 113, 1
35, 246, 11, 15, 131, 254, 88, 248, 187, 5, 142, 228, 199, 52, 81, 28,
2, 128, 142, 112, 141, 64, 0, 65, 21, 90, 173, 16, 21, 81, 21, 0, 84,
4, 98, 138, 2, 160, 209, 24, 52, 65, 0, 68, 81, 21, 85, 80, 16, 0, 230,
7, 64, 14, 96, 0, 189, 17, 5, 1, 21, 1, 26, 57, 69, 6, 162, 187, 152,
0, 2, 34, 42, 138, 32, 10, 2, 8, 57, 94, 112, 69, 80, 26, 3, 149, 64,
4, 81, 20, 87, 40, 196, 0, 68, 85, 21, 202, 141, 80, 69, 112, 168, 42,
0, 10, 40, 130, 138, 32, 170, 168, 40, 174, 0, 0, 21, 1, 81, 85, 28, 18
9, 37, 243, 75, 143, 84, 247, 95, 213, 31, 75, 243, 11, 205, 53, 158, 1
26, 39, 167, 121, 181, 175, 174, 236, 126, 54, 248, 235, 231, 159, 134,
185, 198, 0, 0, 14, 136, 141, 21, 65, 160, 138, 0, 209, 85, 21, 24, 10,
130, 162, 136, 0, 32, 212, 112, 52, 85, 17, 17, 136, 13, 69, 64, 68, 11
2, 2, 162, 136, 0, 115, 14, 141, 28, 12, 20, 84, 106, 142, 81, 5, 26, 1
43, 17, 81, 17, 92, 0, 209, 4, 1, 80, 84, 26, 138, 170, 130, 160, 40,
2, 42, 42, 188, 224, 40, 34, 0, 57, 84, 81, 81, 90, 0, 116, 68, 104, 0,
32, 40, 174, 16, 17, 21, 202, 0, 42, 0, 162, 170, 10, 168, 2, 128, 10,
138, 229, 0, 7, 13, 5, 21, 85, 83, 170, 200, 178, 245, 15, 170, 126, 21
9, 244, 173, 222, 10, 215, 79, 42, 142, 7, 165, 249, 154, 123, 61, 206,
187, 229, 207, 13, 252, 72, 244, 47, 13, 26, 128, 0, 42, 162, 0, 168,
0, 0, 0, 0, 214, 128, 0, 0, 128, 52, 4, 84, 85, 81, 168, 49, 17, 81, 16
2, 0, 0, 2, 130, 0, 12, 64, 5, 81, 170, 143, 84, 1, 7, 32, 163, 85, 64,
1, 81, 80, 6, 168, 215, 176, 64, 1, 17, 84, 1, 20, 28, 208, 17, 69, 12
1, 29, 65, 0, 1, 92, 160, 116, 26, 208, 1, 85, 189, 24, 128, 8, 10, 10,
8, 0, 231, 0, 0, 3, 148, 65, 84, 64, 20, 0, 20, 28, 160, 2, 168, 32, 3
5, 157, 208, 153, 198, 95, 211, 190, 143, 247, 206, 186, 239, 208, 50,
209, 182, 83, 106, 51, 62, 193, 230, 21, 126, 207, 7, 210, 116, 158, 9
1, 248, 71, 35, 226, 89, 117, 206, 230, 208, 0, 0, 0, 84, 0, 0, 0, 4, 6
4, 1, 160, 35, 84, 4, 20, 69, 81, 4, 115, 81, 17, 160, 208, 104, 2, 16
3, 154, 224, 68, 5, 26, 192, 0, 20, 67, 163, 7, 128, 34, 160, 10, 0, 4

2, 170, 52, 0, 4, 86, 32, 0, 53, 21, 194, 185, 128, 174, 86, 34, 10, 1
0, 242, 58, 170, 32, 0, 43, 148, 0, 84, 0, 14, 141, 86, 40, 168, 128,
0, 0, 0, 175, 0, 0, 7, 40, 10, 8, 130, 168, 170, 8, 224, 21, 64, 17, 8
5, 81, 21, 170, 249, 73, 62, 193, 127, 64, 190, 201, 244, 173, 173, 23,
168, 121, 12, 141, 100, 172, 180, 219, 238, 88, 31, 90, 174, 244, 86, 2
50, 55, 192, 159, 63, 126, 95, 105, 124, 59, 215, 252, 93, 1, 80, 0, 2
1, 68, 123, 16, 0, 0, 0, 0, 4, 68, 64, 106, 34, 170, 2, 56, 84, 16, 84,
68, 68, 68, 21, 26, 128, 225, 81, 21, 194, 48, 20, 107, 0, 68, 114, 17
0, 8, 40, 240, 0, 17, 17, 224, 3, 149, 90, 52, 0, 17, 168, 2, 160, 12,
114, 143, 84, 99, 156, 28, 196, 84, 81, 206, 35, 170, 136, 128, 0, 231,
0, 138, 0, 42, 61, 26, 40, 244, 104, 128, 0, 0, 3, 156, 0, 0, 40, 170,
40, 128, 131, 133, 112, 10, 138, 0, 57, 88, 170, 170, 208, 84, 124, 17
4, 151, 219, 141, 135, 236, 30, 206, 207, 91, 3, 97, 230, 18, 230, 89,
209, 216, 92, 179, 57, 169, 137, 168, 177, 212, 250, 79, 231, 231, 230,
20, 79, 55, 249, 218, 169, 160, 42, 0, 14, 85, 70, 160, 0, 0, 0, 0, 0,
136, 168, 49, 16, 69, 0, 20, 64, 81, 16, 106, 8, 8, 212, 28, 162, 53, 9
2, 130, 32, 13, 104, 0, 42, 171, 81, 85, 84, 84, 0, 65, 7, 8, 174, 80,
26, 128, 0, 214, 128, 10, 128, 3, 212, 26, 56, 70, 2, 10, 43, 200, 239,
65, 160, 0, 57, 84, 68, 81, 64, 232, 213, 6, 136, 162, 136, 128, 0, 0,
15, 80, 0, 1, 92, 138, 168, 138, 40, 3, 156, 130, 142, 106, 142, 70, 18
8, 106, 138, 42, 0, 233, 22, 254, 151, 233, 63, 164, 158, 157, 233, 24
4, 150, 154, 218, 248, 157, 158, 145, 228, 221, 114, 169, 190, 198, 12
2, 20, 126, 222, 173, 153, 252, 190, 241, 63, 61, 248, 182, 165, 99, 16
1, 209, 168, 128, 174, 80, 106, 32, 0, 2, 160, 0, 0, 0, 49, 4, 106, 16
0, 0, 34, 136, 170, 208, 65, 170, 209, 16, 107, 208, 26, 170, 130, 0, 1
41, 106, 128, 29, 0, 69, 84, 0, 84, 84, 0, 0, 122, 141, 17, 0, 6, 162,
0, 56, 26, 0, 231, 0, 197, 26, 168, 2, 40, 229, 226, 168, 173, 0, 0, 12
0, 168, 40, 0, 229, 80, 17, 26, 175, 4, 94, 106, 138, 32, 2, 185, 64, 2
1, 80, 65, 202, 160, 0, 168, 162, 170, 138, 162, 42, 56, 106, 138, 130,
128, 138, 175, 145, 160, 247, 31, 209, 95, 127, 190, 181, 227, 31, 125,
25, 215, 177, 56, 67, 147, 111, 22, 29, 190, 115, 89, 149, 209, 105, 12
5, 47, 243, 239, 229, 127, 152, 252, 86, 183, 204, 248, 55, 160, 131, 6
5, 92, 3, 90, 10, 128, 10, 32, 0, 0, 0, 136, 209, 138, 128, 130, 136, 4
2, 42, 141, 81, 1, 173, 4, 86, 170, 32, 42, 181, 81, 4, 96, 0, 14, 112,
10, 231, 53, 128, 0, 0, 0, 57, 68, 64, 64, 0, 98, 34, 142, 112, 48, 64,
28, 52, 4, 87, 43, 0, 21, 220, 129, 160, 0, 2, 170, 138, 0, 29, 0, 0, 1
06, 168, 8, 208, 21, 7, 43, 81, 234, 32, 225, 85, 160, 57, 192, 208, 2
8, 42, 40, 160, 170, 10, 8, 162, 40, 160, 215, 35, 186, 203, 219, 125,
243, 246, 198, 170, 241, 236, 149, 125, 38, 86, 150, 19, 171, 186, 247,
228, 157, 31, 51, 31, 182, 77, 205, 103, 205, 95, 147, 23, 62, 13, 224,
12, 107, 129, 0, 80, 4, 104, 128, 2, 162, 160, 0, 0, 2, 13, 69, 68, 64,
65, 90, 160, 8, 168, 228, 64, 98, 0, 34, 162, 32, 228, 1, 163, 4, 16, 8
0, 122, 138, 117, 151, 198, 59, 0, 1, 205, 0, 5, 20, 65, 65, 4, 4, 106,
0, 245, 26, 213, 17, 20, 21, 0, 21, 205, 64, 21, 252, 85, 168, 0, 0, 3,
149, 64, 1, 206, 0, 0, 5, 26, 215, 34, 168, 138, 0, 170, 209, 84, 122,
32, 163, 129, 26, 170, 229, 0, 1, 85, 28, 0, 40, 138, 130, 10, 175, 14
7, 63, 221, 191, 99, 175, 52, 177, 244, 241, 231, 73, 235, 160, 176, 17
6, 225, 78, 168, 230, 156, 164, 82, 154, 185, 222, 161, 130, 252, 164,
190, 252, 209, 241, 14, 108, 4, 20, 0, 4, 104, 32, 174, 98, 160, 0, 10,
138, 128, 35, 65, 1, 17, 194, 8, 168, 42, 53, 85, 81, 17, 173, 80, 16,
84, 70, 138, 0, 208, 21, 136, 128, 175, 21, 95, 54, 85, 95, 38, 0, 0, 5
7, 4, 0, 5, 28, 138, 141, 16, 17, 16, 122, 131, 68, 65, 170, 160, 2, 4
0, 162, 1, 208, 226, 173, 64, 1, 81, 200, 131, 149, 64, 1, 94, 0, 2, 18
5, 168, 8, 40, 168, 40, 130, 170, 162, 136, 224, 16, 87, 10, 130, 60,
0, 21, 1, 85, 81, 71, 10, 136, 128, 35, 151, 189, 190, 135, 239, 255,
0, 208, 75, 169, 220, 244, 221, 217, 221, 246, 150, 146, 165, 194, 56,
69, 239, 21, 34, 187, 51, 232, 54, 61, 55, 63, 16, 103, 255, 0, 49, 19

0, 56, 227, 207, 155, 64, 0, 0, 68, 71, 10, 28, 192, 0, 0, 17, 175, 96,
138, 128, 138, 10, 32, 128, 136, 56, 70, 163, 81, 206, 68, 106, 136, 3
2, 32, 130, 10, 238, 220, 184, 34, 185, 92, 143, 116, 251, 42, 46, 109,
98, 0, 10, 138, 173, 84, 0, 81, 65, 81, 162, 8, 209, 92, 162, 53, 84, 9
6, 0, 0, 213, 20, 1, 207, 94, 44, 64, 1, 65, 200, 138, 228, 80, 17, 65,
234, 0, 168, 228, 4, 0, 0, 21, 7, 2, 168, 130, 136, 168, 162, 184, 20,
0, 0, 5, 85, 20, 81, 205, 64, 65, 71, 75, 216, 125, 105, 250, 137, 125,
121, 35, 164, 171, 14, 105, 97, 50, 109, 143, 88, 51, 219, 14, 79, 22,
66, 139, 7, 77, 22, 183, 222, 60, 63, 199, 62, 36, 252, 195, 163, 233,
11, 146, 52, 0, 0, 0, 94, 220, 83, 152, 0, 0, 3, 81, 194, 35, 145, 1, 8
4, 65, 90, 32, 209, 85, 17, 17, 163, 129, 16, 4, 104, 128, 130, 162, 24
5, 153, 11, 136, 247, 35, 214, 70, 131, 208, 60, 122, 51, 70, 181, 92,
212, 5, 28, 141, 0, 85, 80, 5, 230, 13, 104, 175, 1, 5, 6, 32, 40, 34,
43, 85, 64, 21, 206, 119, 6, 162, 10, 130, 138, 163, 92, 10, 8, 168, 16
2, 185, 64, 1, 202, 53, 0, 0, 0, 5, 87, 8, 168, 168, 40, 174, 80, 5, 6
4, 0, 81, 69, 80, 28, 136, 0, 11, 210, 199, 220, 63, 77, 126, 137, 211,
247, 157, 36, 144, 215, 207, 151, 218, 79, 78, 61, 228, 191, 167, 120,
208, 98, 187, 181, 85, 199, 165, 104, 190, 63, 249, 203, 243, 31, 195,
235, 107, 248, 48, 230, 2, 40, 0, 15, 145, 25, 141, 64, 0, 0, 68, 85,
0, 17, 64, 64, 6, 160, 130, 42, 34, 2, 40, 168, 138, 34, 177, 4, 65, 8
1, 95, 42, 85, 111, 46, 189, 23, 181, 214, 131, 208, 125, 79, 228, 236,
215, 38, 162, 32, 163, 64, 81, 81, 90, 168, 174, 20, 65, 88, 130, 32, 5
7, 170, 162, 40, 53, 160, 168, 168, 52, 85, 1, 85, 194, 241, 96, 230, 1
85, 17, 202, 138, 8, 228, 80, 65, 64, 85, 84, 28, 7, 70, 136, 128, 160,
32, 228, 65, 92, 10, 2, 42, 138, 224, 20, 84, 64, 5, 28, 130, 168, 8, 1
0, 43, 149, 23, 166, 179, 244, 23, 244, 7, 93, 115, 214, 107, 216, 73,
231, 39, 188, 142, 18, 208, 155, 34, 87, 126, 113, 227, 245, 100, 115,
83, 233, 222, 5, 65, 249, 139, 242, 159, 206, 245, 145, 216, 214, 180,
4, 80, 0, 233, 62, 189, 141, 112, 196, 0, 1, 1, 64, 0, 0, 65, 81, 26,
3, 68, 17, 20, 20, 16, 1, 26, 128, 34, 190, 77, 146, 87, 44, 253, 218,
125, 127, 169, 202, 250, 87, 157, 252, 139, 228, 40, 136, 130, 43, 68,
5, 81, 65, 21, 81, 194, 0, 212, 0, 26, 138, 224, 1, 168, 128, 42, 32, 1
0, 40, 157, 17, 83, 138, 40, 160, 42, 128, 53, 84, 0, 0, 1, 92, 160, 5
7, 90, 128, 60, 68, 5, 16, 21, 84, 107, 128, 122, 128, 42, 168, 13, 69,
17, 192, 42, 170, 8, 2, 187, 188, 216, 103, 95, 75, 253, 134, 247, 13,
110, 141, 36, 175, 62, 93, 59, 71, 115, 100, 71, 146, 238, 179, 100, 24
5, 147, 195, 151, 25, 218, 108, 92, 191, 86, 225, 224, 191, 57, 254, 10
8, 124, 201, 65, 152, 70, 52, 104, 0, 0, 189, 229, 87, 32, 13, 104, 10,
128, 130, 128, 0, 0, 8, 168, 141, 1, 168, 163, 90, 56, 20, 64, 4, 96, 1
42, 232, 211, 164, 237, 138, 104, 227, 251, 158, 83, 238, 28, 55, 167,
45, 231, 193, 127, 45, 231, 17, 4, 17, 16, 104, 174, 80, 28, 8, 224, 2
0, 104, 136, 0, 140, 85, 112, 42, 53, 160, 0, 8, 163, 145, 0, 113, 205,
170, 160, 34, 185, 206, 107, 80, 80, 0, 0, 1, 234, 2, 185, 17, 5, 85, 6
9, 21, 16, 107, 148, 80, 69, 85, 112, 3, 133, 81, 81, 17, 20, 85, 0, 8
4, 86, 138, 138, 238, 157, 172, 221, 232, 223, 81, 126, 158, 236, 46, 1
11, 23, 172, 142, 92, 218, 206, 9, 18, 108, 14, 157, 229, 204, 237, 22
3, 179, 211, 141, 164, 140, 157, 230, 139, 103, 138, 249, 235, 242, 14
3, 229, 138, 175, 60, 230, 196, 68, 64, 21, 0, 117, 131, 96, 180, 1, 13
6, 0, 0, 0, 162, 10, 32, 0, 34, 52, 16, 65, 26, 10, 42, 180, 4, 106, 7,
94, 220, 201, 30, 145, 244, 142, 126, 223, 223, 233, 254, 164, 188, 21
9, 225, 99, 99, 127, 59, 126, 81, 140, 136, 8, 141, 64, 7, 168, 170, 3
4, 43, 132, 85, 4, 96, 0, 32, 34, 130, 52, 0, 0, 3, 167, 53, 30, 208, 2
28, 142, 84, 65, 23, 163, 213, 26, 209, 20, 0, 0, 1, 206, 0, 20, 17, 8
4, 81, 0, 114, 43, 92, 138, 42, 170, 128, 168, 170, 57, 85, 162, 8, 42,
128, 174, 106, 8, 40, 249, 42, 255, 0, 66, 246, 223, 213, 223, 71, 215,
217, 90, 140, 120, 112, 139, 203, 151, 14, 61, 121, 77, 149, 38, 67, 16
4, 75, 235, 35, 129, 89, 162, 235, 162, 209, 124, 207, 240, 79, 194, 9

4, 69, 131, 143, 204, 104, 136, 0, 3, 175, 105, 248, 160, 2, 35, 64, 0,
80, 114, 136, 35, 64, 0, 26, 34, 32, 136, 128, 0, 0, 13, 104, 249, 46,
111, 91, 255, 0, 209, 162, 87, 183, 249, 183, 222, 183, 58, 44, 69, 16
6, 95, 226, 175, 201, 10, 126, 40, 162, 12, 64, 7, 56, 7, 141, 69, 114,
128, 34, 34, 42, 0, 35, 20, 64, 69, 0, 0, 1, 81, 71, 243, 121, 192, 81,
90, 11, 209, 234, 35, 90, 0, 0, 3, 144, 69, 30, 141, 87, 42, 10, 160, 5
7, 16, 81, 81, 64, 28, 224, 80, 5, 81, 85, 16, 81, 5, 21, 30, 53, 1, 2
1, 206, 124, 185, 146, 255, 0, 88, 254, 201, 212, 233, 116, 111, 227, 1
99, 139, 90, 200, 188, 248, 62, 63, 2, 100, 222, 210, 18, 68, 233, 178,
35, 164, 45, 20, 73, 27, 15, 158, 126, 101, 252, 227, 241, 220, 46, 61,
141, 26, 136, 0, 10, 109, 49, 98, 0, 3, 16, 0, 232, 2, 40, 7, 48, 0, 6,
170, 52, 106, 32, 52, 112, 40, 128, 34, 18, 45, 98, 88, 122, 95, 215, 1
58, 211, 234, 86, 191, 88, 122, 60, 232, 25, 234, 124, 100, 191, 197, 1
11, 141, 168, 249, 32, 3, 16, 1, 206, 5, 1, 21, 202, 0, 212, 64, 0, 26,
208, 0, 0, 0, 0, 81, 0, 114, 241, 1, 0, 58, 57, 84, 70, 0, 7, 70, 180,
0, 84, 5, 71, 56, 0, 114, 168, 138, 130, 160, 170, 130, 185, 65, 64, 5,
80, 5, 4, 5, 30, 53, 0, 21, 202, 189, 108, 253, 119, 246, 159, 121, 17
6, 209, 105, 217, 26, 39, 46, 17, 210, 11, 23, 169, 202, 23, 75, 30, 21
0, 56, 204, 180, 145, 45, 31, 81, 127, 87, 123, 161, 119, 137, 126, 11
0, 252, 119, 243, 69, 79, 54, 8, 192, 0, 28, 158, 131, 231, 138, 0, 2,
53, 0, 7, 14, 69, 1, 20, 98, 0, 32, 12, 68, 68, 17, 163, 148, 20, 21, 2
1, 77, 60, 174, 255, 0, 69, 251, 95, 213, 63, 85, 233, 183, 241, 45, 22
4, 29, 166, 86, 252, 187, 248, 143, 243, 236, 6, 32, 3, 90, 10, 142, 11
2, 0, 10, 240, 17, 70, 180, 0, 17, 168, 8, 160, 42, 42, 0, 0, 168, 1, 2
09, 156, 128, 115, 65, 85, 84, 21, 0, 7, 10, 136, 60, 107, 64, 84, 115,
128, 5, 28, 162, 10, 14, 6, 138, 229, 28, 128, 130, 168, 168, 143, 84,
16, 80, 114, 32, 53, 71, 57, 78, 147, 126, 161, 253, 137, 244, 107, 75,
253, 35, 99, 197, 137, 22, 47, 56, 188, 28, 201, 12, 225, 217, 100, 24
7, 108, 139, 137, 18, 92, 54, 202, 186, 242, 234, 85, 151, 231, 183, 22
5, 118, 110, 167, 143, 49, 26, 128, 2, 135, 166, 121, 131, 128, 0, 14,
96, 0, 169, 208, 1, 4, 86, 0, 2, 35, 81, 17, 1, 162, 168, 14, 85, 28, 2
31, 125, 3, 79, 244, 71, 212, 223, 68, 253, 43, 222, 84, 151, 69, 111,
85, 157, 168, 249, 95, 240, 99, 194, 162, 243, 64, 1, 4, 68, 85, 112,
8, 161, 209, 80, 4, 106, 0, 7, 49, 80, 0, 21, 202, 196, 84, 0, 120, 19
2, 5, 228, 42, 57, 16, 84, 114, 138, 0, 3, 149, 68, 20, 17, 26, 170, 14
3, 0, 1, 85, 200, 168, 174, 112, 35, 85, 200, 170, 130, 136, 10, 160, 1
62, 128, 10, 138, 32, 136, 170, 170, 227, 165, 143, 234, 183, 223, 155,
185, 151, 243, 150, 60, 120, 144, 160, 242, 229, 193, 56, 114, 144, 23
1, 58, 60, 246, 203, 177, 147, 46, 66, 118, 179, 73, 22, 253, 246, 191,
14, 126, 68, 249, 95, 207, 92, 121, 35, 80, 0, 20, 79, 98, 241, 197, 0,
1, 173, 0, 0, 21, 202, 32, 160, 141, 64, 17, 17, 26, 8, 8, 136, 224, 2
8, 231, 57, 242, 123, 125, 171, 237, 31, 74, 253, 137, 168, 141, 203, 1
75, 40, 49, 249, 245, 113, 172, 240, 63, 198, 223, 145, 168, 249, 32, 2
08, 6, 162, 3, 156, 0, 2, 189, 20, 6, 181, 174, 0, 107, 85, 1, 16, 85,
20, 64, 1, 71, 48, 85, 83, 131, 132, 81, 170, 142, 1, 84, 0, 5, 21, 17
5, 85, 65, 0, 20, 20, 69, 65, 206, 0, 122, 170, 0, 10, 128, 162, 2, 13
0, 168, 160, 34, 168, 32, 13, 112, 174, 85, 233, 103, 251, 109, 246, 4
5, 166, 162, 195, 191, 40, 81, 161, 199, 143, 199, 156, 126, 81, 227, 5
9, 187, 185, 36, 132, 157, 48, 178, 156, 217, 86, 173, 124, 185, 190, 1
33, 241, 175, 202, 95, 141, 57, 248, 220, 6, 32, 0, 175, 57, 251, 55, 1
40, 128, 2, 35, 69, 64, 20, 17, 69, 81, 28, 2, 53, 1, 26, 141, 84, 21,
17, 17, 81, 65, 207, 119, 89, 19, 189, 3, 239, 111, 190, 126, 154, 233,
68, 238, 12, 137, 80, 145, 21, 13, 231, 194, 95, 147, 95, 63, 114, 98,
34, 52, 1, 136, 168, 175, 0, 0, 85, 112, 13, 106, 40, 8, 212, 4, 65, 1
7, 206, 0, 1, 170, 162, 163, 156, 162, 71, 232, 209, 81, 1, 68, 71, 42,
128, 0, 2, 188, 84, 0, 0, 85, 84, 64, 112, 162, 184, 80, 84, 81, 0, 0,
21, 69, 20, 0, 20, 16, 64, 84, 87, 245, 114, 109, 127, 126, 61, 147, 9

1, 164, 148, 144, 235, 163, 112, 143, 199, 135, 22, 242, 229, 92, 72, 8
8, 233, 213, 39, 186, 84, 155, 117, 179, 143, 63, 157, 132, 253, 190, 1
27, 243, 135, 242, 23, 200, 43, 248, 32, 53, 160, 174, 83, 159, 178, 24
8, 216, 130, 10, 196, 0, 20, 87, 53, 16, 85, 112, 0, 139, 204, 17, 168,
43, 69, 106, 32, 0, 170, 254, 157, 245, 62, 207, 237, 191, 169, 222, 23
3, 167, 163, 175, 23, 156, 104, 92, 34, 242, 231, 206, 223, 242, 251, 2
43, 127, 206, 185, 140, 70, 32, 3, 81, 28, 225, 80, 0, 7, 43, 145, 17,
168, 40, 53, 160, 141, 16, 7, 168, 0, 136, 138, 42, 244, 85, 19, 131, 1
44, 20, 24, 230, 160, 60, 80, 0, 1, 206, 20, 64, 6, 57, 92, 162, 162, 3
4, 170, 130, 131, 148, 5, 104, 138, 40, 168, 175, 69, 0, 1, 68, 4, 81,
21, 21, 242, 174, 61, 143, 221, 255, 0, 94, 117, 26, 15, 64, 94, 21, 21
3, 145, 249, 241, 225, 198, 55, 54, 240, 227, 195, 163, 185, 115, 87, 7
3, 139, 101, 215, 182, 169, 31, 46, 37, 133, 253, 197, 15, 138, 255, 0,
56, 153, 58, 88, 236, 70, 162, 53, 71, 170, 39, 175, 120, 248, 131, 80,
64, 0, 87, 140, 16, 84, 232, 52, 112, 8, 192, 68, 68, 84, 21, 24, 130,
10, 175, 31, 214, 103, 208, 95, 112, 253, 207, 237, 18, 228, 199, 141,
217, 141, 78, 53, 149, 208, 156, 239, 24, 252, 75, 241, 122, 182, 13, 1
04, 7, 49, 168, 229, 7, 13, 0, 7, 14, 115, 78, 104, 40, 35, 1, 21, 160,
136, 225, 192, 2, 53, 5, 123, 220, 162, 199, 81, 20, 69, 17, 4, 71, 42,
128, 0, 43, 149, 85, 16, 1, 143, 21, 226, 8, 215, 42, 128, 42, 168, 16
0, 130, 40, 160, 170, 42, 162, 128, 56, 26, 10, 136, 10, 61, 253, 181,
223, 163, 223, 166, 27, 107, 141, 87, 104, 53, 85, 220, 78, 92, 91, 14,
52, 84, 229, 209, 221, 120, 113, 136, 143, 228, 249, 211, 244, 83, 249,
88, 214, 116, 213, 65, 244, 111, 40, 252, 2, 240, 47, 52, 224, 113, 99,
68, 84, 81, 123, 251, 39, 137, 177, 4, 70, 170, 42, 0, 0, 0, 170, 57, 2
0, 3, 152, 52, 16, 84, 17, 24, 0, 249, 108, 116, 175, 70, 253, 40, 253,
27, 214, 74, 94, 252, 29, 217, 88, 145, 107, 42, 224, 182, 195, 230, 1
5, 195, 223, 57, 98, 8, 163, 210, 43, 68, 114, 128, 162, 0, 2, 170, 16
8, 140, 0, 17, 128, 8, 34, 35, 149, 81, 65, 81, 26, 139, 209, 234, 168,
177, 220, 0, 0, 136, 213, 85, 80, 0, 5, 87, 40, 128, 0, 138, 174, 80, 1
04, 142, 85, 71, 162, 8, 175, 4, 20, 1, 65, 202, 0, 2, 136, 168, 42, 0,
43, 222, 251, 79, 212, 63, 210, 207, 68, 186, 212, 196, 169, 172, 128,
198, 55, 131, 120, 68, 141, 31, 131, 165, 51, 135, 78, 208, 97, 241, 23
1, 218, 226, 254, 251, 189, 149, 27, 237, 99, 250, 231, 153, 254, 105,
124, 159, 240, 71, 53, 130, 196, 64, 1, 214, 30, 153, 228, 35, 90, 8, 1
88, 250, 28, 192, 0, 5, 30, 196, 232, 138, 3, 90, 52, 68, 5, 65, 26, 12
8, 189, 239, 160, 88, 122, 191, 233, 23, 233, 176, 174, 123, 122, 244,
238, 254, 28, 226, 197, 167, 173, 231, 160, 252, 231, 252, 104, 130, 19
6, 7, 73, 235, 89, 193, 1, 234, 2, 136, 0, 0, 170, 227, 152, 0, 140, 0,
17, 121, 138, 240, 0, 84, 68, 87, 116, 81, 14, 66, 40, 0, 8, 32, 42, 12
8, 0, 43, 192, 0, 4, 87, 56, 1, 5, 30, 42, 42, 10, 42, 42, 0, 2, 170, 1
68, 0, 40, 32, 40, 40, 11, 211, 185, 233, 63, 171, 223, 119, 122, 38, 1
38, 108, 26, 58, 216, 138, 222, 60, 89, 207, 156, 120, 28, 14, 208, 99,
190, 55, 94, 220, 88, 75, 191, 213, 72, 189, 139, 69, 38, 94, 183, 121,
243, 79, 229, 167, 230, 45, 111, 24, 13, 107, 84, 21, 166, 218, 239, 20
4, 17, 17, 168, 53, 85, 173, 0, 0, 21, 237, 17, 28, 170, 53, 16, 68, 10
4, 0, 12, 81, 93, 63, 79, 7, 208, 255, 0, 82, 190, 239, 209, 202, 232,
227, 172, 142, 138, 49, 34, 85, 86, 67, 149, 101, 249, 91, 249, 55, 13,
140, 71, 72, 177, 163, 229, 204, 21, 224, 162, 162, 0, 2, 40, 60, 65,
4, 17, 168, 0, 28, 199, 42, 138, 138, 157, 57, 128, 245, 112, 139, 193,
81, 64, 0, 16, 0, 20, 5, 7, 130, 43, 92, 42, 2, 170, 168, 0, 57, 192,
8, 42, 184, 98, 160, 160, 142, 112, 0, 40, 2, 40, 160, 224, 7, 246, 15
1, 180, 253, 186, 250, 167, 125, 171, 155, 85, 151, 173, 226, 28, 184,
49, 145, 227, 197, 111, 52, 137, 95, 217, 176, 86, 77, 230, 98, 126, 15
4, 253, 154, 238, 181, 212, 58, 123, 237, 164, 63, 131, 191, 28, 188, 6
3, 47, 92, 212, 67, 171, 17, 169, 235, 25, 124, 146, 57, 172, 106, 8, 1
68, 136, 128, 0, 10, 170, 136, 0, 32, 8, 136, 128, 136, 170, 142, 94, 1

86, 29, 47, 166, 125, 155, 250, 253, 142, 191, 147, 195, 172, 174, 242,
14, 77, 99, 33, 68, 172, 172, 177, 179, 252, 26, 248, 190, 12, 110, 10
5, 38, 206, 151, 131, 90, 138, 174, 5, 16, 0, 6, 184, 21, 192, 138, 35,
81, 128, 0, 136, 213, 114, 138, 131, 156, 140, 7, 168, 14, 142, 170, 0,
0, 136, 170, 128, 160, 40, 14, 5, 1, 85, 16, 85, 85, 1, 200, 229, 0, 1,
85, 90, 138, 32, 170, 142, 85, 80, 16, 81, 5, 65, 71, 0, 14, 237, 105,
244, 159, 236, 159, 178, 250, 62, 253, 115, 185, 72, 28, 156, 206, 81,
184, 167, 40, 112, 184, 245, 88, 145, 218, 232, 238, 103, 72, 211, 110,
53, 212, 154, 59, 174, 245, 107, 180, 182, 210, 249, 55, 225, 127, 202,
190, 83, 80, 196, 58, 63, 131, 85, 61, 219, 200, 106, 5, 57, 181, 160,
13, 104, 0, 0, 116, 98, 0, 0, 0, 53, 162, 8, 170, 231, 186, 235, 213, 1
73, 126, 206, 253, 94, 184, 58, 73, 231, 218, 87, 110, 143, 88, 220, 18
4, 198, 133, 87, 194, 117, 151, 243, 143, 225, 116, 60, 91, 210, 235, 5
9, 197, 162, 32, 224, 0, 0, 0, 20, 84, 5, 114, 177, 141, 16, 80, 69, 6
8, 20, 21, 68, 122, 48, 87, 130, 53, 204, 0, 4, 84, 84, 69, 16, 81, 85,
1, 195, 145, 64, 85, 16, 85, 20, 85, 85, 69, 1, 80, 5, 1, 20, 84, 80, 2
8, 57, 192, 0, 128, 40, 0, 3, 186, 109, 254, 167, 253, 145, 212, 250, 1
27, 162, 118, 199, 102, 33, 242, 57, 241, 143, 29, 188, 160, 215, 240,
58, 183, 138, 243, 136, 206, 157, 21, 246, 187, 10, 59, 171, 187, 14, 4
4, 216, 119, 183, 151, 248, 171, 243, 63, 198, 52, 124, 209, 101, 199,
230, 142, 111, 191, 120, 68, 102, 185, 205, 98, 52, 1, 173, 20, 16, 0,
0, 0, 17, 64, 104, 212, 5, 78, 146, 19, 214, 61, 131, 232, 207, 216, 23
5, 42, 110, 160, 201, 110, 147, 50, 75, 163, 65, 143, 22, 182, 23, 41,
110, 254, 99, 241, 181, 228, 171, 156, 71, 6, 136, 138, 0, 0, 138, 10,
8, 57, 4, 7, 57, 83, 147, 0, 0, 84, 0, 21, 236, 65, 94, 199, 40, 49, 7,
42, 52, 16, 4, 5, 84, 5, 5, 71, 56, 106, 168, 162, 185, 168, 42, 168, 5
7, 192, 138, 10, 130, 42, 160, 168, 162, 130, 136, 14, 122, 128, 0, 0,
0, 0, 231, 89, 126, 166, 126, 146, 106, 189, 43, 91, 59, 33, 146, 134,
206, 124, 121, 241, 231, 199, 156, 72, 81, 185, 116, 145, 202, 11, 21,
178, 229, 71, 143, 107, 97, 118, 89, 88, 198, 179, 208, 173, 142, 247,
226, 175, 205, 223, 205, 140, 223, 6, 186, 116, 36, 86, 55, 222, 60, 2
1, 162, 184, 70, 177, 0, 68, 64, 64, 0, 0, 0, 26, 162, 131, 68, 4, 94,
211, 172, 190, 174, 247, 31, 214, 191, 76, 168, 133, 37, 156, 251, 119,
235, 43, 191, 71, 197, 173, 227, 22, 5, 116, 91, 47, 158, 127, 1, 240,
252, 37, 235, 48, 124, 152, 212, 17, 64, 1, 24, 231, 3, 144, 69, 84, 6
4, 85, 120, 222, 109, 17, 68, 0, 69, 5, 69, 64, 28, 162, 34, 1, 208, 7
0, 181, 1, 70, 170, 170, 10, 138, 42, 15, 84, 17, 195, 129, 85, 128, 22
9, 69, 122, 168, 136, 142, 69, 69, 20, 0, 112, 2, 42, 131, 156, 0, 0,
0, 0, 10, 231, 217, 126, 129, 126, 189, 237, 125, 42, 215, 182, 51, 33,
9, 141, 229, 203, 155, 121, 243, 141, 25, 56, 34, 204, 231, 86, 216, 25
3, 219, 10, 100, 233, 253, 229, 94, 219, 200, 131, 167, 237, 43, 210, 1
78, 191, 150, 95, 141, 249, 30, 81, 159, 54, 24, 222, 77, 247, 111, 6,
84, 112, 163, 90, 214, 130, 11, 204, 0, 0, 0, 4, 17, 28, 168, 160, 15,
98, 205, 191, 246, 111, 94, 253, 157, 246, 234, 168, 220, 100, 57, 253,
157, 214, 95, 102, 178, 31, 30, 49, 224, 84, 195, 188, 252, 206, 252, 1
45, 204, 77, 213, 121, 255, 0, 49, 172, 17, 170, 170, 128, 53, 170, 24
0, 84, 1, 194, 32, 160, 228, 107, 90, 0, 8, 168, 40, 0, 7, 70, 162, 8,
161, 208, 98, 53, 4, 122, 180, 20, 20, 5, 30, 3, 91, 208, 85, 5, 0, 20,
87, 56, 70, 181, 202, 2, 170, 160, 40, 224, 0, 5, 114, 128, 0, 8, 10,
0, 175, 235, 113, 250, 57, 250, 181, 174, 245, 43, 62, 216, 92, 116, 6
5, 156, 163, 179, 159, 54, 51, 147, 90, 146, 18, 4, 6, 198, 233, 25, 17
4, 155, 38, 84, 203, 221, 28, 233, 26, 20, 143, 189, 208, 254, 123, 25
4, 68, 124, 170, 185, 222, 157, 248, 17, 249, 39, 208, 31, 62, 170, 13
6, 160, 196, 98, 0, 136, 53, 65, 0, 0, 4, 84, 106, 131, 128, 124, 158,
11, 184, 221, 125, 11, 250, 233, 244, 7, 74, 164, 115, 187, 244, 237, 2
15, 175, 68, 127, 62, 17, 185, 115, 135, 159, 141, 175, 252, 96, 252, 2
04, 181, 211, 121, 107, 4, 106, 35, 17, 94, 0, 168, 35, 144, 0, 114, 14

1, 4, 0, 17, 26, 0, 53, 20, 114, 3, 92, 160, 0, 128, 160, 35, 81, 81, 9
2, 8, 162, 162, 128, 225, 192, 34, 128, 10, 142, 85, 0, 87, 40, 136, 19
7, 28, 168, 241, 236, 5, 85, 21, 0, 84, 28, 56, 0, 1, 4, 112, 2, 244, 1
80, 217, 126, 151, 254, 142, 232, 253, 58, 245, 216, 156, 148, 46, 92,
213, 156, 249, 113, 227, 29, 6, 181, 209, 234, 17, 189, 229, 208, 172,
206, 18, 102, 89, 104, 173, 221, 101, 167, 185, 198, 239, 181, 103, 22
4, 247, 203, 191, 58, 230, 230, 112, 34, 114, 230, 159, 71, 124, 224, 1
28, 0, 140, 107, 64, 1, 5, 230, 0, 0, 32, 173, 71, 2, 138, 157, 44, 12
1, 244, 247, 255, 0, 161, 191, 83, 126, 137, 181, 170, 128, 189, 158, 1
18, 237, 215, 163, 186, 63, 159, 14, 44, 143, 89, 159, 39, 255, 0, 53,
184, 27, 143, 35, 69, 68, 70, 163, 80, 115, 133, 28, 193, 92, 141, 0, 2
32, 34, 42, 34, 0, 12, 64, 6, 141, 80, 64, 114, 136, 42, 3, 144, 21, 1
7, 4, 85, 5, 1, 21, 28, 3, 133, 20, 64, 81, 20, 87, 128, 42, 43, 132, 2
6, 2, 170, 185, 20, 81, 69, 0, 1, 84, 71, 40, 32, 138, 168, 131, 128, 2
1, 242, 118, 223, 169, 95, 160, 155, 109, 213, 244, 188, 6, 90, 154, 1
1, 186, 245, 78, 124, 120, 194, 97, 24, 236, 69, 141, 107, 38, 190, 21,
108, 217, 252, 157, 123, 160, 144, 217, 18, 246, 181, 211, 189, 63, 91,
249, 251, 249, 221, 249, 249, 138, 182, 135, 69, 196, 226, 223, 168, 6
2, 104, 134, 128, 0, 49, 26, 34, 42, 0, 140, 17, 64, 69, 68, 115, 92,
2, 175, 86, 63, 66, 186, 255, 0, 162, 191, 91, 254, 185, 141, 7, 135, 3
8, 246, 103, 94, 143, 233, 39, 171, 158, 206, 44, 229, 14, 170, 173, 19
0, 23, 248, 179, 228, 222, 76, 196, 81, 173, 26, 208, 115, 135, 56, 98,
185, 26, 128, 170, 224, 17, 160, 32, 49, 0, 65, 26, 0, 2, 168, 208, 1,
202, 168, 48, 80, 81, 20, 112, 32, 40, 10, 43, 128, 0, 1, 92, 40, 2, 17
0, 180, 64, 87, 40, 40, 56, 69, 20, 4, 81, 85, 68, 81, 0, 112, 128, 16
0, 3, 250, 204, 253, 116, 253, 7, 244, 187, 141, 29, 167, 159, 98, 168,
251, 40, 231, 179, 147, 56, 69, 231, 197, 59, 204, 238, 202, 24, 253, 2
26, 84, 232, 248, 112, 208, 107, 37, 72, 131, 210, 214, 233, 186, 141,
134, 183, 203, 127, 62, 63, 31, 124, 227, 93, 136, 205, 189, 240, 57, 2
53, 13, 226, 20, 98, 0, 2, 43, 70, 136, 212, 0, 104, 212, 80, 68, 112,
138, 7, 73, 46, 107, 125, 19, 77, 238, 127, 161, 223, 163, 181, 144, 24
9, 242, 227, 217, 68, 235, 34, 87, 87, 189, 121, 115, 103, 8, 117, 245,
82, 62, 12, 252, 144, 240, 158, 77, 17, 90, 214, 57, 160, 116, 85, 81,
160, 168, 128, 225, 85, 21, 163, 64, 70, 171, 70, 162, 42, 0, 0, 0, 0,
3, 213, 170, 199, 10, 213, 65, 69, 80, 17, 69, 17, 85, 192, 0, 0, 229,
81, 1, 71, 3, 65, 202, 160, 160, 61, 0, 20, 17, 69, 114, 130, 2, 160, 1
60, 0, 10, 143, 233, 39, 245, 227, 244, 35, 211, 38, 232, 110, 113, 56,
74, 94, 175, 98, 40, 211, 156, 62, 92, 73, 40, 206, 53, 253, 95, 198, 1
90, 109, 141, 247, 77, 239, 42, 86, 72, 189, 172, 155, 191, 208, 107, 1
08, 62, 115, 252, 42, 193, 193, 249, 153, 31, 210, 139, 151, 176, 224,
242, 45, 84, 0, 0, 70, 160, 212, 114, 59, 185, 9, 168, 160, 7, 94, 64,
174, 145, 50, 83, 36, 123, 167, 167, 126, 188, 125, 81, 203, 132, 70, 6
1, 156, 90, 215, 77, 155, 34, 71, 100, 107, 57, 241, 227, 22, 158, 36,
191, 197, 31, 205, 218, 190, 109, 104, 208, 98, 0, 174, 87, 34, 34, 13
0, 3, 156, 136, 174, 70, 136, 8, 212, 233, 205, 163, 85, 0, 0, 0, 0, 0,
114, 162, 177, 206, 68, 69, 20, 20, 0, 28, 40, 160, 2, 160, 3, 149, 81,
26, 241, 206, 6, 131, 192, 28, 213, 28, 10, 32, 162, 40, 225, 69, 17, 6
4, 0, 0, 85, 85, 235, 55, 245, 251, 244, 11, 210, 228, 202, 189, 161, 2
43, 234, 14, 224, 168, 156, 156, 188, 163, 199, 228, 238, 140, 72, 108,
177, 44, 105, 167, 119, 244, 234, 43, 22, 192, 143, 165, 175, 235, 181,
191, 218, 220, 101, 254, 5, 248, 47, 230, 111, 26, 143, 5, 244, 182, 12
3, 232, 158, 52, 140, 68, 81, 0, 4, 106, 42, 202, 180, 182, 201, 67, 23
0, 214, 185, 64, 124, 136, 236, 14, 182, 118, 59, 204, 151, 164, 253, 1
29, 245, 215, 233, 166, 106, 177, 88, 229, 143, 204, 237, 222, 92, 137,
50, 157, 199, 139, 78, 124, 32, 82, 71, 147, 252, 233, 124, 207, 67, 19
7, 136, 208, 14, 96, 7, 64, 106, 43, 152, 60, 21, 81, 85, 26, 0, 140, 2
1, 17, 162, 0, 0, 42, 0, 0, 10, 228, 84, 65, 192, 142, 64, 80, 0, 87, 1

0, 0, 2, 162, 160, 57, 205, 16, 115, 149, 192, 136, 174, 64, 84, 114, 1
4, 1, 64, 0, 7, 56, 0, 0, 0, 21, 202, 117, 151, 123, 251, 101, 245, 17
4, 199, 77, 198, 210, 39, 153, 84, 39, 68, 67, 170, 41, 203, 135, 14,
2, 177, 150, 115, 108, 56, 213, 66, 200, 237, 54, 249, 254, 126, 152, 1
80, 50, 39, 225, 253, 95, 182, 251, 65, 167, 240, 127, 135, 127, 34, 22
7, 249, 228, 52, 175, 244, 188, 206, 243, 230, 215, 193, 99, 64, 17, 1
6, 37, 219, 162, 69, 175, 141, 203, 152, 8, 162, 167, 73, 145, 121, 18
1, 103, 219, 105, 126, 167, 249, 179, 234, 79, 211, 175, 182, 44, 51, 6
0, 248, 73, 68, 86, 117, 119, 121, 29, 164, 200, 94, 81, 152, 36, 122,
186, 118, 224, 255, 0, 25, 254, 79, 242, 30, 12, 70, 128, 35, 0, 30, 16
3, 91, 209, 20, 5, 81, 90, 230, 0, 13, 68, 6, 32, 0, 0, 42, 0, 0, 14, 6
9, 69, 65, 85, 81, 17, 202, 42, 10, 10, 224, 0, 5, 84, 16, 85, 112, 13
0, 57, 226, 130, 40, 57, 17, 192, 160, 34, 168, 42, 136, 10, 143, 0, 0,
0, 21, 224, 189, 244, 30, 215, 251, 57, 235, 59, 139, 206, 207, 145, 23
0, 25, 230, 40, 222, 242, 68, 103, 30, 92, 88, 213, 11, 139, 41, 182, 1
43, 227, 93, 31, 17, 203, 107, 181, 217, 99, 184, 71, 203, 250, 21, 24
7, 169, 165, 158, 187, 230, 159, 202, 239, 148, 60, 35, 75, 67, 232, 14
6, 60, 239, 215, 254, 61, 209, 101, 217, 92, 196, 115, 198, 244, 244, 1
26, 20, 116, 48, 220, 206, 60, 185, 128, 3, 156, 58, 111, 30, 28, 221,
181, 151, 187, 250, 219, 206, 62, 205, 253, 77, 212, 231, 106, 134, 18
2, 74, 199, 119, 123, 9, 146, 251, 117, 57, 198, 143, 197, 6, 85, 231,
79, 154, 127, 53, 126, 10, 199, 49, 168, 208, 1, 136, 3, 156, 12, 21, 1
92, 42, 170, 42, 52, 69, 17, 128, 35, 0, 5, 81, 170, 162, 32, 0, 0, 17
0, 208, 21, 85, 28, 0, 228, 28, 160, 34, 128, 3, 144, 64, 28, 40, 174,
85, 0, 21, 200, 40, 8, 170, 131, 84, 85, 87, 3, 71, 40, 0, 0, 0, 61, 6
5, 95, 170, 253, 2, 251, 187, 218, 247, 179, 224, 88, 206, 196, 224, 9
6, 60, 59, 204, 232, 47, 46, 103, 22, 184, 105, 109, 234, 55, 113, 243,
19, 23, 25, 156, 182, 208, 234, 116, 184, 158, 54, 61, 37, 123, 22, 63,
109, 232, 190, 87, 240, 159, 226, 79, 187, 50, 183, 211, 188, 114, 98,
124, 211, 125, 155, 102, 35, 147, 122, 73, 54, 123, 255, 0, 21, 171, 7
8, 124, 231, 215, 187, 143, 36, 5, 28, 229, 122, 202, 176, 165, 85, 24
7, 70, 122, 103, 213, 30, 177, 250, 89, 162, 117, 108, 35, 163, 221, 20
5, 205, 147, 97, 99, 35, 187, 184, 195, 131, 197, 122, 63, 133, 69, 63,
15, 205, 15, 201, 207, 49, 227, 201, 173, 65, 65, 24, 0, 175, 17, 88, 6
0, 5, 85, 17, 163, 85, 68, 96, 13, 104, 2, 142, 24, 61, 17, 7, 34, 0, 1
0, 163, 64, 85, 112, 2, 188, 0, 0, 0, 0, 1, 85, 84, 21, 227, 145, 81, 2
0, 114, 128, 34, 136, 13, 28, 160, 229, 84, 81, 68, 0, 0, 5, 120, 10, 1
16, 159, 245, 143, 235, 159, 160, 122, 60, 234, 139, 121, 149, 222, 51,
73, 217, 93, 38, 92, 137, 47, 123, 70, 240, 86, 186, 199, 211, 175, 42,
233, 117, 181, 158, 123, 167, 127, 158, 110, 230, 122, 39, 154, 118, 17
5, 217, 66, 244, 186, 107, 175, 81, 180, 248, 19, 242, 159, 235, 111, 1
55, 114, 222, 189, 132, 245, 95, 156, 124, 171, 231, 191, 83, 204, 96,
233, 120, 246, 181, 216, 122, 223, 131, 101, 135, 200, 166, 245, 79, 4
3, 233, 202, 59, 85, 202, 175, 234, 217, 132, 125, 78, 70, 117, 207, 21
3, 58, 107, 15, 182, 126, 220, 244, 233, 144, 107, 122, 189, 142, 17, 1
23, 77, 155, 50, 66, 182, 45, 124, 52, 233, 33, 91, 69, 158, 79, 231, 1
83, 230, 252, 7, 6, 49, 173, 78, 128, 140, 0, 87, 40, 8, 160, 42, 170,
177, 26, 215, 168, 140, 1, 173, 0, 115, 132, 68, 81, 16, 20, 64, 21, 8
5, 26, 2, 184, 84, 81, 206, 1, 80, 0, 0, 21, 0, 21, 202, 3, 213, 92, 8,
40, 231, 13, 64, 4, 16, 112, 10, 229, 7, 136, 208, 0, 5, 85, 80, 14, 15
7, 102, 254, 130, 126, 150, 123, 12, 203, 24, 51, 37, 39, 143, 97, 166,
186, 92, 169, 18, 230, 201, 120, 208, 71, 76, 210, 117, 218, 220, 203,
233, 157, 166, 249, 195, 221, 99, 193, 189, 216, 204, 242, 108, 159, 17
7, 187, 65, 65, 170, 244, 29, 222, 11, 224, 122, 15, 19, 196, 106, 247,
217, 156, 47, 207, 95, 20, 123, 237, 13, 7, 143, 34, 110, 126, 191, 24
9, 123, 150, 26, 187, 191, 173, 120, 87, 214, 95, 38, 79, 137, 1, 189,
228, 220, 213, 39, 5, 208, 37, 94, 155, 35, 232, 91, 239, 117, 246, 22

1, 79, 219, 223, 78, 44, 170, 222, 61, 58, 49, 98, 55, 164, 169, 242, 3
6, 247, 86, 87, 215, 71, 235, 43, 163, 227, 231, 104, 176, 63, 135, 19
1, 23, 86, 114, 99, 24, 142, 112, 35, 0, 21, 232, 160, 0, 61, 67, 152,
214, 171, 149, 173, 4, 70, 128, 61, 68, 106, 42, 0, 244, 68, 81, 85, 6
8, 98, 138, 40, 160, 61, 64, 81, 0, 28, 208, 21, 81, 1, 69, 81, 85, 23
8, 85, 85, 4, 81, 88, 173, 84, 107, 134, 170, 168, 43, 148, 114, 171, 8
1, 0, 0, 232, 0, 11, 34, 230, 243, 245, 79, 239, 61, 37, 139, 163, 147,
79, 63, 241, 251, 9, 210, 101, 73, 176, 153, 217, 210, 58, 72, 101, 18
0, 233, 144, 234, 253, 3, 209, 123, 121, 189, 14, 151, 25, 229, 94, 20
9, 139, 147, 232, 183, 25, 63, 24, 245, 235, 219, 202, 91, 173, 87, 16
9, 118, 249, 126, 159, 229, 238, 124, 54, 179, 188, 143, 229, 95, 145,
125, 255, 0, 25, 233, 63, 158, 51, 224, 253, 117, 244, 143, 199, 92, 23
6, 124, 14, 23, 208, 126, 53, 246, 87, 192, 62, 201, 146, 243, 52, 213,
195, 250, 11, 27, 229, 50, 121, 95, 89, 26, 202, 45, 71, 212, 254, 149,
245, 79, 161, 125, 77, 99, 41, 177, 154, 254, 124, 225, 68, 116, 174, 2
10, 36, 72, 238, 145, 161, 194, 108, 169, 146, 25, 3, 43, 7, 228, 255,
0, 198, 127, 18, 203, 68, 228, 198, 34, 188, 26, 208, 0, 87, 128, 1, 20
9, 80, 230, 49, 1, 68, 16, 70, 128, 175, 84, 70, 0, 168, 170, 138, 2, 1
68, 35, 28, 142, 5, 0, 115, 133, 71, 8, 128, 160, 128, 225, 1, 81, 84,
21, 94, 231, 57, 202, 128, 136, 130, 53, 90, 170, 0, 10, 174, 81, 64, 6
4, 0, 21, 202, 0, 47, 109, 87, 220, 255, 0, 120, 123, 79, 161, 244, 14
1, 27, 141, 135, 74, 15, 158, 45, 46, 164, 77, 147, 101, 45, 121, 241,
116, 171, 173, 38, 184, 235, 63, 61, 129, 210, 110, 153, 225, 251, 207,
23, 210, 109, 60, 127, 233, 248, 244, 24, 171, 221, 197, 190, 75, 107,
97, 232, 218, 124, 207, 132, 96, 168, 242, 217, 255, 0, 116, 242, 207,
157, 254, 113, 186, 249, 219, 223, 60, 31, 231, 251, 31, 173, 62, 230,
248, 239, 9, 236, 127, 32, 121, 167, 210, 156, 252, 135, 198, 190, 219,
242, 207, 148, 89, 238, 25, 63, 164, 101, 120, 17, 50, 147, 87, 232, 5
3, 90, 186, 31, 210, 29, 31, 182, 251, 150, 254, 37, 167, 72, 145, 223,
93, 19, 152, 178, 59, 74, 153, 33, 89, 6, 28, 110, 114, 109, 101, 112,
168, 197, 70, 252, 236, 249, 43, 230, 79, 10, 136, 198, 53, 162, 184, 1
07, 69, 64, 7, 168, 0, 61, 200, 214, 141, 104, 0, 35, 80, 28, 222, 128,
53, 160, 42, 136, 174, 69, 65, 65, 136, 240, 80, 0, 87, 138, 42, 32, 0,
2, 184, 64, 81, 64, 87, 57, 207, 114, 170, 32, 141, 64, 104, 142, 16, 8
0, 81, 94, 0, 168, 0, 0, 231, 0, 3, 229, 250, 31, 232, 255, 0, 181, 12
5, 5, 232, 54, 81, 248, 193, 89, 209, 124, 66, 162, 234, 198, 108, 219,
73, 71, 8, 156, 30, 55, 214, 253, 26, 198, 150, 63, 159, 71, 206, 250,
245, 55, 140, 92, 210, 250, 102, 7, 221, 168, 168, 157, 103, 175, 159,
5, 102, 122, 78, 194, 63, 141, 96, 61, 79, 230, 216, 122, 24, 30, 87, 2
43, 212, 239, 151, 63, 66, 190, 95, 249, 139, 234, 31, 162, 62, 143, 22
9, 226, 223, 57, 122, 103, 196, 30, 245, 244, 87, 229, 54, 31, 235, 42,
31, 139, 56, 123, 157, 47, 209, 246, 126, 119, 15, 23, 69, 244, 15, 14
0, 122, 228, 95, 89, 253, 8, 215, 107, 61, 199, 143, 41, 221, 43, 57, 1
99, 134, 196, 227, 215, 175, 121, 50, 251, 47, 40, 117, 241, 89, 103, 1
05, 33, 185, 252, 66, 126, 100, 254, 101, 248, 172, 30, 44, 230, 136, 2
13, 6, 128, 0, 225, 69, 5, 69, 86, 130, 13, 64, 1, 173, 0, 114, 138, 4
9, 0, 232, 208, 68, 28, 40, 34, 39, 68, 80, 4, 80, 232, 56, 68, 0, 0, 8
5, 112, 130, 138, 170, 136, 170, 238, 143, 28, 52, 70, 160, 136, 8, 17
0, 32, 162, 168, 229, 0, 0, 1, 71, 40, 0, 117, 188, 190, 251, 43, 238,
111, 164, 36, 92, 47, 26, 216, 147, 79, 42, 195, 94, 75, 179, 159, 107,
96, 216, 209, 56, 247, 181, 218, 122, 15, 167, 80, 114, 173, 227, 231,
185, 15, 36, 250, 151, 206, 15, 12, 250, 62, 131, 219, 49, 121, 109, 17
3, 78, 247, 182, 103, 99, 219, 105, 169, 119, 157, 212, 207, 243, 250,
106, 122, 125, 119, 202, 220, 173, 61, 147, 87, 226, 62, 199, 166, 244,
38, 231, 249, 226, 124, 163, 73, 232, 127, 46, 86, 120, 61, 87, 204, 12
1, 79, 72, 221, 123, 167, 133, 200, 147, 71, 219, 215, 62, 110, 250, 11
9, 11, 245, 103, 213, 125, 125, 87, 127, 69, 218, 117, 108, 14, 108, 14

2, 215, 43, 251, 244, 145, 211, 183, 58, 250, 184, 145, 122, 218, 220,
118, 227, 158, 198, 192, 252, 188, 252, 203, 243, 40, 241, 56, 177, 4,
17, 4, 0, 20, 85, 80, 0, 17, 17, 28, 213, 5, 104, 214, 128, 2, 185, 16
8, 2, 142, 70, 170, 10, 224, 68, 84, 112, 40, 128, 40, 231, 0, 0, 0, 17
0, 170, 2, 185, 21, 68, 81, 122, 60, 4, 70, 130, 13, 0, 112, 128, 229,
21, 84, 0, 0, 58, 53, 84, 0, 23, 165, 143, 188, 125, 119, 246, 183, 17
7, 119, 233, 101, 6, 13, 127, 103, 96, 60, 174, 233, 45, 238, 109, 108,
223, 195, 132, 66, 91, 238, 125, 91, 212, 242, 28, 97, 182, 223, 199, 1
07, 61, 167, 231, 202, 29, 245, 183, 111, 89, 240, 187, 221, 189, 229,
110, 71, 208, 44, 54, 23, 253, 168, 163, 166, 71, 1, 22, 174, 206, 187,
222, 124, 75, 210, 108, 248, 50, 101, 215, 94, 82, 104, 167, 199, 139,
159, 179, 242, 223, 37, 249, 235, 200, 87, 212, 247, 191, 51, 250, 111,
155, 253, 117, 243, 79, 180, 84, 222, 102, 190, 202, 222, 210, 251, 69,
229, 76, 245, 170, 136, 248, 156, 222, 189, 31, 218, 71, 110, 172, 137,
85, 79, 81, 83, 161, 186, 186, 235, 18, 142, 135, 7, 240, 95, 193, 127,
44, 230, 184, 115, 230, 8, 168, 141, 1, 68, 5, 114, 128, 3, 81, 0, 7, 4
2, 181, 141, 69, 4, 5, 16, 0, 30, 48, 1, 202, 212, 1, 226, 42, 160, 40,
57, 192, 0, 0, 10, 163, 145, 92, 224, 4, 80, 232, 245, 28, 214, 32, 21
2, 17, 65, 194, 43, 92, 170, 230, 189, 20, 0, 0, 114, 168, 0, 29, 29, 3
5, 223, 190, 246, 250, 107, 214, 186, 204, 182, 167, 174, 106, 46, 39,
196, 238, 97, 89, 107, 109, 173, 165, 182, 60, 104, 145, 59, 90, 119, 2
45, 189, 52, 56, 105, 6, 223, 35, 105, 97, 227, 253, 117, 118, 254, 20
3, 224, 182, 182, 187, 186, 207, 48, 217, 238, 250, 239, 164, 80, 39, 7
6, 230, 86, 215, 49, 62, 227, 211, 188, 127, 67, 217, 210, 59, 201, 18
0, 119, 78, 210, 96, 192, 175, 184, 165, 225, 133, 248, 219, 217, 252,
182, 195, 193, 61, 87, 3, 244, 255, 0, 205, 191, 90, 248, 151, 182, 23
0, 254, 131, 147, 159, 246, 71, 69, 176, 35, 68, 140, 206, 99, 27, 210,
83, 156, 59, 159, 10, 234, 190, 50, 109, 236, 250, 215, 210, 230, 243,
159, 155, 159, 147, 88, 180, 227, 23, 152, 0, 128, 130, 140, 5, 120, 0,
136, 130, 0, 14, 112, 214, 181, 21, 4, 17, 64, 0, 21, 90, 34, 142, 26,
0, 230, 171, 148, 65, 65, 94, 0, 13, 81, 64, 115, 132, 87, 168, 3, 84,
87, 189, 202, 228, 107, 90, 214, 130, 142, 20, 98, 170, 185, 28, 10, 0,
0, 231, 0, 0, 47, 93, 109, 87, 167, 126, 176, 239, 189, 70, 77, 237, 13
3, 31, 39, 195, 137, 158, 241, 152, 177, 44, 53, 183, 29, 244, 253, 24
8, 242, 88, 156, 36, 246, 218, 198, 245, 168, 89, 43, 44, 198, 162, 92,
62, 117, 222, 121, 236, 58, 173, 15, 153, 197, 217, 109, 252, 67, 95, 1
11, 99, 191, 103, 58, 27, 84, 204, 171, 78, 215, 53, 111, 142, 89, 74,
233, 111, 97, 222, 206, 37, 92, 72, 83, 106, 37, 79, 160, 233, 135, 24
3, 111, 155, 125, 82, 143, 72, 207, 107, 165, 205, 233, 125, 115, 63, 2
11, 210, 185, 119, 232, 188, 97, 112, 103, 56, 253, 206, 163, 209, 122,
39, 38, 69, 103, 59, 25, 221, 170, 51, 212, 184, 143, 199, 31, 152, 25
2, 207, 29, 95, 201, 130, 160, 0, 0, 141, 69, 122, 160, 8, 197, 1, 0, 1
20, 212, 26, 230, 2, 8, 160, 0, 0, 169, 207, 160, 175, 98, 0, 228, 85,
80, 69, 14, 128, 0, 8, 160, 57, 84, 7, 56, 6, 138, 41, 211, 162, 185, 9
0, 141, 99, 80, 71, 128, 2, 143, 21, 5, 1, 94, 196, 28, 224, 0, 3, 173,
207, 213, 186, 111, 183, 53, 186, 237, 118, 185, 196, 90, 216, 85, 21,
62, 53, 91, 26, 203, 75, 161, 186, 178, 158, 141, 232, 156, 184, 78, 21
2, 106, 105, 183, 89, 202, 27, 187, 59, 111, 77, 249, 199, 68, 148, 21
4, 158, 201, 150, 172, 109, 213, 142, 2, 229, 61, 82, 227, 142, 34, 24
5, 181, 240, 109, 163, 201, 166, 145, 105, 81, 58, 210, 79, 91, 107, 7
5, 9, 16, 107, 120, 241, 109, 42, 90, 218, 241, 143, 159, 168, 160, 24
2, 217, 81, 125, 83, 41, 199, 125, 189, 243, 173, 221, 219, 151, 135, 6
8, 229, 93, 199, 137, 42, 75, 156, 43, 90, 206, 28, 185, 49, 108, 167,
54, 174, 30, 55, 192, 62, 40, 252, 241, 240, 90, 254, 40, 0, 0, 10, 14
2, 107, 16, 232, 14, 26, 53, 170, 170, 141, 7, 59, 154, 32, 160, 141, 1
7, 138, 163, 128, 0, 70, 171, 148, 64, 7, 10, 170, 128, 42, 184, 0, 0,
0, 120, 40, 57, 206, 65, 162, 14, 30, 231, 188, 6, 140, 70, 138, 213, 8

1, 1, 94, 160, 0, 57, 205, 104, 231, 0, 0, 29, 167, 239, 255, 0, 73, 12
5, 209, 146, 181, 158, 221, 152, 215, 103, 108, 51, 152, 248, 30, 87, 1
58, 117, 142, 155, 67, 54, 218, 119, 103, 18, 187, 197, 167, 244, 63, 6
6, 207, 82, 75, 204, 109, 56, 103, 125, 139, 67, 225, 62, 145, 129, 21
0, 220, 90, 59, 52, 251, 168, 253, 251, 90, 236, 27, 90, 182, 144, 40,
44, 223, 174, 196, 194, 220, 230, 173, 122, 75, 233, 111, 7, 77, 51, 14
3, 42, 238, 107, 142, 225, 107, 125, 220, 157, 223, 25, 146, 131, 99, 2
11, 45, 119, 99, 166, 205, 95, 219, 217, 85, 61, 157, 35, 215, 180, 23
3, 221, 21, 9, 71, 10, 248, 104, 249, 29, 38, 76, 141, 10, 14, 111, 21,
240, 31, 229, 183, 131, 197, 230, 0, 0, 0, 229, 107, 26, 43, 149, 202,
141, 230, 14, 1, 204, 123, 90, 136, 160, 168, 212, 98, 3, 129, 69, 0, 6
9, 20, 64, 7, 163, 135, 13, 81, 92, 0, 0, 2, 138, 14, 21, 21, 234, 168,
212, 71, 40, 174, 123, 156, 170, 13, 70, 163, 81, 20, 69, 81, 94, 0, 1
0, 229, 107, 78, 130, 162, 162, 136, 47, 121, 254, 129, 244, 215, 223,
210, 162, 203, 250, 34, 178, 197, 110, 241, 222, 85, 199, 27, 133, 149,
43, 77, 126, 249, 154, 14, 210, 78, 166, 127, 35, 232, 26, 47, 68, 175,
204, 249, 221, 87, 173, 218, 187, 234, 95, 48, 241, 221, 69, 199, 73, 1
89, 56, 67, 155, 127, 67, 180, 242, 175, 80, 188, 174, 124, 200, 148, 1
78, 45, 244, 245, 185, 77, 182, 70, 194, 108, 190, 243, 235, 180, 125,
59, 75, 170, 108, 124, 245, 103, 93, 68, 169, 29, 165, 69, 229, 18, 20,
26, 158, 181, 23, 86, 76, 155, 113, 94, 156, 221, 22, 39, 62, 253, 88,
51, 147, 163, 162, 213, 197, 167, 91, 107, 43, 217, 60, 226, 87, 230, 1
88, 179, 240, 139, 230, 76, 175, 6, 128, 0, 0, 57, 200, 198, 32, 61, 9
2, 137, 204, 81, 84, 84, 86, 32, 128, 160, 196, 107, 69, 81, 7, 40, 0,
0, 0, 61, 28, 168, 228, 71, 42, 184, 65, 0, 5, 80, 84, 28, 163, 199, 5
7, 173, 17, 64, 87, 57, 239, 20, 84, 107, 70, 32, 34, 128, 231, 0, 3, 1
49, 88, 60, 80, 120, 32, 189, 102, 253, 7, 247, 55, 184, 232, 179, 117,
254, 225, 166, 239, 125, 162, 242, 159, 43, 143, 143, 243, 123, 126, 15
4, 61, 28, 142, 250, 21, 177, 148, 50, 167, 167, 91, 31, 65, 215, 97, 1
88, 227, 17, 59, 210, 93, 232, 255, 0, 74, 249, 135, 141, 233, 161, 23
4, 178, 219, 175, 59, 235, 233, 52, 90, 143, 43, 221, 237, 169, 166, 5
4, 70, 95, 181, 214, 192, 201, 222, 195, 143, 214, 108, 137, 93, 172, 1
61, 174, 205, 148, 117, 240, 104, 233, 246, 111, 147, 98, 140, 187, 22
6, 149, 249, 203, 60, 187, 172, 96, 93, 76, 233, 7, 179, 226, 243, 237,
209, 156, 250, 170, 161, 156, 162, 129, 210, 54, 162, 194, 202, 106, 7
1, 137, 79, 11, 243, 7, 242, 159, 203, 168, 248, 180, 0, 1, 1, 85, 232,
214, 52, 1, 206, 26, 213, 69, 114, 128, 196, 69, 0, 24, 140, 64, 0, 11
5, 128, 0, 16, 80, 28, 228, 87, 40, 168, 42, 170, 141, 64, 5, 84, 28, 1
68, 60, 85, 85, 65, 17, 65, 69, 115, 156, 175, 20, 70, 180, 107, 64, 6
9, 28, 224, 1, 206, 107, 154, 170, 2, 188, 21, 23, 183, 166, 251, 31, 2
18, 94, 185, 171, 199, 211, 122, 183, 173, 86, 111, 45, 188, 127, 201,
230, 96, 114, 117, 83, 181, 122, 123, 73, 182, 216, 223, 73, 235, 25, 3
3, 166, 134, 163, 67, 183, 155, 159, 204, 224, 108, 125, 15, 151, 185,
237, 252, 62, 76, 238, 143, 192, 117, 129, 236, 248, 93, 71, 156, 236,
181, 201, 214, 46, 183, 202, 175, 234, 189, 150, 186, 166, 84, 200, 14
0, 129, 42, 222, 93, 213, 67, 61, 13, 185, 234, 222, 113, 177, 183, 13
9, 97, 160, 237, 22, 195, 188, 135, 231, 37, 192, 168, 229, 194, 108, 2
15, 70, 232, 169, 211, 161, 197, 123, 18, 10, 210, 37, 87, 77, 20, 195,
135, 14, 92, 235, 41, 243, 95, 138, 95, 159, 89, 184, 140, 96, 10, 128,
131, 92, 167, 68, 99, 90, 0, 40, 142, 84, 20, 81, 26, 128, 0, 49, 26, 2
09, 194, 32, 231, 42, 0, 35, 81, 234, 7, 64, 85, 84, 20, 87, 42, 141, 6
4, 16, 85, 28, 130, 171, 149, 28, 170, 212, 21, 1, 71, 43, 156, 174, 8
4, 4, 104, 214, 128, 10, 225, 69, 122, 40, 213, 80, 1, 238, 112, 163, 2
53, 51, 244, 35, 233, 62, 86, 217, 88, 154, 111, 121, 170, 245, 62, 15
8, 123, 230, 17, 104, 60, 34, 37, 174, 143, 85, 180, 208, 89, 227, 253,
6, 60, 38, 67, 44, 189, 179, 206, 245, 51, 115, 217, 63, 52, 157, 164,
214, 251, 46, 207, 2, 82, 34, 46, 78, 211, 91, 117, 203, 206, 118, 247,

36, 166, 232, 124, 115, 211, 115, 155, 93, 14, 122, 193, 208, 96, 203,
133, 173, 153, 37, 177, 183, 210, 234, 50, 147, 120, 115, 198, 205, 18
7, 180, 178, 233, 26, 202, 69, 181, 111, 30, 121, 238, 80, 121, 74, 14
2, 231, 18, 223, 197, 120, 204, 225, 217, 164, 72, 34, 79, 139, 30, 95,
116, 135, 207, 141, 71, 128, 254, 60, 124, 45, 6, 63, 16, 21, 21, 4, 10
6, 57, 192, 28, 193, 80, 5, 71, 56, 96, 56, 26, 128, 0, 49, 26, 215, 5
6, 230, 162, 185, 200, 128, 53, 170, 170, 41, 208, 20, 21, 192, 57, 92,
52, 6, 185, 5, 1, 65, 85, 92, 168, 32, 162, 42, 56, 28, 57, 84, 114, 16
2, 32, 214, 128, 43, 148, 7, 56, 69, 1, 64, 115, 213, 92, 222, 151, 31,
116, 253, 99, 178, 151, 145, 225, 234, 246, 62, 133, 233, 153, 8, 94, 4
1, 111, 230, 62, 39, 91, 121, 111, 179, 244, 173, 36, 142, 189, 136, 17
7, 98, 67, 109, 175, 211, 30, 47, 214, 203, 175, 207, 180, 219, 15, 70,
181, 244, 168, 85, 221, 171, 236, 35, 227, 116, 189, 61, 53, 158, 123,
 161, 184, 235, 75, 11, 210, 190, 126, 247, 28, 141, 199, 168, 226, 23
6, 241, 246, 240, 175, 34, 237, 36, 117, 94, 189, 237, 59, 81, 74, 141,
105, 229, 22, 26, 37, 157, 99, 38, 206, 68, 206, 145, 160, 102, 251, 8
1, 191, 159, 22, 118, 153, 35, 156, 118, 183, 175, 78, 61, 93, 22, 12,
 40, 82, 221, 107, 53, 144, 35, 242, 139, 15, 225, 95, 201, 255, 0, 19,
199, 209, 71, 96, 42, 0, 136, 215, 57, 80, 107, 67, 163, 16, 1, 206, 7
0, 160, 29, 57, 136, 162, 40, 212, 70, 170, 130, 43, 92, 57, 160, 2, 3,
71, 157, 0, 85, 85, 1, 206, 84, 0, 64, 5, 4, 5, 122, 168, 168, 0, 2, 13
0, 170, 42, 171, 148, 64, 68, 104, 57, 69, 5, 114, 128, 2, 128, 229, 12
2, 142, 239, 235, 95, 83, 125, 147, 59, 142, 106, 127, 174, 192, 250, 1
4, 223, 21, 7, 206, 115, 126, 107, 229, 53, 247, 247, 251, 15, 73, 211,
203, 115, 221, 6, 28, 122, 150, 245, 212, 125, 97, 227, 121, 170, 83, 1
95, 228, 123, 206, 63, 234, 191, 51, 205, 251, 207, 130, 125, 53, 224,
 222, 125, 232, 246, 250, 12, 198, 202, 179, 59, 180, 137, 155, 245, 4
7, 11, 245, 54, 211, 122, 244, 44, 214, 123, 75, 153, 223, 174, 166, 10
3, 71, 246, 135, 34, 84, 11, 154, 187, 15, 56, 216, 197, 89, 214, 93, 2
38, 58, 75, 235, 194, 53, 92, 106, 62, 124, 35, 132, 233, 80, 152, 245,
115, 151, 141, 53, 62, 94, 167, 173, 190, 147, 71, 97, 18, 190, 57, 95,
85, 249, 249, 248, 159, 141, 108, 94, 76, 68, 84, 6, 168, 199, 56, 86,
 177, 71, 8, 208, 20, 81, 160, 10, 130, 10, 2, 35, 64, 0, 1, 68, 0, 1,
 136, 175, 232, 168, 169, 208, 0, 115, 129, 68, 4, 81, 20, 64, 85, 87,
 40, 173, 5, 84, 85, 68, 80, 85, 7, 168, 2, 34, 3, 148, 21, 202, 2, 40,
40, 2, 185, 92, 189, 165, 250, 191, 213, 127, 87, 217, 214, 64, 216, 11
1, 166, 253, 17, 158, 193, 102, 232, 115, 158, 15, 146, 125, 165, 246,
 219, 81, 232, 150, 157, 58, 244, 164, 175, 227, 93, 207, 103, 163, 15
3, 234, 120, 26, 185, 255, 0, 54, 196, 212, 250, 238, 135, 71, 81, 107,
91, 79, 173, 242, 203, 157, 102, 139, 5, 233, 149, 30, 75, 236, 59, 63,
12, 213, 90, 182, 201, 219, 127, 61, 205, 108, 114, 26, 235, 139, 251,
 27, 27, 30, 188, 236, 33, 241, 229, 117, 92, 153, 253, 173, 43, 86, 23
2, 180, 127, 75, 24, 172, 134, 234, 40, 145, 234, 219, 34, 82, 177, 19
9, 35, 131, 115, 173, 204, 240, 144, 92, 233, 175, 57, 212, 211, 55, 16
3, 251, 126, 30, 126, 98, 215, 241, 107, 81, 163, 65, 0, 98, 184, 26, 2
09, 234, 28, 192, 115, 185, 138, 128, 0, 138, 0, 53, 160, 2, 160, 40, 1
28, 0, 12, 30, 116, 85, 21, 64, 5, 87, 130, 160, 130, 40, 138, 34, 185,
21, 84, 112, 32, 168, 57, 20, 7, 42, 128, 162, 34, 56, 80, 7, 56, 0, 2
1, 5, 120, 136, 170, 189, 39, 236, 244, 27, 15, 160, 254, 142, 214, 20
9, 84, 123, 84, 255, 0, 75, 245, 159, 54, 240, 41, 18, 188, 87, 206, 16
2, 72, 237, 117, 232, 87, 219, 125, 20, 217, 29, 233, 170, 97, 67, 141,
235, 114, 124, 219, 232, 91, 204, 62, 90, 63, 146, 75, 247, 127, 21, 25
0, 42, 239, 202, 246, 94, 119, 164, 235, 90, 158, 137, 85, 117, 182, 24
9, 143, 222, 221, 228, 26, 75, 202, 249, 91, 10, 93, 231, 148, 173, 16
4, 171, 239, 71, 207, 118, 210, 223, 182, 77, 79, 106, 29, 79, 24, 176,
175, 115, 140, 237, 121, 221, 150, 189, 237, 97, 156, 163, 193, 175, 13
1, 18, 58, 58, 84, 135, 68, 134, 206, 145, 140, 243, 169, 91, 39, 83, 1

64, 117, 70, 98, 190, 84, 158, 180, 127, 138, 63, 0, 96, 185, 181, 6, 1
81, 4, 68, 6, 170, 136, 141, 85, 112, 196, 5, 123, 16, 0, 0, 0, 1, 173,
0, 28, 136, 168, 2, 42, 10, 35, 30, 167, 78, 141, 112, 0, 10, 229, 28,
 163, 90, 34, 130, 170, 43, 144, 85, 21, 205, 21, 174, 69, 112, 34, 18
5, 192, 8, 212, 81, 202, 0, 175, 5, 4, 0, 115, 156, 208, 7, 246, 149, 1
19, 244, 87, 216, 94, 191, 11, 61, 89, 244, 22, 143, 209, 55, 222, 83,
 242, 86, 211, 47, 225, 161, 214, 71, 95, 78, 214, 108, 45, 174, 231, 1
49, 53, 85, 116, 206, 244, 143, 68, 242, 61, 182, 246, 4, 122, 28, 166,
83, 103, 15, 211, 119, 30, 97, 103, 151, 180, 216, 121, 14, 203, 67, 14
2, 218, 250, 47, 207, 126, 177, 186, 243, 44, 180, 191, 78, 202, 202, 1
37, 233, 30, 89, 158, 245, 207, 5, 245, 207, 87, 242, 61, 159, 165, 10
4, 68, 135, 13, 250, 56, 245, 203, 156, 225, 99, 222, 229, 57, 93, 216,
75, 116, 104, 252, 226, 113, 132, 149, 124, 122, 244, 239, 28, 129, 21
4, 44, 42, 142, 212, 252, 164, 232, 174, 229, 85, 231, 234, 162, 119, 1
53, 218, 15, 243, 225, 241, 157, 87, 52, 104, 49, 17, 17, 1, 20, 26, 13
1, 249, 185, 90, 32, 42, 0, 42, 0, 0, 2, 48, 0, 7, 176, 84, 70, 160, 17
4, 81, 137, 208, 94, 142, 80, 0, 1, 238, 20, 86, 53, 200, 160, 10, 170,
0, 170, 225, 168, 168, 241, 65, 175, 114, 128, 141, 64, 87, 40, 3, 213,
84, 115, 16, 21, 92, 174, 69, 64, 87, 117, 176, 244, 175, 182, 190, 16
1, 169, 198, 94, 125, 47, 162, 190, 179, 242, 239, 1, 135, 226, 20, 22
1, 186, 147, 87, 210, 182, 58, 221, 77, 157, 164, 72, 117, 148, 57, 18
4, 158, 165, 235, 94, 109, 179, 147, 158, 215, 192, 177, 249, 210, 87,
 176, 78, 245, 47, 55, 205, 108, 49, 123, 142, 181, 59, 47, 46, 157, 23
6, 120, 74, 79, 104, 205, 249, 46, 99, 233, 31, 57, 216, 102, 253, 111,
231, 28, 199, 211, 63, 58, 125, 53, 111, 230, 54, 27, 207, 89, 173, 15
5, 91, 85, 59, 79, 35, 49, 119, 135, 141, 127, 10, 76, 163, 89, 38, 10
4, 218, 183, 199, 226, 113, 139, 7, 156, 123, 20, 165, 159, 26, 4, 138,
90, 218, 119, 91, 105, 236, 224, 212, 83, 82, 199, 147, 103, 101, 242,
 183, 226, 191, 202, 241, 24, 48, 6, 53, 162, 0, 3, 90, 231, 48, 112, 1
41, 0, 0, 0, 5, 64, 69, 70, 130, 0, 61, 17, 4, 96, 29, 1, 20, 85, 234,
 40, 0, 2, 185, 92, 162, 34, 32, 160, 225, 64, 80, 85, 4, 17, 194, 56,
 69, 94, 138, 173, 26, 130, 42, 170, 128, 175, 87, 40, 35, 80, 87, 43,
 156, 0, 42, 246, 184, 247, 255, 0, 185, 125, 4, 198, 109, 190, 167, 15
5, 103, 91, 134, 161, 249, 127, 205, 105, 59, 200, 235, 54, 215, 113, 1
82, 212, 106, 236, 244, 153, 22, 72, 164, 243, 186, 143, 93, 208, 251,
 239, 135, 213, 125, 41, 131, 148, 96, 48, 126, 149, 207, 91, 233, 62,
 17, 121, 27, 209, 161, 124, 185, 246, 143, 137, 73, 218, 175, 159, 23
4, 61, 195, 194, 252, 83, 220, 61, 95, 207, 170, 182, 63, 57, 81, 125,
 33, 231, 62, 203, 63, 79, 73, 121, 177, 139, 218, 235, 21, 167, 210, 1
52, 77, 63, 28, 247, 94, 55, 85, 178, 180, 118, 54, 81, 227, 211, 91, 5
8, 36, 154, 206, 85, 209, 146, 87, 30, 35, 250, 209, 211, 212, 206, 13
7, 166, 188, 135, 89, 159, 205, 175, 73, 218, 27, 47, 130, 255, 0, 24,
 62, 127, 226, 198, 177, 205, 24, 214, 10, 32, 13, 71, 43, 16, 87, 12,
 0, 0, 0, 21, 4, 21, 4, 17, 0, 28, 136, 3, 65, 84, 0, 85, 236, 213, 80,
0, 28, 170, 162, 138, 141, 69, 21, 92, 128, 160, 10, 42, 42, 10, 131, 1
33, 115, 156, 52, 104, 213, 115, 92, 162, 170, 171, 156, 162, 8, 193, 9
4, 61, 64, 80, 124, 221, 127, 164, 125, 107, 244, 63, 92, 246, 143, 23
3, 125, 191, 12, 156, 63, 18, 242, 15, 51, 243, 235, 153, 110, 157, 12
5, 232, 58, 237, 117, 190, 158, 203, 41, 194, 238, 143, 202, 89, 115, 2
38, 187, 172, 238, 95, 214, 115, 217, 143, 84, 149, 243, 79, 182, 248,
 151, 210, 26, 143, 29, 219, 249, 159, 171, 89, 99, 25, 232, 255, 0, 5
8, 110, 174, 234, 119, 250, 92, 223, 207, 190, 253, 57, 248, 191, 56, 1
76, 193, 251, 127, 138, 253, 73, 95, 233, 180, 151, 157, 229, 215, 234,
96, 228, 253, 22, 247, 205, 244, 47, 207, 113, 235, 161, 161, 93, 21, 1
65, 148, 120, 213, 151, 125, 59, 182, 20, 88, 48, 237, 57, 192, 56, 16
5, 69, 86, 46, 230, 191, 208, 231, 194, 175, 171, 197, 86, 150, 250, 13
9, 199, 126, 105, 124, 13, 242, 143, 159, 181, 168, 209, 57, 140, 71, 5

2, 106, 181, 28, 170, 214, 131, 154, 0, 0, 0, 0, 34, 40, 32, 136, 2, 13
8, 136, 0, 2, 160, 1, 208, 114, 138, 0, 43, 148, 112, 160, 196, 85, 21,
64, 80, 5, 16, 20, 84, 85, 28, 241, 85, 162, 42, 40, 1, 209, 20, 232, 2
24, 68, 24, 170, 174, 112, 10, 131, 186, 78, 221, 253, 43, 245, 167, 16
2, 198, 206, 123, 229, 231, 161, 193, 243, 219, 188, 47, 135, 120, 247,
148, 222, 118, 101, 198, 203, 113, 181, 211, 79, 209, 89, 214, 87, 224,
235, 38, 100, 165, 104, 190, 185, 205, 230, 116, 58, 220, 236, 59, 60,
 110, 155, 231, 95, 167, 119, 254, 105, 162, 192, 250, 246, 94, 247, 20
2, 253, 251, 194, 180, 220, 124, 207, 216, 122, 123, 23, 200, 255, 0, 7
7, 231, 183, 30, 17, 71, 195, 194, 126, 198, 249, 139, 212, 61, 235, 20
8, 208, 75, 140, 117, 205, 237, 53, 207, 122, 244, 133, 67, 222, 167, 9
9, 79, 115, 161, 177, 231, 203, 148, 93, 5, 133, 149, 116, 104, 177, 21
9, 197, 171, 73, 202, 36, 92, 117, 173, 222, 186, 158, 174, 158, 131, 5
1, 64, 182, 155, 59, 185, 87, 159, 38, 126, 61, 118, 127, 138, 47, 44,
 38, 51, 59, 201, 88, 32, 214, 0, 174, 68, 64, 0, 0, 0, 0, 1, 21, 21, 1
68, 32, 0, 61, 128, 2, 160, 0, 7, 64, 87, 10, 0, 170, 170, 175, 1, 136,
160, 130, 170, 138, 34, 128, 40, 10, 56, 232, 116, 104, 208, 0, 1, 92,
 163, 250, 40, 28, 192, 21, 85, 71, 8, 143, 237, 113, 238, 127, 64, 12
5, 121, 109, 67, 99, 237, 82, 117, 206, 167, 168, 197, 248, 137, 242, 2
2, 190, 184, 182, 245, 75, 109, 254, 162, 231, 189, 204, 13, 7, 207, 11
7, 62, 241, 243, 60, 143, 113, 62, 155, 196, 249, 55, 189, 192, 201, 22
0, 214, 121, 238, 211, 51, 236, 119, 57, 209, 32, 122, 135, 140, 108, 2
44, 62, 105, 174, 241, 173, 116, 191, 94, 143, 95, 232, 54, 126, 93, 22
8, 213, 24, 207, 116, 240, 157, 159, 212, 90, 125, 69, 205, 54, 175, 3,
97, 53, 247, 21, 217, 206, 239, 198, 94, 229, 61, 51, 1, 187, 189, 176,
227, 206, 4, 251, 155, 126, 209, 33, 193, 143, 223, 155, 51, 81, 234, 1
10, 234, 57, 91, 236, 242, 117, 181, 149, 89, 252, 213, 110, 134, 214,
 206, 198, 202, 215, 202, 127, 28, 60, 187, 196, 45, 58, 81, 93, 235, 2
36, 93, 23, 191, 156, 124, 247, 78, 196, 84, 65, 65, 0, 0, 0, 0, 0, 16,
71, 32, 173, 26, 0, 0, 0, 40, 128, 2, 117, 0, 87, 40, 2, 185, 92, 231,
 52, 70, 171, 81, 69, 85, 81, 20, 0, 81, 21, 92, 43, 206, 141, 68, 0, 2
0, 17, 94, 47, 71, 130, 35, 64, 81, 224, 174, 84, 23, 173, 183, 165, 12
5, 117, 244, 174, 190, 135, 69, 236, 150, 18, 236, 162, 121, 77, 135, 1
37, 218, 252, 127, 170, 194, 73, 159, 172, 250, 3, 166, 134, 250, 76, 2
00, 144, 189, 11, 192, 117, 21, 94, 125, 234, 126, 33, 246, 238, 215, 2
27, 95, 111, 244, 122, 44, 6, 251, 199, 189, 91, 201, 61, 186, 234, 11
8, 103, 203, 61, 207, 205, 125, 183, 230, 175, 164, 49, 253, 106, 51, 6
2, 149, 199, 109, 203, 209, 252, 203, 71, 243, 13, 101, 220, 140, 247,
 63, 164, 244, 91, 171, 109, 79, 26, 164, 171, 182, 211, 225, 219, 105,
79, 139, 182, 171, 180, 206, 111, 54, 11, 218, 63, 13, 34, 216, 203, 13
5, 89, 94, 201, 37, 61, 51, 102, 77, 74, 183, 213, 66, 175, 129, 198, 1
53, 207, 169, 231, 195, 71, 171, 149, 228, 223, 155, 127, 7, 252, 199,
 22, 78, 142, 5, 76, 46, 125, 36, 250, 87, 184, 201, 196, 252, 227, 86,
140, 0, 5, 64, 20, 64, 0, 4, 5, 64, 84, 84, 70, 138, 128, 0, 0, 2, 10,
 3, 212, 1, 234, 0, 175, 85, 114, 191, 154, 13, 64, 28, 224, 65, 64, 7,
32, 168, 231, 14, 81, 205, 0, 28, 162, 177, 224, 175, 232, 12, 64, 30,
 142, 5, 120, 224, 87, 207, 189, 245, 63, 208, 159, 90, 173, 233, 239,
 118, 148, 18, 165, 249, 189, 199, 137, 115, 242, 111, 62, 201, 93, 11
4, 210, 250, 63, 177, 118, 178, 213, 215, 222, 227, 189, 70, 7, 151, 12
3, 71, 201, 254, 133, 154, 219, 125, 11, 224, 90, 13, 182, 122, 214, 18
7, 1, 244, 63, 133, 122, 196, 29, 46, 93, 150, 48, 55, 89, 10, 217, 48,
189, 26, 234, 151, 63, 232, 26, 111, 69, 160, 194, 84, 249, 77, 85, 38,
203, 9, 236, 219, 255, 0, 69, 159, 232, 113, 12, 4, 251, 61, 110, 102,
 142, 254, 138, 158, 118, 6, 226, 94, 155, 72, 182, 112, 102, 90, 117,
 180, 153, 73, 65, 203, 164, 164, 133, 155, 151, 61, 82, 182, 187, 140,
56, 21, 89, 105, 244, 49, 122, 114, 129, 233, 58, 217, 63, 51, 124, 31,
241, 47, 200, 76, 107, 223, 38, 12, 86, 160, 118, 215, 253, 23, 211, 20

2, 188, 67, 136, 40, 0, 29, 17, 160, 128, 2, 34, 162, 180, 112, 43, 81,
21, 0, 0, 0, 6, 163, 148, 58, 0, 43, 197, 85, 5, 114, 168, 3, 81, 26, 2
25, 234, 2, 40, 42, 40, 10, 40, 161, 208, 84, 0, 58, 2, 10, 15, 122, 16
2, 34, 42, 188, 1, 234, 116, 233, 34, 254, 234, 71, 169, 125, 79, 245,
87, 88, 19, 61, 79, 85, 6, 54, 126, 179, 59, 148, 193, 231, 188, 130,
178, 182, 94, 159, 212, 253, 210, 211, 35, 171, 153, 166, 206, 80, 12
5, 31, 75, 230, 209, 29, 71, 203, 232, 28, 70, 42, 234, 118, 223, 202,
43, 61, 95, 15, 35, 127, 129, 184, 178, 93, 87, 144, 251, 103, 200, 9
5, 89, 198, 141, 123, 63, 43, 131, 250, 39, 85, 234, 57, 204, 14, 19, 3
5, 229, 62, 143, 228, 95, 64, 231, 189, 219, 209, 55, 110, 175, 198, 8
7, 122, 28, 14, 48, 239, 42, 112, 12, 151, 142, 222, 89, 78, 180, 159,
206, 217, 247, 243, 122, 229, 234, 35, 88, 72, 227, 6, 162, 81, 223, 1
33, 12, 30, 113, 107, 98, 104, 124, 126, 61, 181, 89, 174, 244, 119, 12
0, 39, 207, 126, 35, 249, 15, 231, 236, 104, 139, 207, 146, 13, 24, 14
3, 149, 235, 254, 227, 149, 249, 154, 144, 86, 170, 131, 156, 128, 34,
32, 34, 181, 70, 163, 145, 200, 136, 42, 0, 0, 0, 2, 40, 7, 85, 85, 0,
114, 142, 7, 128, 3, 81, 1, 206, 5, 64, 80, 80, 26, 174, 84, 80, 115, 1
28, 0, 122, 136, 142, 7, 168, 34, 35, 156, 224, 85, 114, 143, 237, 46,
87, 164, 253, 1, 244, 15, 214, 123, 140, 54, 35, 69, 235, 69, 102, 20
7, 43, 151, 242, 206, 24, 106, 207, 34, 49, 54, 186, 159, 87, 223, 238,
164, 224, 253, 38, 219, 77, 226, 94, 167, 234, 190, 3, 234, 24, 7, 99,
61, 102, 211, 207, 250, 219, 85, 236, 188, 199, 214, 115, 89, 31, 108,
242, 185, 155, 220, 238, 235, 206, 104, 111, 44, 116, 81, 57, 207, 239,
146, 188, 245, 63, 89, 177, 165, 240, 106, 252, 189, 29, 62, 250, 135,
214, 62, 142, 214, 246, 171, 202, 38, 247, 48, 219, 75, 92, 22, 79, 9
3, 65, 136, 245, 62, 83, 123, 219, 77, 233, 47, 71, 37, 249, 170, 37, 1
57, 194, 35, 57, 56, 231, 85, 159, 129, 194, 254, 195, 53, 139, 184, 20
7, 216, 193, 149, 185, 149, 131, 249, 75, 17, 249, 21, 139, 242, 126, 1
04, 141, 78, 109, 17, 26, 192, 94, 142, 188, 250, 7, 167, 134, 226, 68,
5, 122, 128, 28, 192, 26, 143, 68, 64, 86, 128, 0, 0, 0, 42, 2, 40, 29,
92, 160, 163, 148, 21, 69, 80, 4, 68, 21, 21, 194, 160, 10, 42, 162, 16
0, 142, 5, 17, 122, 0, 2, 185, 65, 20, 7, 8, 128, 174, 122, 168, 61, 8
5, 253, 109, 47, 61, 7, 213, 254, 215, 250, 159, 75, 229, 126, 117, 99,
237, 86, 216, 132, 211, 241, 241, 122, 159, 51, 60, 119, 142, 90, 101,
143, 170, 110, 55, 123, 220, 124, 31, 75, 188, 175, 174, 245, 92, 52,
92, 142, 203, 196, 253, 167, 97, 227, 27, 27, 234, 221, 207, 142, 89,
122, 63, 205, 63, 77, 249, 76, 187, 221, 2, 179, 43, 236, 254, 85, 89,
233, 152, 15, 83, 229, 227, 254, 201, 215, 212, 173, 39, 211, 124, 225,
207, 57, 135, 221, 72, 244, 191, 169, 45, 249, 71, 175, 93, 45, 13, 12
5, 181, 231, 150, 107, 177, 151, 244, 22, 54, 178, 99, 221, 234, 115, 1
9, 117, 86, 41, 83, 159, 108, 9, 12, 127, 7, 182, 12, 124, 142, 62, 93,
140, 156, 66, 96, 245, 186, 123, 94, 210, 27, 135, 243, 63, 148, 126, 6
6, 249, 235, 230, 134, 53, 26, 214, 160, 214, 163, 80, 30, 168, 146, 25
3, 154, 255, 0, 201, 176, 162, 185, 192, 0, 196, 1, 160, 130, 0, 0, 0,
14, 104, 0, 0, 49, 202, 72, 1, 69, 112, 138, 61, 85, 4, 21, 21, 20, 0,
80, 65, 200, 170, 3, 69, 81, 64, 122, 128, 10, 240, 69, 0, 81, 3, 163,
92, 175, 1, 234, 231, 207, 183, 222, 123, 135, 175, 253, 183, 238, 21
3, 222, 41, 143, 153, 234, 187, 200, 120, 174, 181, 125, 252, 71, 49, 2
31, 185, 76, 204, 234, 91, 13, 7, 164, 122, 231, 183, 100, 114, 122, 18
9, 38, 171, 9, 112, 249, 84, 20, 189, 171, 125, 39, 183, 147, 250, 30,
223, 13, 89, 221, 29, 53, 34, 81, 250, 78, 135, 204, 189, 7, 206, 253,
18, 111, 159, 250, 12, 123, 202, 118, 239, 246, 245, 154, 43, 239, 20,
249, 243, 105, 224, 9, 232, 223, 72, 253, 7, 58, 174, 251, 43, 214, 21
0, 85, 53, 221, 207, 155, 94, 64, 203, 89, 46, 162, 60, 219, 59, 202, 2
4, 155, 59, 114, 146, 158, 153, 29, 61, 236, 226, 176, 41, 177, 110, 23
1, 198, 206, 31, 143, 201, 244, 133, 172, 144, 182, 115, 114, 159, 25,
124, 165, 249, 143, 143, 107, 81, 168, 214, 3, 6, 180, 87, 128, 215, 1

17, 245, 157, 23, 151, 99, 58, 57, 81, 160, 115, 81, 6, 160, 128, 2, 4
0, 0, 10, 163, 64, 0, 6, 170, 175, 112, 7, 14, 20, 30, 10, 141, 28, 0,
 0, 0, 14, 84, 85, 17, 174, 71, 0, 42, 184, 20, 69, 120, 0, 0, 0, 57, 2
06, 112, 15, 87, 59, 172, 187, 47, 160, 191, 67, 254, 130, 211, 65, 24
2, 188, 195, 253, 23, 209, 238, 168, 170, 114, 75, 230, 88, 143, 22, 12
9, 119, 139, 204, 76, 216, 123, 151, 175, 179, 211, 43, 36, 230, 61, 1
9, 51, 18, 242, 210, 150, 36, 78, 111, 211, 197, 243, 237, 239, 161, 6
5, 194, 224, 189, 37, 38, 192, 205, 103, 125, 63, 182, 190, 118, 126, 1
43, 208, 104, 227, 251, 30, 12, 220, 223, 91, 86, 105, 233, 124, 71, 20
4, 233, 48, 190, 239, 237, 222, 153, 186, 135, 115, 230, 183, 86, 75, 2
18, 223, 158, 59, 69, 27, 202, 53, 246, 87, 79, 158, 151, 48, 97, 106,
 174, 29, 156, 162, 226, 76, 147, 201, 26, 218, 106, 136, 45, 225, 156,
190, 243, 247, 217, 190, 166, 60, 137, 87, 54, 63, 54, 227, 127, 63, 25
4, 53, 249, 229, 136, 214, 162, 49, 4, 104, 209, 84, 4, 80, 127, 160, 2
22, 249, 245, 34, 181, 136, 3, 16, 24, 128, 0, 0, 0, 14, 17, 0, 0, 6, 1
70, 187, 176, 10, 157, 17, 202, 215, 40, 170, 136, 160, 10, 42, 32, 35,
144, 87, 40, 168, 141, 112, 163, 209, 28, 162, 189, 17, 5, 112, 0, 0, 1
68, 29, 7, 57, 85, 85, 207, 124, 139, 79, 77, 251, 99, 237, 127, 102, 2
31, 147, 242, 140, 140, 175, 77, 244, 102, 208, 209, 83, 102, 35, 87, 2
52, 191, 145, 244, 207, 155, 122, 245, 185, 246, 159, 179, 188, 55, 21
7, 124, 251, 223, 243, 241, 38, 99, 171, 61, 155, 23, 123, 65, 46, 174,
175, 93, 81, 103, 7, 214, 60, 126, 124, 91, 142, 125, 177, 16, 116, 25
1, 248, 51, 55, 251, 44, 103, 182, 124, 185, 233, 147, 239, 117, 178, 2
48, 70, 145, 165, 249, 235, 13, 152, 198, 125, 55, 232, 155, 93, 124, 2
7, 127, 41, 235, 127, 207, 65, 117, 154, 235, 33, 222, 61, 177, 237, 18
5, 173, 191, 229, 99, 31, 157, 237, 146, 80, 103, 213, 243, 78, 47, 13
9, 23, 141, 66, 165, 36, 137, 93, 60, 155, 171, 57, 186, 201, 211, 39,
 246, 249, 155, 243, 131, 225, 111, 11, 98, 53, 168, 28, 198, 162, 8, 4
0, 34, 170, 2, 116, 244, 27, 47, 59, 128, 192, 57, 130, 34, 32, 0, 0, 3
4, 128, 162, 162, 0, 0, 8, 162, 247, 1, 71, 163, 212, 69, 20, 0, 1, 85,
81, 16, 20, 71, 40, 160, 136, 224, 30, 10, 43, 208, 84, 80, 1, 80, 0,
 1, 92, 189, 7, 170, 191, 188, 185, 55, 27, 143, 161, 254, 157, 250, 3,
212, 169, 124, 219, 21, 35, 101, 235, 178, 170, 51, 89, 252, 245, 10, 1
24, 129, 169, 194, 227, 235, 153, 127, 105, 247, 202, 231, 125, 140, 12
7, 40, 180, 17, 229, 220, 230, 247, 250, 156, 196, 239, 41, 215, 165, 5
4, 214, 126, 1, 254, 185, 231, 119, 108, 243, 235, 25, 9, 232, 154, 61,
183, 180, 120, 215, 160, 216, 211, 192, 244, 93, 7, 26, 72, 221, 188, 1
99, 21, 219, 198, 253, 43, 232, 139, 189, 45, 44, 156, 28, 107, 134, 23
6, 45, 114, 114, 167, 72, 243, 93, 111, 125, 84, 57, 182, 61, 184, 241,
180, 152, 204, 196, 7, 247, 115, 229, 68, 170, 226, 206, 85, 188, 230,
 247, 161, 243, 202, 110, 144, 18, 213, 210, 118, 87, 89, 188, 151, 13
8, 255, 0, 62, 190, 40, 214, 181, 173, 65, 168, 212, 65, 17, 205, 85,
 0, 24, 143, 237, 38, 11, 69, 106, 3, 68, 16, 69, 0, 0, 20, 1, 0, 1, 20
0, 128, 4, 128, 21, 81, 238, 85, 0, 0, 0, 114, 14, 104, 130, 130, 170,
 138, 130, 10, 163, 148, 7, 56, 4, 5, 20, 69, 5, 28, 192, 87, 43, 149,
 206, 87, 118, 145, 222, 203, 232, 79, 188, 190, 158, 209, 166, 103, 7,
149, 155, 170, 219, 95, 179, 61, 129, 172, 165, 168, 249, 191, 101, 22
5, 245, 41, 27, 105, 233, 223, 81, 64, 191, 221, 210, 187, 51, 123, 30,
178, 83, 238, 232, 13, 244, 61, 142, 62, 157, 217, 45, 165, 28, 248, 25
4, 181, 70, 243, 49, 222, 223, 135, 166, 205, 247, 29, 4, 118, 68, 125,
133, 15, 173, 211, 88, 97, 39, 248, 230, 98, 127, 111, 160, 169, 111, 1
05, 187, 231, 170, 150, 202, 195, 111, 89, 135, 215, 90, 208, 192, 184,
181, 239, 101, 39, 179, 161, 218, 116, 118, 122, 169, 58, 204, 147, 20
2, 170, 3, 85, 88, 210, 147, 23, 83, 38, 182, 254, 6, 90, 54, 162, 254,
237, 48, 95, 49, 254, 118, 124, 139, 224, 168, 198, 13, 70, 141, 68, 4,
84, 80, 0, 107, 78, 138, 198, 128, 12, 80, 68, 0, 0, 0, 20, 64, 0, 0,
 0, 9, 0, 42, 170, 171, 209, 64, 0, 21, 5, 1, 205, 5, 86, 171, 129, 65,

```
7, 160, 170, 10, 240, 0, 1, 80, 80, 114, 130, 2, 171, 149, 238, 115, 18
6, 117, 237, 123, 247, 111, 223, 94, 197, 210, 143, 55, 138, 199, 118,
 213, 107, 237, 226, 242, 243, 152, 220, 114, 126, 115, 199, 230, 39, 2
36, 231, 123, 167, 178, 76, 210, 101, 118, 149, 47, 175, 108, 218, 185,
214, 174, 235, 79, 73, 164, 209, 95, 103, 146, 190, 162, 194, 85, 12, 2
51, 91, 57, 146, 165, 81, 250, 254, 136, 117, 238, 190, 37, 94, 214, 24
7, 63, 85, 232, 58, 126, 89, 204, 231, 206, 94, 195, 168, 176, 233, 85,
99, 71, 146, 169, 95, 68, 227, 175, 145, 129, 176, 177, 175, 145, 46, 1
4, 246, 103, 59, 46, 60, 229, 119, 43, 50, 45, 180, 155, 219, 141, 124,
86, 241, 108, 126, 212, 88, 171, 134, 199, 172, 242, 188, 52, 77, 30, 1
79, 117, 189, 159, 130, 249, 219, 240, 111, 107, 235, 63, 30, 121, 19,
 90, 131, 90, 209, 160, 0, 0, 8, 212, 232, 49, 1, 21, 21, 170, 136, 32,
0, 168, 0, 10, 138, 128, 0, 0, 1, 221, 84, 28, 40, 231, 0, 0, 10, 138,
 138, 56, 65, 64, 5, 28, 168, 138, 160, 160, 57, 192, 2, 40, 40, 128, 1
75, 0, 64, 115, 151, 163, 221, 209, 253, 187, 236, 126, 167, 251, 95, 2
13, 53, 29, 105, 178, 185, 142, 155, 13, 87, 122, 200, 85, 121, 212, 20
5, 100, 232, 60, 90, 219, 209, 52, 214, 222, 179, 5, 210, 169, 61, 34,
 153, 148, 252, 246, 25, 73, 122, 168, 204, 50, 26, 221, 52, 124, 198,
 227, 59, 140, 210, 218, 99, 159, 26, 231, 95, 125, 232, 26, 215, 85, 2
16, 179, 62, 106, 50, 135, 173, 196, 245, 155, 18, 23, 79, 36, 216, 11
0, 113, 150, 89, 91, 104, 158, 78, 181, 30, 167, 211, 127, 194, 137, 14
3, 170, 180, 156, 235, 253, 19, 103, 113, 26, 233, 181, 56, 27, 11, 41,
189, 160, 241, 225, 197, 188, 251, 85, 229, 53, 20, 206, 242, 111, 43,
 215, 102, 114, 52, 222, 129, 233, 94, 183, 50, 183, 243, 175, 242, 25
1, 228, 62, 223, 88, 244, 248, 202, 59, 70, 35, 68, 16, 0, 0, 84, 68, 1
12, 214, 128, 0, 53, 200, 32, 136, 0, 10, 168, 128, 2, 160, 0, 3, 147,
 163, 149, 85, 194, 170, 168, 0, 2, 162, 136, 162, 184, 17, 80, 71, 10,
174, 106, 168, 0, 43, 193, 5, 17, 64, 81, 7, 56, 115, 68, 69, 115, 156,
247, 244, 235, 101, 58, 239, 222, 62, 143, 247, 71, 109, 52, 236, 165,
 167, 145, 171, 184, 43, 33, 116, 160, 133, 23, 21, 93, 188, 141, 101,
 164, 175, 178, 141, 75, 161, 169, 208, 44, 222, 149, 48, 61, 7, 31, 10
7, 107, 9, 38, 101, 111, 175, 32, 69, 155, 69, 77, 59, 189, 244, 199, 9
0, 110, 33, 58, 179, 219, 188, 162, 175, 21, 89, 65, 7, 3]
```

In [ ]:
```
f = open('my_file.mp3', 'w+b')
file_content = f.read()
f.write(b'Hello')
f.close()
```

In [ ]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
---------------------------------------------------------------------------
MessageError                                Traceback (most recent call last)
<ipython-input-4-21dc3c638f66> in <module>()
      1 from google.colab import files
----> 2 uploaded = files.upload()

/usr/local/lib/python3.7/dist-packages/google/colab/files.py in upload()
     44   """
     45
---> 46   uploaded_files = _upload_files(multiple=True)
     47   # Mapping from original filename to filename as saved locally.
     48   local_filenames = dict()

/usr/local/lib/python3.7/dist-packages/google/colab/files.py in _upload_files(multiple)
    121   result = _output.eval_js(
    122       'google.colab._files._uploadFiles("{input_id}", "{output_id}")'.format(
--> 123           input_id=input_id, output_id=output_id))
    124   files = _collections.defaultdict(_six.binary_type)
    125

/usr/local/lib/python3.7/dist-packages/google/colab/output/_js.py in eval_js(script, ignore_result, timeout_sec)
     38   if ignore_result:
     39     return
---> 40   return _message.read_reply_from_input(request_id, timeout_sec)
     41
     42

/usr/local/lib/python3.7/dist-packages/google/colab/_message.py in read_reply_from_input(message_id, timeout_sec)
    104         reply.get('colab_msg_id') == message_id):
    105       if 'error' in reply:
--> 106         raise MessageError(reply['error'])
    107       return reply.get('data', None)
    108

MessageError: TypeError: google.colab._files is undefined
```

In [ ]:
```
#The following program will copy oldpic.jpeg into newpic.jpeg
f1=open("/content/pexels-pixabay-45201.jpg", "rb")
f2=open("newpirc.jpeg", "wb")
bytesdata=f1.read()
f2.write(bytesdata)
print("New Image is available with the name: newpic.jpg")
```

```
New Image is available with the name: newpic.jpg
```

In [ ]:
```
#word search in python
cnt = 0
word_search = input("Enter the words to search:")
with open("book.txt","r") as f1:
  data=f1.read()
  words = data.split()
  for word in words:
    if (word == word_search):
      cnt+=1
print(word_search, "found ", cnt, " times from the file")
```

In [ ]:
```
#creating a class
class first:
  pass
obj1=first()
print(obj1)
```

```
<__main__.first object at 0x7f3e8a73fad0>
```

In [ ]:
```
class add1:
  def addition(self,a,b):
    self.a=a
    self.b=b
    self.sum=self.a+self.b
    print("afterr addition",self.sum)
obj1=add1()
obj1.addition(2,4)
```

```
afterr addition 6
```

In [ ]:
```
#simple class
class computer:
    def features(self):
       print("this is a new model system")
comp1=computer()
comp1.features()
computer.features(comp1)
```

```
this is a new model system
this is a new model system
```

In [ ]:
```
class student:
  pass
s1=student()
s2=student()
s1.name='vidhya'
s2.name='ann'
print(s1.name)
```

```
vidhya
```

In [ ]:
```
#init method
class computer:
    def __init__(self):
       print("you are in init method")
    def features(self):
       print("this is a new model system")
comp1=computer()
comp1.features()
computer.features(comp1)
```

```
you are in init method
this is a new model system
this is a new model system
```

In [1]:
```
#instance variable
class computer:

    def features(self):
        self.name="mary"
        print(self.name)
    def nwfeatures(self):

        print(self.name)
comp1=computer()
comp1.features()
comp1.nwfeatures()
```

```
mary
mary
```

In [ ]:
```
#passing variabls to method
class computer:
    def __init__(self,cpu,ram):
       self.processor=cpu
       self.memory=ram
    def features(self):
       print("this is a new model system")
       print(self.processor,self.memory)
comp1=computer("i5","16gb")
comp2=computer("i4","18gb")

comp2.features()
```

```
this is a new model system
i4 18gb
```

In [ ]:
```
class computer:
      def features(self,processor,ram):
       self.processor=processor
       self.ram=ram
       print("this is a new model system")
       print(self.processor,self.ram)
      def new(self):
        print(self.processor)
comp1=computer()
comp1.features("i5","16gb")
comp1.new()
```

```
this is a new model system
i5 16gb
i5
```

In [9]:
```
#instance and class variables
class Person:
  def __init__(self, name):
    self.name = name
class Employee(Person):
  def isEmployee(self):
    return True
  def isEmployee(self):
    return False
  def getName(self):
    return self.name

e = Employee("Ammu")
print(e.getName(),  e.isEmployee())
p = Person("Anu")
print(p.getName(),  p.isEmployee())
```

```
Ammu False

---------------------------------------------------------------------------
----
AttributeError                            Traceback (most recent call l
ast)
<ipython-input-9-f23ee7e2f6be> in <module>()
     14 print(e.getName(),  e.isEmployee())
     15 p = Person("Anu")
---> 16 print(p.getName(),  p.isEmployee())

AttributeError: 'Person' object has no attribute 'getName'
```

In [ ]:
```
#display complex numbers
class complex1:
    def __init__(self,i,j):
       self.real=i
       self.imaginary=j
    def number(self):
       print("{}+{}j".format(self.real,self.imaginary))
comp1=complex1(5,6)
comp1.number()
```

```
5+6j
```

In [ ]:
```
#add two complex numbers
class complexnw:
    def __init__(self,i,j):
       self.real=i
       self.imaginary=j
    def add(self,obj):
       print(self.real+obj.real)
       print(self.imaginary+obj.imaginary)

complex1=complexnw(5,6)

complex2=complex(7,8)
complex1.add(complex2)
```

```
12
14
```

In [ ]:
```
class Rectangle:
    def __init__(self,length=0,breadth=0):
        self.length=length
        self.breadth=breadth
    def area(self):
        print("area=",self.length*self.breadth)
R1=Rectangle(10,20)
R1.area()
R2=Rectangle(12,13)
R2.area()
R3=Rectangle()
R3.area()
```

```
area= 200
area= 156
area= 0
```

In [ ]:
```
#instance variable and class variable
class Rectangle:
    perimeter=15
    def __init__(self,length=0,breadth=0):
        self.length=length
        self.breadth=breadth
    def area(self):
        print("area=",self.length*self.breadth)
R1=Rectangle(10,20)
R2=Rectangle()
print(R1.length)

print(R2.length)
print(R1.perimeter)

print(R2.perimeter)
Rectangle.perimeter=13
print(R1.perimeter)

print(R2.perimeter)
```

```
10
0
15
15
13
13
```

In [14]:
```
class Rectangle:

    def __init__(self,length=0,breadth=0):
        self.length=length
        self.breadth=breadth
    def area(self):
        print("area=",self.length)
    def classmethod(cls):
        print("this is a class method", self.breadth)
R1=Rectangle(10,20)
R2=Rectangle()
print(R1.length)
```

```
10
```

In [ ]:
```
#Create a class car with attributes model,
 #year and price and a method cost() for displaying the prize.
 #Create two instance of the class and call the method for each instan
ce.(university question)"""
class Car:
     def __init__(self,model,year,prize):
        self.model=model
        self.year=year
        self.prize=prize
     def cost(self):
        print ("Prize of the car=",self.prize)
C1=Car("Maruti",2004,200000)
C2=Car("Ford",2014,5000000)
C1.cost()
C2.cost()
```

```
Prize of the car= 200000
Prize of the car= 5000000
```

In [ ]:
```
#Create a class student with attribute #name and roll number and a
#method dataprint() for displaying the same.
#Create two instance of the class and call the method for each instanc
e.( university question)
class Student:
    def __init__(self,name,rno):
        self.name=name
        self.rno=rno
    def dataprint(self):
        print ("Name=",self.name)
        print ("Rno=",self.rno)
s1=Student("devi",101)
s2=Student("anjana",102)
sq1.dataprint()
s2.dataprint()
```

```
Name= devi
Rno= 101
Name= anjana
Rno= 102
```

In [10]:
```
#Define a class in Python to
#store the details of students( rollno, mark1,mark2)
#with the following methods
#readData()- to assign values to class attributes
#computeTotal()-to find the total marks
#printDetails()- to print the attribute values and total marks.
#Create an object of this class and invoke the methods. ( Univesrsity
 question)
class Student:
    def readData(self):
        self.rollno=input("enter roll number...")
        self.mark1=int(input("enter mark1.."))
        self.mark2=int(input("enter mark2.."))
    def computeTotal(self):
        self.total=self.mark1+self.mark2
    def printDetails(self):
        print ("roll number-->",self.rollno)
        print ("Mark1-------->",self.mark1)
        print( "Mark2-------->",self.mark2)
        print( "Total Marks---",self.total)
S=Student()
S.readData()
S.computeTotal()
S.printDetails()
```

```
enter roll number...2
enter mark1..34
enter mark2..33
roll number--> 2
Mark1--------> 34
Mark2--------> 33
Total Marks--- 67
```

In [17]:
```
#mutator and accessor
class Fruit:
    def   __init__(self, name):
       self.name = name

    def setFruitName(self, name):
       self.name = name

    def getFruitName(self):
        return self.name

f1 = Fruit("Apple")
print("First fruit name: ", f1.getFruitName())
f1.setFruitName("Grape")
print("Second fruit name: ", f1.getFruitName())
```

```
First fruit name:  Apple
Second fruit name:  Grape
```

In [18]:
```
class Student:
    def __init__(self, name):
      print("This is parameterized constructor")
      self.name = name
    def show(self):
       print("Hello ", self.name)
s1 = Student("Rahul")
s1.show()
```

```
This is parameterized constructor
Hello  Rahul
```

In [ ]:

In [ ]:
```
#multiple inheritance
class A:
   def funct1(self):
     print("ur in A")
class B:
   def funct1(self):
      print("ur in B")
class C(A,B):
   def funct1(self):
      print("ur in C")
objc=C()
objc.funct1()
```

```
ur in C
```

In [ ]:
```
#multiple inheritance-init method
class A:
   def __init__(self):
     print("init of A")
   def funct1(self):
     print("ur in A")
class B(A):
   def funct2(self):
      print("ur in B")
objc=B()
```

```
init of A
```

In [ ]:
```
#not invoking init method of parent
class Parent:
    def __init__(self):
        self.parent_attribute = 'I am a parent'
    def parent_method(self):
        print('parent class')
# Create a child class that inherits from Parent
class Child(Parent):
    def __init__(self):
        self.child_attribute = 'I am a child'
# Create instance of child
child = Child()
# Show attributes and methods of child class
print(child.child_attribute)
print(child.parent_attribute)
child.parent_method()
```

```
I am a child

---------------------------------------------------------------------------
----
AttributeError                            Traceback (most recent call l
ast)
<ipython-input-2-3b08e731fe08> in <module>()
     12 # Show attributes and methods of child class
     13 print(child.child_attribute)
---> 14 print(child.parent_attribute)
     15 child.parent_method()

AttributeError: 'Child' object has no attribute 'parent_attribute'
```

In [ ]:
```
#one way to invoke parent init
class Parent:
    def __init__(self):
        self.parent_attribute = 'I am a parent'
    def parent_method(self):
        print('parent class')
# Create a child class that inherits from Parent
class Child(Parent):
    def __init__(self):
        Parent.__init__(self)
        self.child_attribute = 'I am a child'
# Create instance of child
child = Child()
# Show attributes and methods of child class
print(child.child_attribute)
print(child.parent_attribute)
child.parent_method()
```

```
I am a child
I am a parent
parent class
```

In [ ]:
```python
#using super()
class Parent:
    def __init__(self):
        self.parent_attribute = 'I am a parent'
    def parent_method(self):
        print('parent class')
# Create a child class that inherits from Parent
class Child(Parent):
    def __init__(self):
        super().__init__()
        self.child_attribute = 'I am a child'
# Create instance of child
child = Child()
# Show attributes and methods of child class
print(child.child_attribute)
print(child.parent_attribute)
child.parent_method()
```

```
I am a child
I am a parent
parent class
```

In [ ]:
```python
#multiple inheritance
class B:
    def b(self):
        print('b')
class C:
    def c(self):
        print('c')
class D(B, C):
    def d(self):
        print('d')
d = D()
d.b()
d.c()
d.d()
```

```
b
c
d
```

In [ ]:
```
#multiple resolution order(MRO)
class B:
    def x(self):
        print('x: B')
class C:
    def x(self):
        print('x: C')
class D(B, C):
    pass
d = D()
d.x()
print(D.mro())
```

```
x: B
[<class '__main__.D'>, <class '__main__.B'>, <class '__main__.C'>, <cla
ss 'object'>]
```

In [ ]:
```
class First():
  def __init__(self):
    print ("First(): entering")
    super().__init__()
    print ("First(): exiting")

class Second():
  def __init__(self):
    print ("Second(): entering")
    super().__init__()
    print ("Second(): exiting")

class Third(First, Second):
  def __init__(self):
    print ("Third(): entering")
    super().__init__()
    print ("Third(): exiting")
print(Third.mro())
t1=Third()
```

```
[<class '__main__.Third'>, <class '__main__.First'>, <class '__main__.S
econd'>, <class 'object'>]
Third(): entering
First(): entering
Second(): entering
Second(): exiting
First(): exiting
Third(): exiting
```

In [ ]:

```
# Base class
class Parent:
      def func1(self):
          print("This function is in parent class.")
# Derived class1
class Child1(Parent):
      def func2(self):
          print("This function is in child 1.")
# Derivied class2
class Child2(Parent):
      def func3(self):
          print("This function is in child 2.")

object1 = Child1()
object2 = Child2()
object1.func1()
object1.func2()
object2.func1()
object2.func3()
```

```
This function is in parent class.
This function is in child 1.
This function is in parent class.
This function is in child 2.
```

In [ ]:
```
#It describes the idea of wrapping data
 #and the methods that work on data within one unit.
 #This puts restrictions on accessing variables and methods directly
 #and can prevent the accidental modification of data
 #public attribute
class encap:
  __a=10
  def hello(self):
    print("hello")
obj=encap()
obj.hello()
print(obj.__a)
```

```
hello

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call l
ast)
<ipython-input-1-d8ac727724a9> in <module>()
     10 obj=encap()
     11 obj.hello()
---> 12 print(obj.__a)

AttributeError: 'encap' object has no attribute '__a'
```

In [ ]:
```
class encap:
  __a=10
  def hello(self):
    print("hello")
obj=encap()
obj.hello()
print (obj.__a)
```

```
hello

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call l
ast)
<ipython-input-3-c75b7f342134> in <module>()
      5 obj=encap()
      6 obj.hello()
----> 7 print (obj.__a)

AttributeError: 'encap' object has no attribute '__a'
```

In [ ]:
```
#private attribute inside the class
class encap:
  __a=10
  def hello(self):
    print("hello")
    print (self.__a)
obj=encap()
obj.hello()
```

```
hello
10
```

In [ ]:
```
#by default all methods are public
class disp:
  def disp1(self):
    print("u r in disp1")
  def disp2(self):
    print("u r in disp2")
obj1=disp()
obj1.disp1()
obj1.disp2()
```

```
u r in disp1
u r in disp2
```

In [ ]:
```
#encapsulating methods
class disp:
  def __disp1(self):
    print("u r in disp1")
  def disp2(self):
    print("u r in disp2")

obj1=disp()
obj1.disp1()
obj1.disp2()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call l
ast)
<ipython-input-14-40a52dcef90f> in <module>()
      5     print("u r in disp2")
      6 obj1=disp()
----> 7 obj1.disp1()
      8 obj1.disp2()

AttributeError: 'disp' object has no attribute 'disp1'
```

In [ ]:
```python
#CALLING A PRIVATE METHOD IN A PUBLIC CLASS
class disp:
  def __disp1(self):
    print("u r in disp1")
  def disp2(self):
    print("u r in disp2")
    self.__disp1()
obj1=disp()
obj1.disp2()
```

```
u r in disp2
u r in disp1
```

In [ ]:
```
#straightforward demonstration of polymorphism in Python
print(4+5)
print("4"+"5")
print("ab"+"cd")
```

```
9
45
abcd
```

In [ ]:
```
#operator overloading
a=3
b=5
print(a+b)
print(int.__add__(a,b))
```

```
8
8
```

In [ ]:
```
class stud:
  def name(self):
    print("hello")
o=stud()
print (o)
```

```
<__main__.stud object at 0x7f344f68f0d0>
```

In [ ]:
```
class stud:
  def name(self):
    print("hello")
  def __str__(self):
    return ( "your modifying ur print function")
o=stud()
print (o)
```

```
your modifying ur print function
```

In [ ]:
```python
#operator overloading
class batsman:
  def __init__(self,a,b):
    self.a=a
    self.b=b
  def __add__(self,other):
    sum1=self.a+other.a
    sum2=self.b+other.b
    print(sum1,sum2)
    bat3=batsman(sum1,sum2)
    return bat3
  def __str__(self):
    return "{} is the sum of first scores and {} is the sum of second
 scores".format(self.a,self.b)
bat1=batsman(40,50)
bat2=batsman(80,20)
bat3=bat1+bat2
print(bat3)
```

```
120 70
120 is the sum of first scores and 70 is the sum of second scores
```

In [ ]:
```python
#operator overloading ""&gt;symbol"
class student:
  def __init__(self,m1,m2,m3):
    self.m1=m1
    self.m2=m2
    self.m3=m3
  def __gt__(self,s1):
    sum1=self.m1+self.m2+self.m3
    sum2=s1.m1+s1.m2+s1.m3
    if (sum1>sum2):
      return True
    else:
      return False

stud1=student(30,40,50)
stud2=student(70,20,50)
if stud1>stud2:
  print ("stud1 wins")
else:
  print ("stud2 wins")
```

```
stud2 wins
```

In [ ]:
```
a=3
print(dir(a))
```

```
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__',
'__delattr__', '__dir__', '__divmod__', '__doc__', '__eq__', '__float_
_', '__floor__', '__floordiv__', '__format__', '__ge__', '__getattribut
e__', '__getnewargs__', '__gt__', '__hash__', '__index__', '__init__',
'__init_subclass__', '__int__', '__invert__', '__le__', '__lshift__',
'__lt__', '__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__or_
_', '__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__', '__red
uce__', '__reduce_ex__', '__repr__', '__rfloordiv__', '__rlshift__', '_
_rmod__', '__rmul__', '__ror__', '__round__', '__rpow__', '__rrshift_
_', '__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr_
_', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__truediv_
_', '__trunc__', '__xor__', 'bit_length', 'conjugate', 'denominator',
'from_bytes', 'imag', 'numerator', 'real', 'to_bytes']
```

In [ ]:
```
#operator overloading
class batsman:
  def __init__(self,a,b):
    self.a=a
    self.b=b
  def __add__(self,other):
    sum1=self.a+other.a
    sum2=self.b+other.b
    return sum1,sum2
bat1=batsman(40,50)
bat2=batsman(80,20)
3,bat4=bat1+bat2
print(bat3,bat4)
```

```
120 70
```

In [ ]:
```
#sub method
class man:
  def __init__(self,hgt):
    self.hgt=hgt
  def __sub__(self,s1):
    difference=self.hgt-s1.hgt
    return difference
man1=man(160)
man2=man(175)
dif=man1-man2
print(dif)
```

```
-15
```

In [ ]:
```python
#polymorphism in class methods
class India():
    def capital(self):
        print("New Delhi is the capital of India.")
    def language(self):
        print("Hindi is the most widely spoken language of India.")
class USA():
    def capital(self):
        print("Washington, D.C. is the capital of USA.")
     def language(self):
        print("English is the primary language of USA.")
obj_ind = India()
obj_usa = USA()
for country in (obj_ind, obj_usa):
    country.capital()
    country.language()
```

```
New Delhi is the capital of India.
Hindi is the most widely spoken language of India.
Washington, D.C. is the capital of USA.
English is the primary language of USA.
```

In [ ]:
```python
#method overriding
class Bird:
     def flight(self):
       print("Most of the birds can fly but some cannot")
class parrot(Bird):
     def flight(self):
       print("Parrots can fly")
class penguin(Bird):
     def flight(self):
      print("Penguins do not fly")
obj_bird = Bird()
obj_parr = parrot()
obj_peng = penguin()
obj_bird.flight()
obj_parr.flight()
obj_peng.flight()
```

```
Most of the birds can fly but some cannot
Parrots can fly
Penguins do not fly
```

In [ ]:
```
#method overloading
class bird:
  def brdclass(self,name=None):
    self.name=name
    if self.name=="parrot":
      print("can fly")
    if self.name=="penguin":
      print("cannt fly")
    if self.name==None:
      print("not a bird")
birdobj=bird()
birdobj.brdclass("parrot")
birdobj.brdclass()
```

```
can fly
not a bird
```

In [ ]:
```
#abstraction-hiding the information-giving access to the information n
eeded
from abc import ABC,abstractmethod
class computer(ABC):
  @abstractmethod
  def process(self):
    pass
c1=computer()
```

```
---------------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-14-c8c68480bb4c> in <module>()
      5   def process(self):
      6     pass
----> 7 c1=computer()

TypeError: Can't instantiate abstract class computer with abstract meth
ods process
```

In [ ]:
```
#abstraction
from abc import ABC,abstractmethod
class computer(ABC):
  @abstractmethod
  def process(self):
    pass
```

In [ ]:
```python
#abstraction
from abc import ABC,abstractmethod
class computer(ABC):
  @abstractmethod
  def process(self):
    pass
class laptop(computer):
  def process(self):
    print("its running")
comp1=laptop()
comp1.process()
```

```
its running
```

In [ ]:
```python
class X(object):
    def __init__(self, a):
        self.num = a
    def doubleup(self):
        self.num *= 2

class Y(X):
    def __init__(self, a):
        X.__init__(self, a)
    def tripleup(self):
        self.num *= 3

obj = Y(4)
print(obj.num)

obj.doubleup()
print(obj.num)

obj.tripleup()
print(obj.num)
```

```
4
8
24
```

In [ ]:
```
#syntax errors
a=2
prin(a)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-a7880554a2fb> in <module>()
      1 #syntax errors
      2 a=2
----> 3 prin(a)

NameError: name 'prin' is not defined
```

In [ ]:
```
a=2
b=0
print(a/b)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-2-c6c186332615> in <module>()
      1 a=2
      2 b=0
----> 3 print(a/b)

ZeroDivisionError: division by zero
```

In [ ]:
```
#try except block to handle zero division error
a=2
b=0
try:
  print(a/b)
except Exception as e:
  print("reason",e)
```

```
reason division by zero
```

In [ ]:

```
#multiple exception
a=2
while True:
 try:
   b=int(input("enter the number"))
   print(a/b)
 except ZeroDivisionError :
   print("division by zero not possible")
 except ValueError :
   print("enter a valid number")
```

```
enter the number4
0.5
enter the numberdf
enter a valid number
enter the number0
division by zero not possible
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident, parent, password)
    728             try:
--> 729                 ident, reply = self.session.recv(self.stdin_socket, 0)
    730             except Exception:

/usr/local/lib/python3.7/dist-packages/jupyter_client/session.py in recv(self, socket, mode, content, copy)
    802         try:
--> 803             msg_list = socket.recv_multipart(mode, copy=copy)
    804         except zmq.ZMQError as e:

/usr/local/lib/python3.7/dist-packages/zmq/sugar/socket.py in recv_multipart(self, flags, copy, track)
    726         """
--> 727         parts = [self.recv(flags, copy=copy, track=track)]
    728         # have first part already, only loop while more to receive

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket._recv_copy()

/usr/local/lib/python3.7/dist-packages/zmq/backend/cython/checkrc.pxd in zmq.backend.cython.checkrc._check_rc()

KeyboardInterrupt:

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-5-553d726e5acb> in <module>()
      3 while True:
      4  try:
----> 5    b=int(input("enter the number"))
      6    print(a/b)
      7  except ZeroDivisionError :

/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in raw_input(self, prompt)
    702             self._parent_ident,
    703             self._parent_header,
--> 704             password=False,
    705         )
    706

/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in _input
```

```
t_request(self, prompt, ident, parent, password)
    732             except KeyboardInterrupt:
    733                 # re-raise KeyboardInterrupt, to truncate trace
back
--> 734                 raise KeyboardInterrupt
    735             else:
    736                 break

KeyboardInterrupt:
```

In [ ]:

```
#multiple exception
a=2
while True:
 try:
   b=int(input("enter the number"))
   print(a/b)
 except (ZeroDivisionError,ValueError) :
   print("invalid entry")
```

```
enter the number4
0.5
enter the numberwe
invalid entry
```

```
---------------------------------------------------------------------
----
KeyboardInterrupt                         Traceback (most recent call l
ast)
/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in _inpu
t_request(self, prompt, ident, parent, password)
    728             try:
--> 729                 ident, reply = self.session.recv(self.stdin_soc
ket, 0)
    730             except Exception:

/usr/local/lib/python3.7/dist-packages/jupyter_client/session.py in rec
v(self, socket, mode, content, copy)
    802         try:
--> 803             msg_list = socket.recv_multipart(mode, copy=copy)
    804         except zmq.ZMQError as e:

/usr/local/lib/python3.7/dist-packages/zmq/sugar/socket.py in recv_mult
ipart(self, flags, copy, track)
    726         """
--> 727         parts = [self.recv(flags, copy=copy, track=track)]
    728         # have first part already, only loop while more to rece
ive

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv
()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv
()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket._recv_copy()

/usr/local/lib/python3.7/dist-packages/zmq/backend/cython/checkrc.pxd i
n zmq.backend.cython.checkrc._check_rc()

KeyboardInterrupt:

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call l
ast)
<ipython-input-2-ec216668ec24> in <module>()
      3 while True:
      4  try:
----> 5    b=int(input("enter the number"))
      6    print(a/b)
      7  except (ZeroDivisionError,ValueError) :

/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in raw_i
nput(self, prompt)
    702             self._parent_ident,
    703             self._parent_header,
--> 704             password=False,
    705         )
    706

/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in _inpu
```

```
t_request(self, prompt, ident, parent, password)
    732             except KeyboardInterrupt:
    733                 # re-raise KeyboardInterrupt, to truncate trace
back
--> 734                 raise KeyboardInterrupt
    735             else:
    736                 break

KeyboardInterrupt:
```

In [ ]:
```
#else block
a=2
while True:
 try:
   b=int(input("enter the number"))
   print(a/b)
 except ZeroDivisionError :
   print("division by zero not possible")
 except ValueError :
   print("enter a valid number")
 else:
   print("the result after division is obtained")
```

```
0.6666666666666666
the result after division is obtained
```

In [5]:
```
#finally block
a=2
while True:
 try:
   b=int(input("enter the number"))
   print(a/b)
 except ZeroDivisionError :
   print("division by zero not possible")
 except ValueError :
   print("enter a valid number")
 else:
   print("the result after division is obtained")
 finally:
   print("this statement will be printed in all cases")
   break
```

```
enter the number0
division by zero not possible
this statement will be printed in all cases
```

In [ ]:
```
#exception as argument
a=2
b=0
try:
  print(a/b)
except Exception as e:
  print("the reason for exception is ",e)
```

```
the reason for exception division by zero
```

In [6]:
```
#exception as argument
a=2
b=0
try:
  print(a/b)
except Exception as e:
  print("the reason for exception is ",e)
```

```
the reason for exception is  division by zero
```

In [8]:
```
#raise an exception
while True:
  age=int(input("enter the age"))
  try:
    if (age<15):
      raise ValueError
    else:
      print("you are elligible for college admission ")
  except ValueError:
      print("age should be avove 15 for college admission")
      break
```

```
enter the age7
age should be avove 15 for college admission
```

In [9]:
```
#user defined exception class
class Error (Exception):
  pass
class ValuetooSmall(Error):
  pass
class ValuetooHigh(Error):
  pass
number=10
while True:
  guess_no=int(input("enter the number"))
  try:
    if (guess_no<number):
      print("number too low")
      raise ValuetooSmall
    elif (guess_no>number):
      print("number too large")
      raise ValuetooHigh
    else:
      print("you won!!!!correct guessing")
      break
  except ValuetooSmall:
    print("entered value is smaller")
  except ValuetooHigh:
    print("entered value is higher")
```

```
enter the number7
number too low
entered value is smaller
enter the number12
number too large
entered value is higher
enter the number10
you won!!!!correct guessing
```

In [ ]:
```
def line_count():
    file = open("/content/book.txt","r")
    count=0
    for line in file:
        print(line)
    file.close()
    print("No of lines not starting with 'T'=",count)

line_count()
```

```
reading and writing files, programs can exchange information.

with each other and generate printable formats like PDF.

Working with files is a lot like working with books.

To use a book, you have to open it. When you’re done,

you have to close it.

 While the book is open, you can either write in it or read from it.

 In either case, you know where you are in the book.

Most of the time, you read the whole book in its natural order, but you
can also skip around.

All of this applies to files as well.
No of lines not starting with 'T'= 0
```

In [ ]:
```
#Write a function in python to count the number of lines from a text f
ile "story.txt"
#which is not starting with an alphabet "T".
def line_count():
    file = open("/content/book.txt","r")
    count=0
    for line in file:
        if line[0] not in 'T':
            count+= 1
    file.close()
    print("No of lines not starting with 'T'=",count)

line_count()
```

```
No of lines not starting with 'T'= 8
```

In [ ]:
```
#display_words() in python to read lines from a text file "story.txt",
and
#display those words, which are less than 4 characters.
def display_words():
    file = open("/content/book.txt","r")
    data = file.read()
    words = data.split()
    for word in words:
        if len(word) < 4:
            print(word, end=" ")
    file.close()

display_words()
```

```
and can and is a lot To use a you to it. you to it. the is you can in i
t or it. In you you are in the of the you the in its but you can All of
to as
```

In [ ]:
```
import pickle
mylist=['a','b','c']
with open ("data.jpg",'wb') as fp:
  pickle.dump(mylist,fp)
```

In [ ]:
```
import pickle
with open ("data.jpg",'rb') as fp:
  data=pickle.load(fp)
print(data)
```

In [ ]:
```
import pickle
def add_record():
    outfile = open('emp.dat', 'ab')
    empcode = int(input('Enter Employee code: '))
    name = input('Enter Employee name: ')
    salary = int(input('Enter salary: '))
    employee = [empcode,name,salary]
    pickle.dump(employee, outfile)
    outfile.close()

def read_records():
      infile = open('emp.dat', 'rb')
      employee = pickle.load(infile)
      print('Employee code:', employee[0])
      print('Employee name:', employee[1])
      print('Salary:', employee[2])
      infile.close()
add_record()
read_record()
```

In [ ]:

```
#1. os.name
import os
print(os.name)
```

```
posix
```

In [ ]:

```
#2.os.getcwd()
import os
print(os.getcwd())
```

```
/content
```

In [ ]:

```
#3.os.listdir('.')-To print files and directories in the current directory  on your sys
tem
import os
print(os.listdir('.'))
```

```
['.config', 'sample_data']
```

In [ ]:

```
#os.chdir('..')-This function is used to change the CWD
import os
print(os.getcwd())
os.chdir('..')
print(os.getcwd())
```

```
/content
/
```

In [ ]:

```
#makedirectory-mkdir
os.mkdir("/content/python")
print(os.listdir('.'))
```

```
['media', 'srv', 'var', 'mnt', 'etc', 'root', 'tmp', 'lib', 'proc', 'sbi
n', 'lib64', 'boot', 'run', 'sys', 'bin', 'dev', 'usr', 'opt', 'home', 'co
ntent', 'python', '.dockerenv', 'tools', 'datalab', 'lib32', 'python-apt',
'NGC-DL-CONTAINER-LICENSE']
```

In [ ]:

```
#remove directory
os.rmdir("/content/python")
```

In [ ]:

```
#remove a file
os.remove(path)
```

In [ ]:

```
#remove a file
os.remove('s.py')
```

```
---------------------------------------------------------------------------
-
FileNotFoundError                         Traceback (most recent call las
t)
<ipython-input-22-b33dfde451db> in <module>
      1 #remove a file
----> 2 os.remove('s.py')

FileNotFoundError: [Errno 2] No such file or directory: 's.py'
```

In [ ]:

```
os.rename(old.new)
```

In [ ]:

```
os.mkdir('/content/remya')
os.chdir('\content\remya')
for (root,dirs,files) in os.walk('.', topdown=True):
        print(root )
        print(dirs )
        print(files)
        print('--------------------------------')
```

```
---------------------------------------------------------------------------
-
FileExistsError                           Traceback (most recent call las
t)
<ipython-input-31-6021539ce6ee> in <module>
----> 1 os.mkdir('/content/remya')
      2 os.chdir('\content\remya')
      3 for (root,dirs,files) in os.walk('.', topdown=True):
      4         print(root )
      5         print(dirs )

FileExistsError: [Errno 17] File exists: '/content/remya'
```

In [ ]:

```
import os
os.mkdir('/content/hello2')
```

In [ ]:

```
#For each directory in the tree rooted at directory top (including top itself), it yiel
ds a 3-tuple (dirpath, dirnames, filenames).
#root : Prints out directories only from what you specified.
import os
os.chdir('/content/hello2')
for (root,dirs,files) in os.walk('.', topdown=True):
        print(root )
        print(dirs )
        print(files)
        print('--------------------------------')
```

```
.
['.ipynb_checkpoints', 'folder1']
['pgm1.py']
--------------------------------
./.ipynb_checkpoints
[]
[]
--------------------------------
./folder1
[]
[]
--------------------------------
```

In [ ]:

```
os.path.isfile("/content/hello2/pgm1.py")
```

Out[ ]:

```
True
```

In [ ]:

```
os.path.isdir("/content/hello2")
```

Out[ ]:

```
True
```

In [ ]:

```
os.path.getsize("/content/hello2/pgm1.py")
```

Out[ ]:

```
15
```

In [ ]:

```
os.path.exists("/content/hello2/pgm1.py")
```

Out[ ]:

```
True
```

In [ ]:

```
#The following program will count the number of
#files in the current directory.
import os
def countfiles(path):
    count=0
    lst=os.listdir(path)
    for f in lst:
        if os.path.isfile(f):
            count=count+1
        else:
            os.chdir(f)
            count=count+countfiles(os.getcwd())
            os.chdir('..')
    return count

c=countfiles('.')
print("number of files...=",c)
```

```
number of files...= 1
```

In [ ]:

```
#Print the names of the files in the current directory having ".py" extension.
import os
path=os.getcwd()
lst=os.listdir(path)
for f in lst:
     if '.py' in f:
           print(f)
```

```
pgm1.py
```

In [ ]:

```
#The current version number of Python
import sys
print(sys.version)
```

```
3.7.13 (default, Apr 24 2022, 01:04:09)
[GCC 7.5.0]
```

In [ ]:

```
import sys
print(sys.version)
print(sys.version_info)
#search path for all Python modules
print(sys.path)
```

```
3.7.13 (default, Apr 24 2022, 01:04:09)
[GCC 7.5.0]
sys.version_info(major=3, minor=7, micro=13, releaselevel='final', serial=
0)
['/content', '/env/python', '/usr/lib/python37.zip', '/usr/lib/python3.7',
'/usr/lib/python3.7/lib-dynload', '', '/usr/local/lib/python3.7/dist-packa
ges', '/usr/lib/python3/dist-packages', '/usr/local/lib/python3.7/dist-pac
kages/IPython/extensions', '/root/.ipython']
```

In [ ]:

```
#This is generally used to safely exit from
#the program in case of generation of an exception.
import sys
sys.exit()
```

```
An exception has occurred, use %tb to see the full traceback.

SystemExit
```

In [ ]:

```
#Name of the platform on which Python is running, e.g. "linux2" for Linux
#and "win32" for Windows
print(sys.platform)
#A string containing the name of the executable binary (path and executable file name)
 for the Python interpreter.
print(sys.executable)
```

```
linux
/usr/bin/python3
```

In [ ]:

```

```

Python has a module named datetime to work with dates and times.

Commonly used classes in the datetime module are: date Class,time Class,datetime Class and timedelta Class

In [ ]:

```
#date class
import datetime
print(datetime.date(2022, 4, 13))
```

```
2022-04-13
```

In [ ]:

```
#today method of date class
print( datetime.date.today())
today=datetime.date.today()
print(today)
print(today.year)
print(today.month)
print(today.day)
```

```
2022-08-05
2022-08-05
2022
8
5
```

In [ ]:

```
#calculate difference between 2 days
from datetime import date
d1 = date(year = 2018, month = 7, day = 12)
d2 = date(year = 2020, month = 12, day = 23)
print(d2-d1)
```

```
895 days, 0:00:00
```

In [ ]:

```
#time class
```

In [ ]:

```
#time(hour, minute and second)
from datetime  import time
print(time(5, 18, 22))
print(time(hour = 11, minute = 34, second = 56))
# time(hour, minute, second, microsecond)
print(time(5, 18, 22,100))
```

```
05:18:22
11:34:56
05:18:22.000100
```

In [ ]:

```
a = time(11, 34, 56)
print("hour =", a.hour)
print("minute =", a.minute)
print("second =", a.second)
print("microsecond =", a.microsecond)
```

```
hour = 11
minute = 34
second = 56
microsecond = 0
```

In [ ]:

```
from datetime import datetime
dt=datetime(2020,11,18)
print(dt)
dt=datetime(2020, 9, 11, 23, 55, 59, 342380)
```

```
2020-11-18 00:00:00
```

In [ ]:

```
from datetime import datetime
a = datetime(2020, 9, 11, 23, 55, 59, 342380)
print("year =", a.year)
print("month =", a.month)
print("day=",a.day)
print("hour =", a.hour)
print("minute =", a.minute)
print("second =", a.second)
print("timestamp =", a.timestamp())
```

```
year = 2020
month = 9
day= 11
hour = 23
minute = 55
second = 59
timestamp = 1599868559.34238
```

In [ ]:

```
#difference between 2 years and time
d1 = datetime(year = 2018, month = 7, day = 12, hour = 7, minute = 9, second = 33)
d2 = datetime(year = 2020, month = 6, day = 10, hour = 5, minute = 55, second = 13)
print(d2-d1)
```

```
698 days, 22:45:40
```

In [ ]:

```
#timedelta class
 from datetime import timedelta
 t1 = timedelta(weeks = 1, days=2,hours = 1, seconds = 30)
 t2 = timedelta(days = 3, hours = 12, minutes = 4, seconds = 55)
 print(t1-t2)
```

```
5 days, 12:55:35
```

In [ ]:

```
#total seconds
from datetime import timedelta
t = timedelta(days = 4, hours = 2, seconds = 34, microseconds = 235673)
t.total_seconds()
```

Out[ ]:

```
352834.235673
```

In [ ]:

```
#strftime() method is defined under classes date, datetime and time.
#The method creates a formatted string from a given date, datetime or time object.
from datetime import datetime
dt=datetime.now()
print(dt)
print("____________________")
print(dt.strftime("%H:%M:%S"))
print("____________________")
print(dt.strftime("%m/%d/%Y, %H:%M:%S"))
print("____________________")
print(dt.strftime("%d/%m/%Y, %H:%M:%S"))
```

```
2022-08-05 18:15:02.758366
____________________
18:15:02
____________________
08/05/2022, 18:15:02
____________________
05/08/2022, 18:15:02
```

In [ ]:

```
from datetime import timedelta
t1=timedelta(seconds=53)
t2=timedelta(seconds=55)
print(t1-t2)
```

```
-1 day, 23:59:58
```

In [ ]:

```
#You can get the total number of seconds in a timedelta object using total_seconds() me
thod.
from datetime import timedelta
t = timedelta(days = 4, hours = 2, seconds = 34, microseconds = 235673)
print(t.total_seconds())
print(t/2)
print(t*2)
```

```
352834.235673
2 days, 1:00:17.117836
8 days, 4:01:08.471346
```

In [ ]:

```
#The strftime() method is defined under classes date, datetime and time.
#The method creates a formatted string from a given date, datetime or time object.
from datetime import datetime
dt=datetime.now()
print(dt)
print(dt.strftime("%H:%M:%S"))
print(dt.strftime("%m/%d/%Y, %H:%M:%S"))
print(dt.strftime("%d/%m/%Y, %H:%M:%S"))
```

```
2022-08-11 17:44:12.790870
17:44:12
08/11/2022, 17:44:12
11/08/2022, 17:44:12
```

In [ ]:

```
#Get today's date from datetime
from datetime import date
today = date.today()
print("Today's date:", today)
```

```
Today's date: 2022-08-11
```

In [ ]:

```
from datetime import date
today = date.today()
# dd/mm/YY
d1 = today.strftime("%d/%m/%Y")
print("d1 =", d1)
# Textual month, day and year
d2 = today.strftime("%B %d, %Y")
print("d2 =", d2)
# mm/dd/y
d3 = today.strftime("%m/%d/%y")
print("d3 =", d3)
# Month abbreviation, day and year
d4 = today.strftime("%b-%d-%Y")
print("d4 =", d4)
```

```
d1 = 11/08/2022
d2 = August 11, 2022
d3 = 08/11/22
d4 = Aug-11-2022
```

In [ ]:

```
from datetime import datetime
 # datetime object containing current date and time
now = datetime.now()
print("now =", now)
 # dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("date and time =", dt_string)
```

```
now = 2022-08-11 17:57:45.475721
date and time = 11/08/2022 17:57:45
```

In [ ]:
```
#importing numpy if not present pip3 install numpy
import numpy as np
```

In [ ]:
```
#1-DIMENSIONAL ARRAY
a=np.array([1,2,3,4])
print(a)
```

```
[1 2 3 4]
```

In [ ]:
```
#2-DIMENSIONAL ARRAY
a=np.array([[1,2,3,4],[4,5,6,7]])
print(a)
```

```
[[1 2 3 4]
 [4 5 6 7]]
```

In [ ]:
```
#GET DIMENSION
print(a.ndim)
```

```
2
```

In [ ]:
```
#GET shape
print(a.shape)
```

```
(2, 4)
```

In [ ]:
```
#GET type
print(a.dtype)
```

```
int64
```

In [ ]:
```
a=np.array([[1,2,3,4],[4,5,6,7]],dtype="int8")
print(a)
print(a.dtype)
```

```
[[1 2 3 4]
 [4 5 6 7]]
int8
```

In [ ]:
```
#print item size and total size
a=np.array([[1,2,3,4],[4,5,6,7]],dtype="int64")
print(a.itemsize)
print(a.nbytes)
```

```
8
64
```

In [ ]:
```
#get a specific element a[r][c]
a=np.array([[1,2,3,4],[0,5,6,7]])
print(a[1][2])
```

```
6
```

In [ ]:
```
#to print an entire row
a=np.array([[1,2,3,4],[0,5,6,7]])
print(a[1,:])
```

```
[0 5 6 7]
```

In [ ]:
```
a=np.array([[1,2,3,4],[0,5,6,7]])
print(a[1,3])
```

```
7
```

In [ ]:
```
#print selected number a[startindex:stopindex,stepsize]
a=np.array([[1,2,3,4,9,10,11],[0,5,6,7,12,14,15]])
print(a[0,2:5:2])
```

```
[3 9]
```

In [ ]:
```
#changing values in an array
a=np.array([[1,2,3,4,9,10,11],[0,5,6,7,12,14,15]])
a[1,5]=30
print(a)
```

```
[[ 1  2  3  4  9 10 11]
 [ 0  5  6  7 12 30 15]]
```

In [ ]:
```
a[1,:]=2
print(a)
```

```
[[ 1  2  3  4  9 10 11]
 [ 2  2  2  2  2  2  2]]
```

In [ ]:
```
#3 dimensional array
a=np.array([[[1,2],[3,4]],[[5,6],[7,8]],[[9,10],[11,12]]])
print (a)
print(a.ndim)
print(a[1,0,1])
```

```
[[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
3
6
```

In [ ]:
```
#different ways to create an array
```

In [ ]:
```
#all zeros
import numpy as np
a=np.zeros ((2, 3),dtype="int8")
print(a)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

In [ ]:
```
import numpy as np
a=np.zeros ((2, 3),dtype="int8")
print(a)
```

```
[[0 0 0]
 [0 0 0]]
```

In [ ]:
```
#all ones
import numpy as np
a=np.ones((2, 3))
print(a)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

In [ ]:
```
#any other number
import numpy as np
a=np.full((2, 3),5)
print(a)
```

```
[[5 5 5]
 [5 5 5]]
```

In [ ]:
```
#shape of first array value is inserted
import numpy as np
a=np.array([[[1,2],[3,4]],[[5,6],[7,8]],[[9,10],[11,12]]])
a=np.full_like(a,5)
print(a)
```

```
[[[5 5]
  [5 5]]

 [[5 5]
  [5 5]]

 [[5 5]
  [5 5]]]
```

In [ ]:
```
#random numbers
import numpy as np
np.random.randint(low=3,high=10,size=5)
```

Out[ ]:
```
array([7, 5, 7, 5, 6])
```

In [ ]:
```
#np.identity matrix
import numpy as np
ar=np.identity(3)
print(ar)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [ ]:
```
b = np.eye(2, dtype = float)
print("Matrix b : \n", b)

# matrix with R=4 C=5 and 1 on diagonal
# below main diagonal
a = np.eye(4, 5, k = 2)

print("\nMatrix a : \n", a)
```

```
Matrix b :
 [[1. 0.]
 [0. 1.]]

Matrix a :
 [[0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0.]]
```

In [ ]:
```
#arrange
import numpy as np
a=np.arange(0,12,1)
print(a)
print(np.reshape(a,(3,4)))
a=np.arange(0,12,2)
print(a)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11]
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
[ 0  2  4  6  8 10]
```

In [ ]:
```
#linespace
import numpy as np
a=np.linspace(2.0, 3.0, num=8)
print(a)
```

```
[2.         2.14285714 2.28571429 2.42857143 2.57142857 2.71428571
 2.85714286 3.        ]
```

In [ ]:
```
#llogspace
import numpy as np
a=np.logspace(2, 3, num=8)
print(a)
```

```
[ 100.          138.94954944  193.06977289  268.26957953  372.75937203
  517.94746792  719.685673   1000.        ]
```

In [ ]:
```
#identity
import numpy as np
a=np.identity(3)
print(a)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [ ]:

```
# Python program to demonstrate
# basic operations on single array
import numpy as np

a = np.array([1, 2, 5, 3])

# add 1 to every element
print ("Adding 1 to every element:", a+1)

# subtract 3 from each element
print ("Subtracting 3 from each element:", a-3)

# multiply each element by 10
print ("Multiplying each element by 10:", a*10)

# square each element
print ("Squaring each element:", a**2)

# modify existing array
a *= 2
print ("Doubled each element of original array:", a)

# transpose of array
a = np.array([[1, 2, 3], [3, 4, 5], [9, 6, 0]])

print ("\nOriginal array:\n", a)
print ("Transpose of array:\n", a.T)
```

```
Adding 1 to every element: [2 3 6 4]
Subtracting 3 from each element: [-2 -1  2  0]
Multiplying each element by 10: [10 20 50 30]
Squaring each element: [ 1  4 25  9]
Doubled each element of original array: [ 2  4 10  6]

Original array:
 [[1 2 3]
 [3 4 5]
 [9 6 0]]
Transpose of array:
 [[1 3 9]
 [2 4 6]
 [3 5 0]]
```

In [ ]:
```
# Python program to demonstrate
# binary operators in Numpy
import numpy as np

a = np.array([[1, 2],
              [3, 4]])
b = np.array([[4, 3],
              [2, 1]])

# add arrays
print ("Array sum:\n", a + b)

# multiply arrays (elementwise multiplication)
print ("Array multiplication:\n", a*b)
 # matrix dot product
# matrix multiplication
print ("Matrix multiplication:\n", a.dot(b))
# matrix dot product
# multiply matrices with @ operator
D = a @ b
print(D)
```

```
Array sum:
 [[5 5]
 [5 5]]
Array multiplication:
 [[4 6]
 [6 4]]
Matrix multiplication:
 [[ 8  5]
 [20 13]]
[[ 8  5]
 [20 13]]
```

In [ ]:
```
#array division
import numpy as np
A = np.array([[1, 2, 3],[4, 5, 6]])
print(A)
# define second matrix
B = np.array([[1, 2, 3],[4, 5, 6]])
print(B)
# divide matrices
C = A / B
print(C)
```

```
[[1 2 3]
 [4 5 6]]
[[1 2 3]
 [4 5 6]]
[[1. 1. 1.]
 [1. 1. 1.]]
```

In [ ]:
```
# create an array of sine values
a = np.array([0,11,1])
print ("Sine values of array elements:", np.sin(a))

# exponential values
a = np.array([0, 1, 2, 3])
print ("Exponent of array elements:", np.exp(a))

# square root of array values
print ("Square root of array elements:", np.sqrt(a))
```

```
Sine values of array elements: [ 0.         -0.99999021  0.84147098]
Exponent of array elements: [ 1.          2.71828183  7.3890561  20.085
53692]
Square root of array elements: [0.         1.         1.41421356 1.7320
5081]
```

In [ ]:
```
# Python program to demonstrate sorting in numpy
import numpy as np
a = np.array([[1, 4, 2],[3, 4, 6], [0, -1, 5]])
 # sorted array
print ("Array elements in sorted order:\n",
                    np.sort(a,axis=None))

# sort array row-wise
print ("Row-wise sorted array:\n",
                np.sort(a, axis = 1))

# specify sort algorithm
print ("Column wise sort by applying merge-sort:\n",
            np.sort(a, axis = 0, kind = 'mergesort'))
```

```
Array elements in sorted order:
 [-1  0  1  2  3  4  4  5  6]
Row-wise sorted array:
 [[ 1  2  4]
 [ 3  4  6]
 [-1  0  5]]
Column wise sort by applying merge-sort:
 [[ 0 -1  2]
 [ 1  4  5]
 [ 3  4  6]]
```

In [ ]:
```
#append list
import numpy as np
A=np.array([10,20,30])
print(A)
A=np.append(A,[40,50])
print(A)
```

```
[10 20 30]
[10 20 30 40 50]
```

In [ ]:
```
#Add two matrix and find the transpose of the result ( university ques
tion)
def readmatrix(x,r,c):
    for i in range(r):
        for j in range(c):
            x[i][j]=int(input('enter elements row by row'))
import numpy as np
r1=int(input('rows of a'))
c1=int(input('columns of a'))
r2=int(input('rows of b'))
c2=int(input('columns of b'))
if r1!=r2 or c1!=c2:
    print("cant add matrices")
else:
    A=np.zeros((r1,c1))
    print("Enter the elements of A")
    readmatrix(A,r1,c1)
    B=np.zeros((r2,c2))
    print("Enter the elements of B")
    readmatrix(B,r2,c2)
    print("Matrix A")
    print(A)
    print("Matrix B")
    print(B)
    C=A+B
    print("sum")
    print(C)
    print("transpose of sum")
    print(C.T)
```

In [ ]:
```
import numpy
# define orthogonal matrix
Q = np.array([[1, 0],[0, -1]])
print(Q)
# inverse equivalence
V = np.linalg.inv(Q)
print(V)
tran=Q.T
if((V==tran).all()):
 print("orthogonal")
```

```
[[ 1  0]
 [ 0 -1]]
[[ 1.  0.]
 [-0. -1.]]
orthogonal
```

In [ ]:
```
#adding a new row to a array
A=np.array([[1,2],[3,4]])
A=np.append(A,[[5,6]],axis=0)
print(A)
```

```
[[1 2]
 [3 4]
 [5 6]]
```

In [ ]:
```
#adding a new coulumn to a array
A=np.append(A,[[5],[6]],axis=1)
print(A)
```

```
[[1 2 5]
 [3 4 6]]
```

In [ ]:
```
#delete an elemnet from array
A=np.array([10,20,30,40,50,60,70,80])
print(A)
A=np.delete(A,1)
print(A)
```

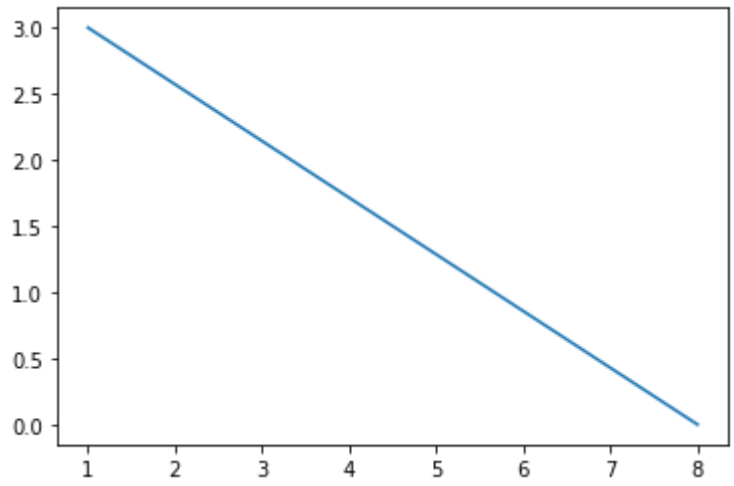
```
[10 20 30 40 50 60 70 80]
[10 30 40 50 60 70 80]
```

In [ ]:
```
A=np.array([[10,20,30],[40,50,60],[70,80,90]])
print(A)
```

```
[[10 20 30]
 [40 50 60]
 [70 80 90]]
```

In [ ]:

```
#Draw a line in a diagram from position (1, 3) to position (8, 10):

import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 8])
ypoints = np.array([3,0])
plt.plot(xpoints,ypoints)
plt.show()
```



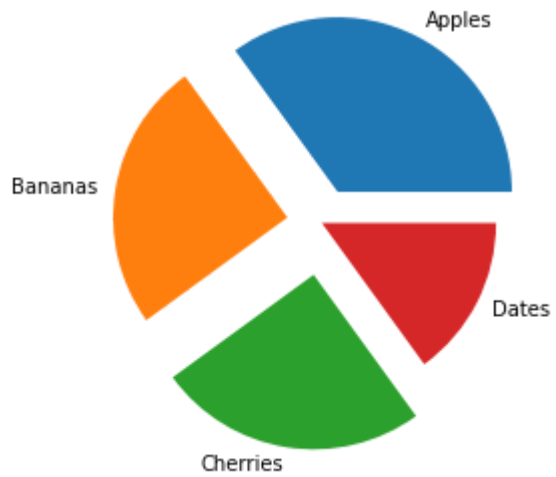
In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
print(np.arange(10))
```

```
[0 1 2 3 4 5 6 7 8 9]
```

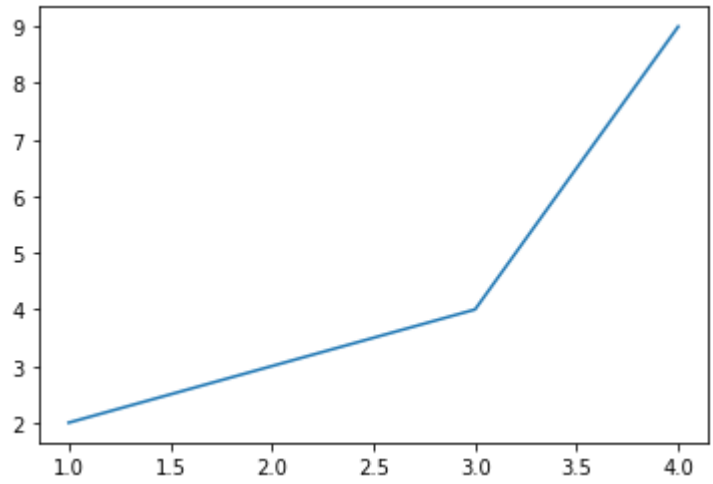
In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
myexplode = [0.2,0.2, 0.3,0]

plt.pie(y, labels = ["Apples", "Bananas", "Cherries", "Dates"],  explode = myexplode)
plt.show()
```



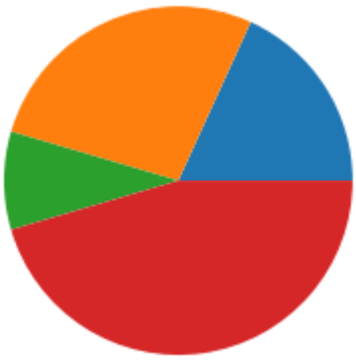
In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.plot([1,2,3,4],[2,3,4,9])
plt.show()
```



In [ ]:

```
#Pie Plot
data=[20,30,10,50]
from pylab import *
pie(data)
show()
```



In [ ]:

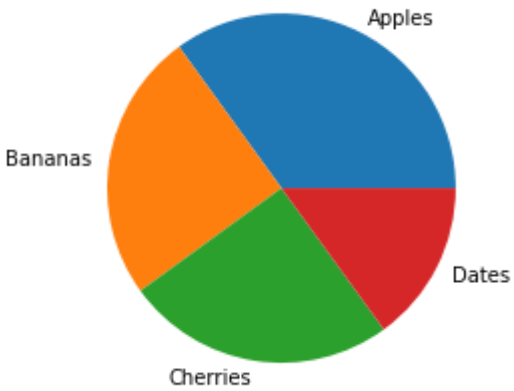
```
#pie chart with explode
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
myexplode = [0.2, 0.3, 0.2, 0]
plt.pie(y, labels = ["Apples", "Bananas", "Cherries", "Dates"],
        explode = myexplode)
plt.show()
```

In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
plt.pie(y)
plt.legend(["Apples", "Bananas","Cherries", "Dates"],title = "Four Fruits:")
plt.show()
```

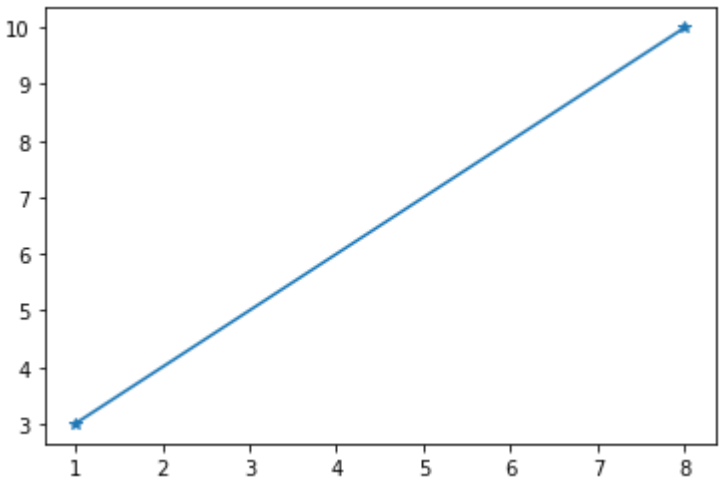
In [ ]:

```
#pie chart
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
plt.pie(y, labels = mylabels)
plt.show()
```



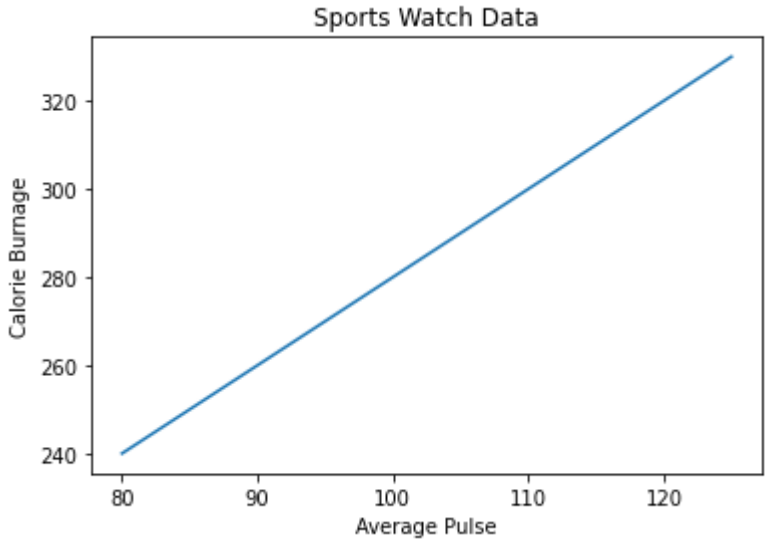
In [ ]:

```
#Draw a line in a diagram from position (1, 3) to position (8, 10): with marker
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints,ypoints,marker='*')
plt.show()
```



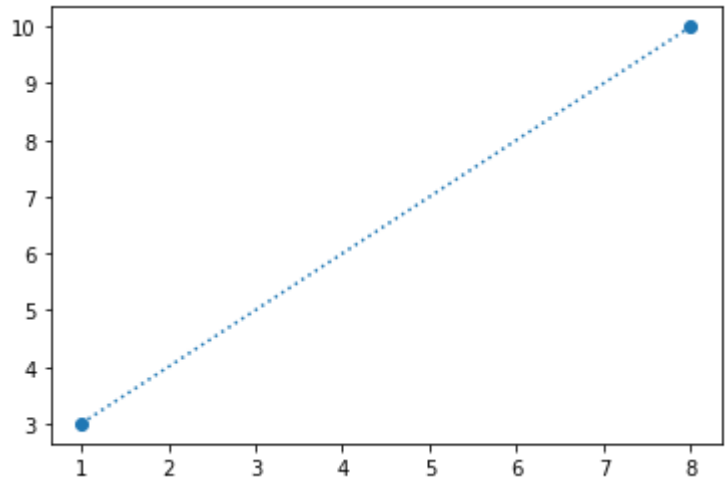
In [ ]:

```
#title-xlabel and ylabel
import numpy as np
import matplotlib.pyplot as plt
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.plot(x, y)
plt.show()
```



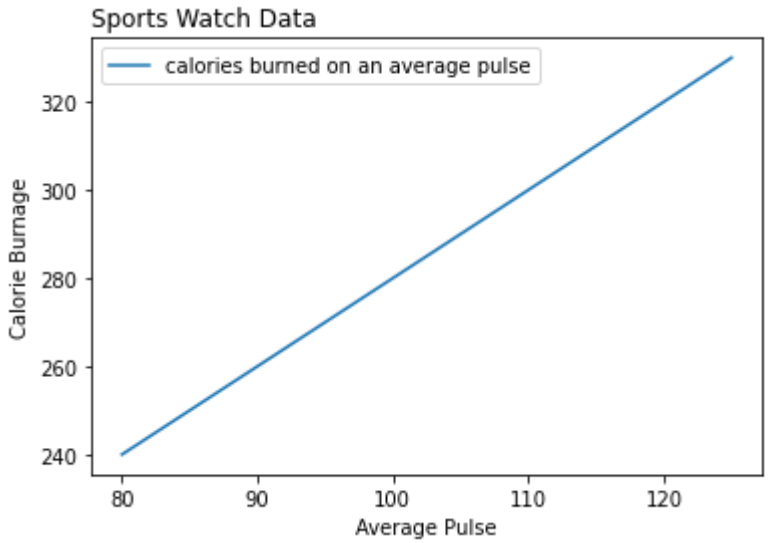
In [ ]:

```
#dotted linestyle
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints,ypoints,marker='o',linestyle="dotted")
plt.show()
```



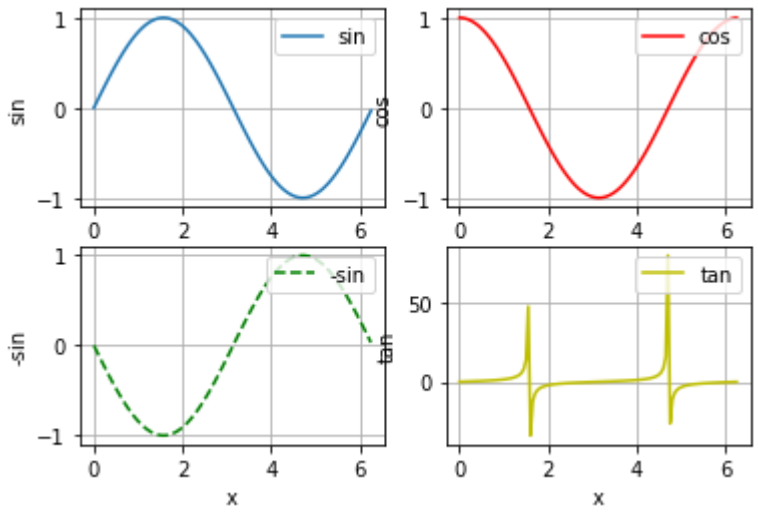
In [ ]:

```
#change location of title and legend
import numpy as np
import matplotlib.pyplot as plt
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.title("Sports Watch Data", loc = 'left')
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.plot(x, y,label="calories burned on an average pulse")
plt.legend()
plt.show()
```



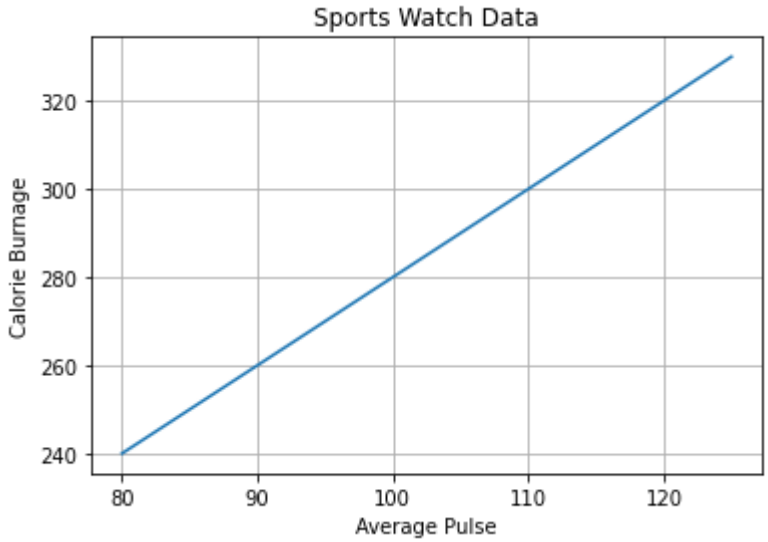
In [ ]:

```
#subplot with  4 figures
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, math.pi*2, 0.05)
subplot(2,2,1)
plot(x, sin(x),label='sin')
xlabel('x')
ylabel('sin')
legend(loc='upper right')
grid(True)
subplot(2,2,2)
plot(x, cos(x), 'r-',label='cos')
xlabel('x')
ylabel('cos')
legend(loc='upper right')
grid(True)
subplot(2,2,3)
xlabel('x')
ylabel('-sin')
plot(x, -sin(x), 'g--',label='-sin')
legend(loc='upper right')
grid(True)
subplot(2,2,4)
xlabel('x')
ylabel('tan')
plot(x, tan(x), 'y-',label='tan')
legend(loc='upper right')
grid(True)
show()
```



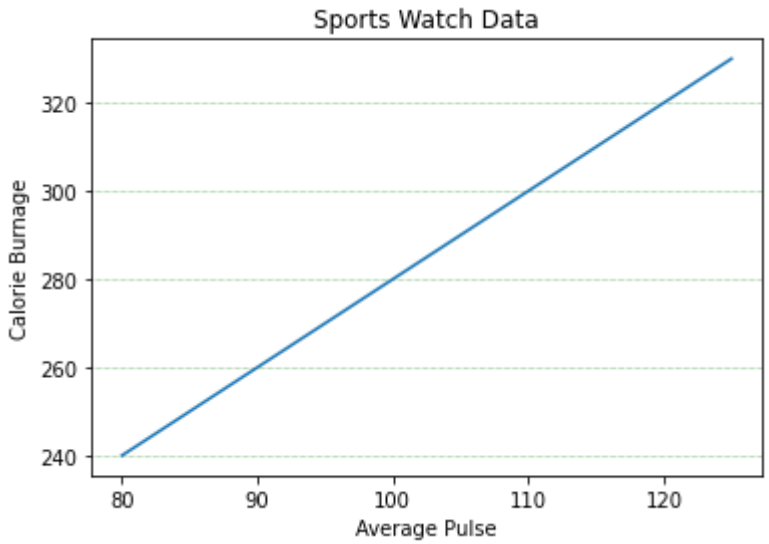
In [ ]:

```
#Add grid lines to the plot:

import numpy as np
import matplotlib.pyplot as plt
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.plot(x, y)
plt.grid()
plt.show()
```



In [ ]:

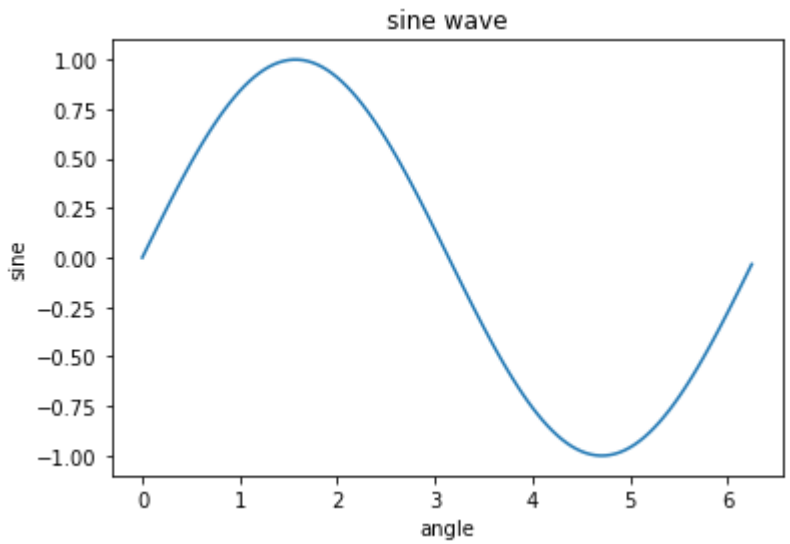
```
#Display only grid lines for the y-axis:

import numpy as np
import matplotlib.pyplot as plt
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.plot(x, y)
plt.grid(color = 'green', linestyle = 'dotted', linewidth = 0.5,axis = 'y')
plt.show()
```



In [ ]:

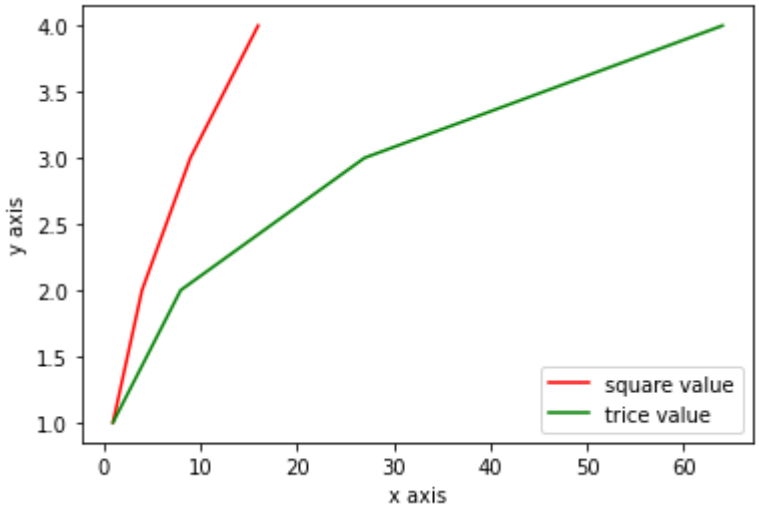
```
import  matplotlib.pyplot as plt
import numpy as np
import math #needed for definition of pi
x = np.arange(0, math.pi*2, 0.05)
print(x)
y = np.sin(x)
plt.plot(x,y)
plt.xlabel("angle")
plt.ylabel("sine")
plt.title('sine wave')
plt.show()
```

```
[0.   0.05 0.1  0.15 0.2  0.25 0.3  0.35 0.4  0.45 0.5  0.55 0.6  0.65
 0.7  0.75 0.8  0.85 0.9  0.95 1.   1.05 1.1  1.15 1.2  1.25 1.3  1.35
 1.4  1.45 1.5  1.55 1.6  1.65 1.7  1.75 1.8  1.85 1.9  1.95 2.   2.05
 2.1  2.15 2.2  2.25 2.3  2.35 2.4  2.45 2.5  2.55 2.6  2.65 2.7  2.75
 2.8  2.85 2.9  2.95 3.   3.05 3.1  3.15 3.2  3.25 3.3  3.35 3.4  3.45
 3.5  3.55 3.6  3.65 3.7  3.75 3.8  3.85 3.9  3.95 4.   4.05 4.1  4.15
 4.2  4.25 4.3  4.35 4.4  4.45 4.5  4.55 4.6  4.65 4.7  4.75 4.8  4.85
 4.9  4.95 5.   5.05 5.1  5.15 5.2  5.25 5.3  5.35 5.4  5.45 5.5  5.55
 5.6  5.65 5.7  5.75 5.8  5.85 5.9  5.95 6.   6.05 6.1  6.15 6.2  6.25]
```



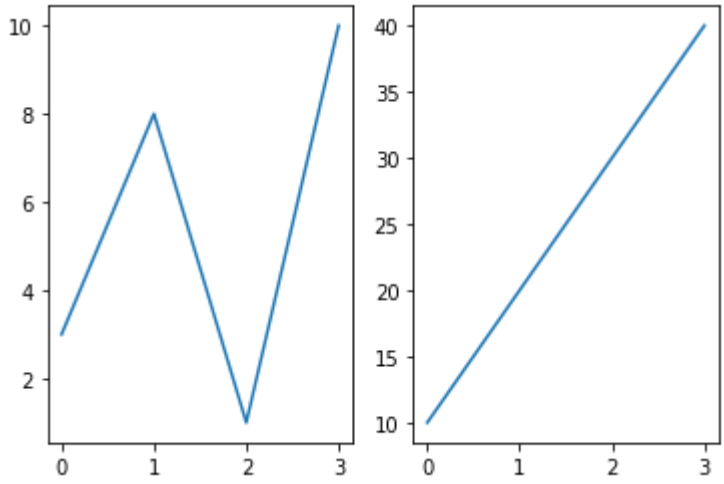
In [ ]:

```
#Legend
import  matplotlib.pyplot as plt
import numpy as np
t=np.array([1,2,3,4])
plt.plot(t**2,t,color="red")
plt.plot(t**3,t,color="green")
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.legend(["square value","trice value"])
plt.show()
```



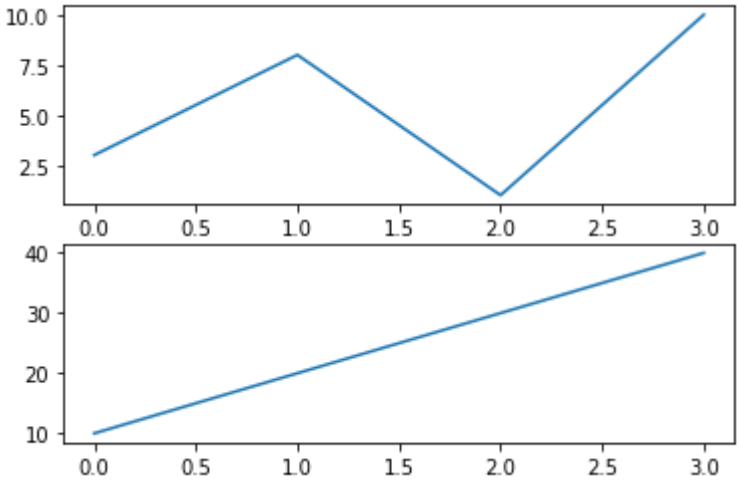
In [ ]:

```
#subplot function
import matplotlib.pyplot as plt
import numpy as np
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
#the figure has 1 row, 2 columns, and this plot is the first plot.
plt.plot(x,y)
#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
#the figure has 1 row, 2 columns, and this plot is the second plot.
plt.plot(x,y)
plt.show()
```



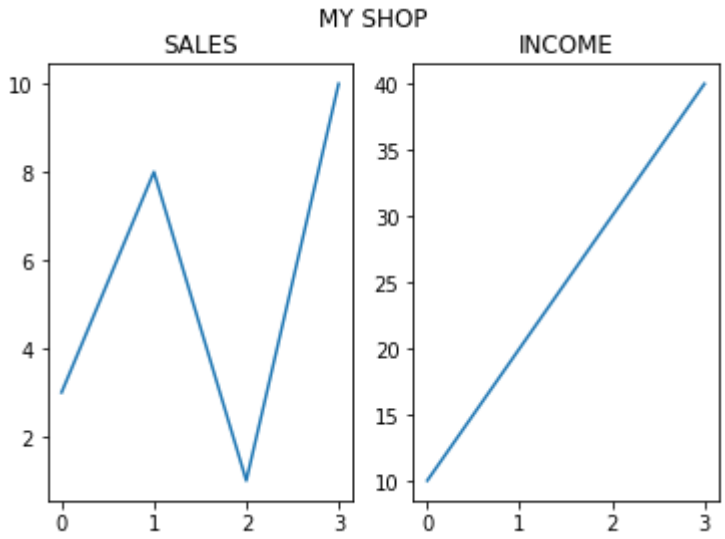
In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot( 2,1, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2,1, 2)
plt.plot(x,y)

plt.show()
```



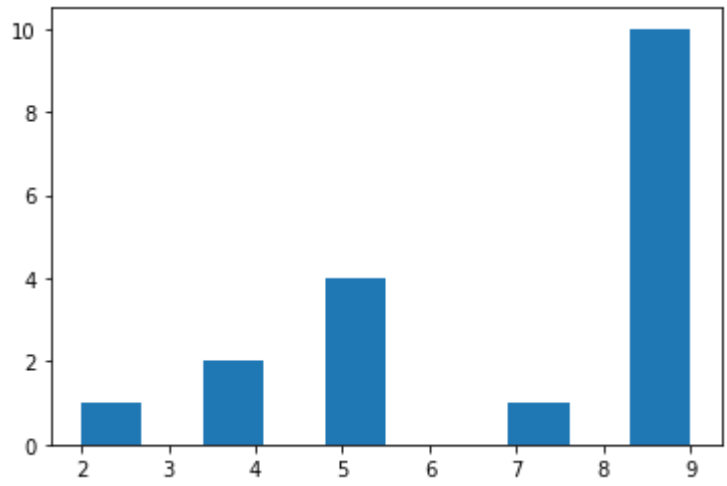
In [ ]:

```
#subtitle and supertitle
import matplotlib.pyplot as plt
import numpy as np
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
plt.plot(x,y)
plt.title("SALES")
#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.title("INCOME")
plt.suptitle("MY SHOP")
plt.show()
```



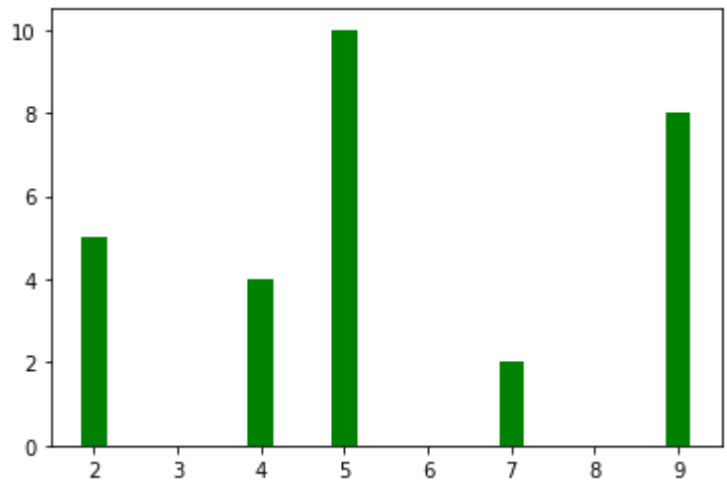
In [ ]:

```
#Creating a histogram
import  matplotlib.pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7,5,5,5,4,9,9,9,9,9,9,9,9,9]
# Function to plot the histogram
plt.hist(x)
# function to show the plot
plt.show()
```



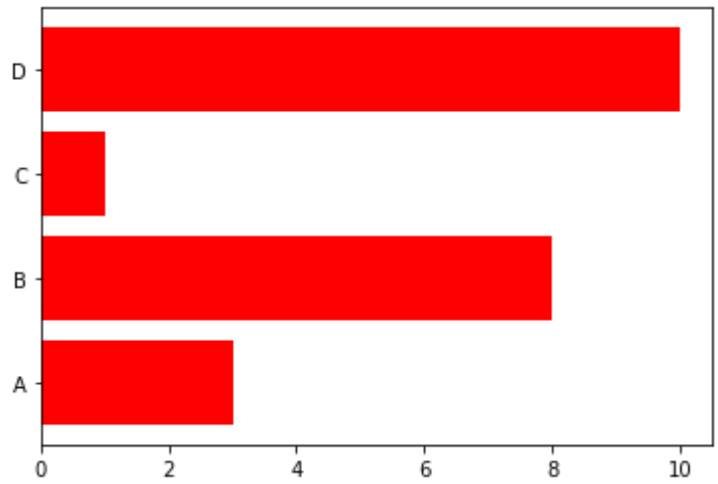
In [ ]:

```
#bar graph with different width
import  matplotlib.pyplot as plt
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]
# Function to plot the bar
plt.bar(x,y,width=0.3,color="green")
# function to show the plot
plt.show()
```



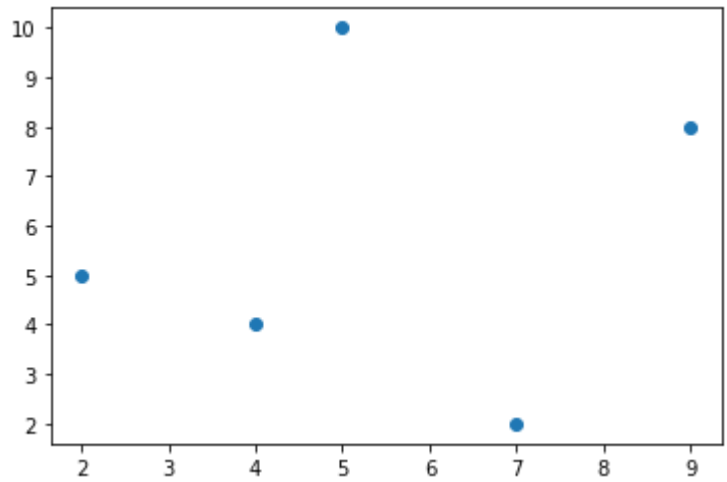
In [ ]:

```
#bar graph along the vertical axis
import matplotlib.pyplot as plt
import numpy as np
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.barh(x, y, color="red")
plt.show()
```



In [ ]:

```
#Scatter Plot
from matplotlib import pyplot as plt
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]
# Function to plot scatter
plt.scatter(x, y)
plt.show()
```

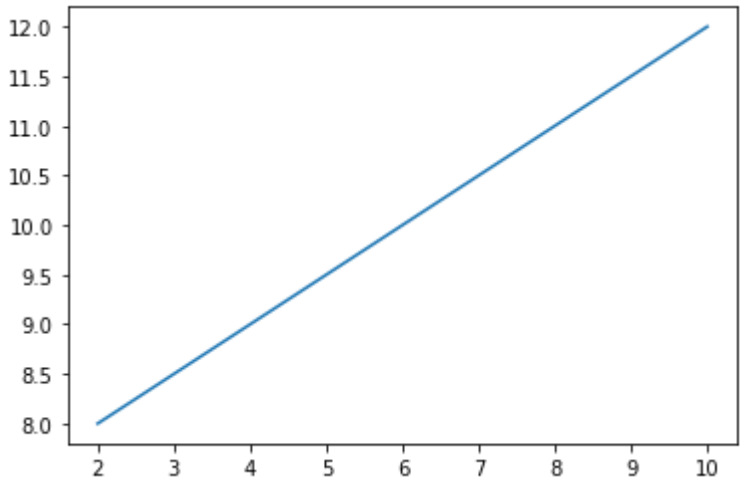


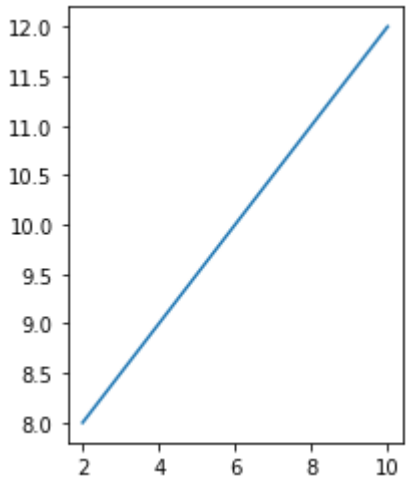
In [ ]:

```
# figure
import matplotlib.pyplot as plt
a = [2, 4, 6, 8, 10]
b = [8, 9, 10, 11, 12]
# Defailt figure size will be shown here
display(plt.plot(a, b))
# Altering the figure size to 3 x 4
plt.figure(figsize = (3, 4))
display(plt.plot(a, b))
```

```
[<matplotlib.lines.Line2D at 0x7f5f8c30d6d0>]
```

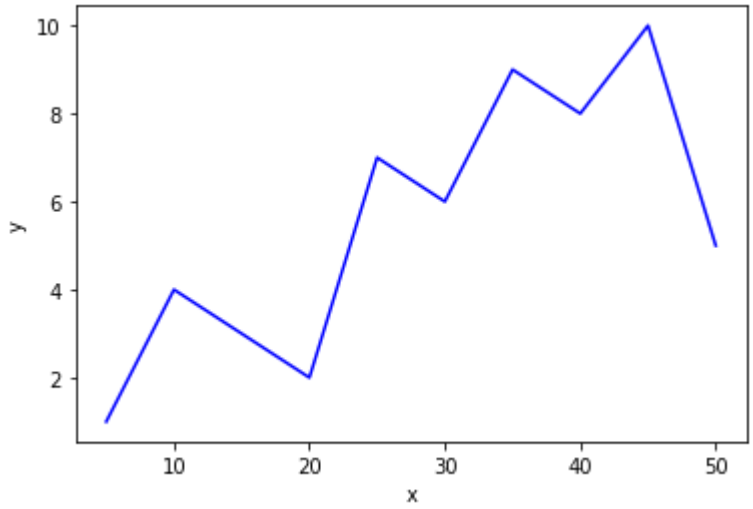
```
[<matplotlib.lines.Line2D at 0x7f5f8c2cd9d0>]
```

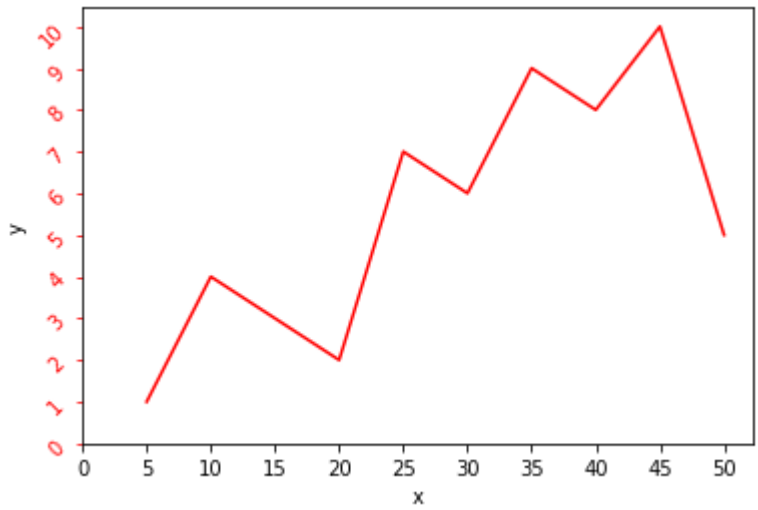




In [ ]:

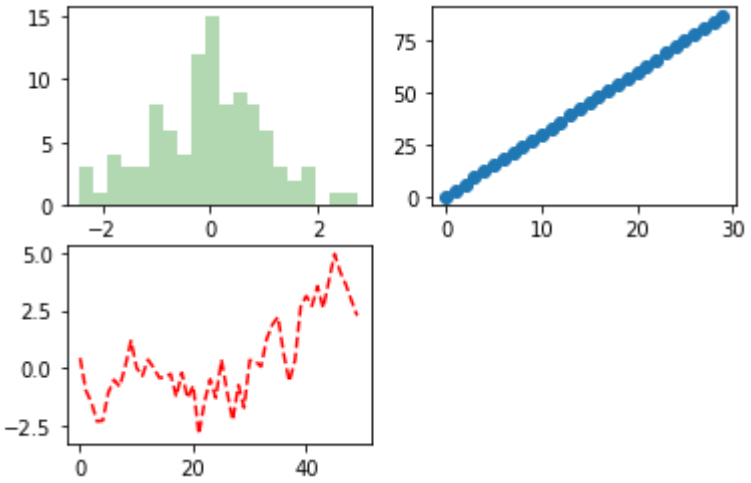
```
#xticks and Yticks
import matplotlib.pyplot as plt
import numpy as np
 # values of x and y axes
x = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
y = [1, 4, 3, 2, 7, 6, 9, 8, 10, 5]
plt.figure(1)
plt.plot(x, y, 'b')
plt.xlabel('x')
plt.ylabel('y')
plt.figure(2)
plt.plot(x, y, 'r')
plt.xlabel('x')
plt.ylabel('y')
plt.xticks(np.arange(0, 51, 5))
plt.yticks(np.arange(0, 11, 1))
plt.tick_params(axis='y',colors='red',
                rotation=45)
plt.show()
```





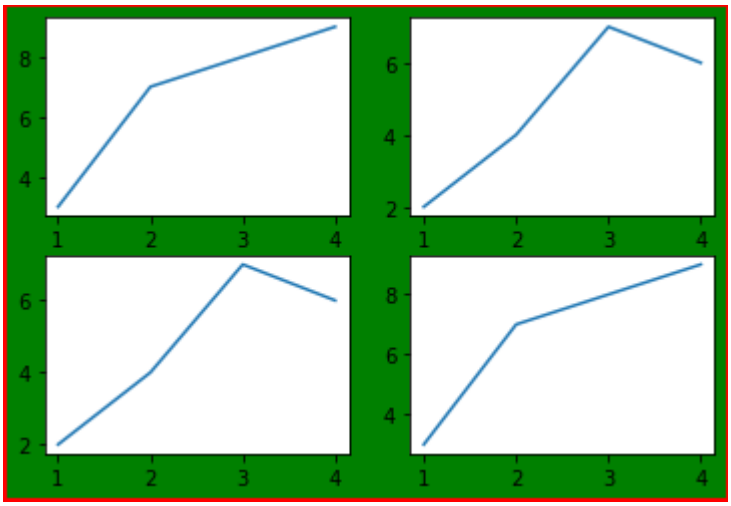
In [ ]:

```
import matplotlib.pyplot as plt
from numpy.random import randn
import numpy as np
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
ax3.plot(randn(50).cumsum(), 'r--')
ax1.hist(randn(100), bins=20, color='g', alpha=0.3)
ax2.scatter(np.arange(30), np.arange(30) + 3 * randn(30))
plt.show()
```



In [ ]:

```
# Adding Subplot to the figure
import matplotlib.pyplot as plt
import numpy as np
t = [1,2,3,4]
s1 = [3,7,8,9]
s2 = [2,4,7,6]
fig=plt.figure(facecolor="green",linewidth=4)# open a new figure
fig.set_edgecolor('red')
plt.subplot(2,2,1)
plt.xticks(t)
plt.plot(t, s1)
# Taking another sub plot
plt.subplot(2,2,2)
plt.plot(t, s2)
plt.xticks(t)
# Taking third sub plot
plt.subplot(2,2,3)
plt.plot(t, s2)
plt.xticks(t)
#taking 4th subplot
plt.subplot(2,2,4)
plt.xticks(t)
plt.plot(t, s1)
plt.show()
```

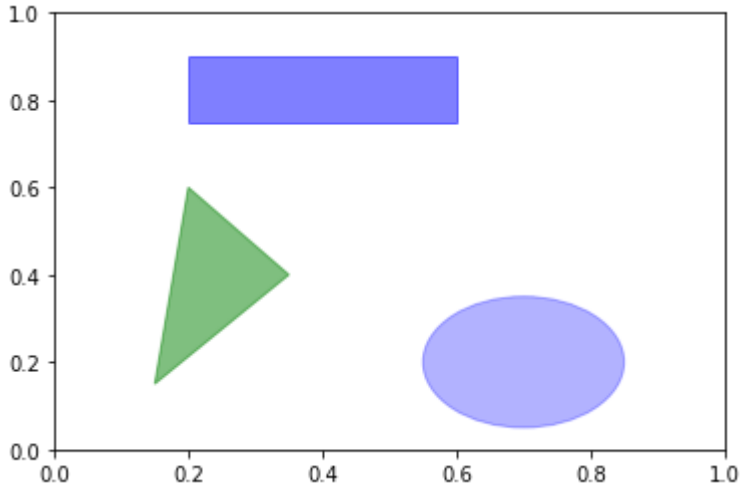


In [ ]:

```
#add_patch() adding triangle rectangle to a figure
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
rect = plt.Rectangle((0.2, 0.75), 0.4, 0.15, color='b', alpha=.5)
circ = plt.Circle((0.7, 0.2), 0.15, color='b', alpha=0.3)
pgon = plt.Polygon([[0.15, 0.15], [0.35, 0.4], [0.2, 0.6]],
color='g', alpha=0.5)
ax.add_patch(rect)
ax.add_patch(circ)
ax.add_patch(pgon)
```

Out[ ]:

```
<matplotlib.patches.Polygon at 0x7f94bcb9ca10>
```



In [ ]:

```
#dataframe from list
import pandas as pd
 # initialize list of lists
data = [["Ohio",2000,1.5],["Ohio",2001,1.7],
        ["Ohio",2002,3.6],
        ["Nevada", 2001, 2.4],["Nevada",2002,2.9]]
  # Create the pandas DataFrame
df = pd.DataFrame(data, columns = ['state',
                                   'year','pop'])
# print dataframe.
print(df)
```

```
    state  year  pop
0    Ohio  2000  1.5
1    Ohio  2001  1.7
2    Ohio  2002  3.6
3  Nevada  2001  2.4
4  Nevada  2002  2.9
```

In [ ]:

```
#Creating dataframe from dictionary
import pandas as pd
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada',
                  'Nevada'],
 'year': [2000, 2001, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
frame = pd.DataFrame(data)
print(frame)
```

```
    state  year  pop
0    Ohio  2000  1.5
1    Ohio  2001  1.7
2    Ohio  2002  3.6
3  Nevada  2001  2.4
4  Nevada  2002  2.9
```

In [ ]:

```
#change allignment of columns
import pandas as pd
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada',
                  'Nevada'],
 'year': [2000, 2001, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
frame = pd.DataFrame(data, columns=['year', 'state',
                                    'pop'])
print(frame)
```

```
   year   state  pop
0  2000    Ohio  1.5
1  2001    Ohio  1.7
2  2002    Ohio  3.6
3  2001  Nevada  2.4
4  2002  Nevada  2.9
```

In [ ]:

```
#given index and a  column debt is inserted
import pandas as pd
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada',
                  'Nevada'],
 'year': [2000, 2001, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
frame2 = pd.DataFrame(data, columns=['year', 'state',
                                     'pop', 'debt'],
                      index=[10,20,30,40,50])

print(frame2)
```

```
    year   state  pop debt
10  2000    Ohio  1.5  NaN
20  2001    Ohio  1.7  NaN
30  2002    Ohio  3.6  NaN
40  2001  Nevada  2.4  NaN
50  2002  Nevada  2.9  NaN
```

In [ ]:

```
#head and tail
print(frame2.head(2))
print("_____________________________")
#tail will display rows from last
print(frame2.tail(2))
```

```
   year state  pop
0  2000  Ohio  1.5
1  2001  Ohio  1.7
_____________________________
   year   state  pop
3  2001  Nevada  2.4
4  2002  Nevada  2.9
```

In [ ]:

```
#retreive row
import pandas as pd
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
 'year': [2000, 2001, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],
                            index=["a","b","c","d","e"])
print(frame2)
print("_____________")
print(frame2.loc["b"])
```

```
   year   state  pop debt
a  2000    Ohio  1.5  NaN
b  2001    Ohio  1.7  NaN
c  2002    Ohio  3.6  NaN
d  2001  Nevada  2.4  NaN
e  2002  Nevada  2.9  NaN
_____________
year     2001
state    Ohio
pop       1.7
debt      NaN
Name: b, dtype: object
```

In [ ]:

```
#retreive column
import pandas as pd
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
 'year': [2000, 2001, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop'])
print(frame2)
print("_____________")
print(frame2["year"])
print("--------------")
print(frame2.year)
print("--------------")
print(frame2[["year","pop"]])
```

```
   year   state  pop
0  2000    Ohio  1.5
1  2001    Ohio  1.7
2  2002    Ohio  3.6
3  2001  Nevada  2.4
4  2002  Nevada  2.9
_____________
0    2000
1    2001
2    2002
3    2001
4    2002
Name: year, dtype: int64
--------------
0    2000
1    2001
2    2002
3    2001
4    2002
Name: year, dtype: int64
--------------
   year  pop
0  2000  1.5
1  2001  1.7
2  2002  3.6
3  2001  2.4
4  2002  2.9
```

In [ ]:

```
#slicing
import pandas as pd
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
 'year': [2000, 2001, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],index=["a","b","c"
,"d","e"])
print(frame2)
print("_____________")
print(frame2.iloc[0:2,2:3])
```

```
   year   state  pop debt
a  2000    Ohio  1.5  NaN
b  2001    Ohio  1.7  NaN
c  2002    Ohio  3.6  NaN
d  2001  Nevada  2.4  NaN
e  2002  Nevada  2.9  NaN
_____________
   pop
a  1.5
b  1.7
```

In [ ]:

```
#Columns can be modified by assignment.
import pandas as pd
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
 'year': [2000, 2001, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],
                      index=['one', 'two', 'three', 'four', 'five'])
frame2['debt'] = 16.5
print(frame2)
```

```
       year   state  pop  debt
one    2000    Ohio  1.5  16.5
two    2001    Ohio  1.7  16.5
three  2002    Ohio  3.6  16.5
four   2001  Nevada  2.4  16.5
five   2002  Nevada  2.9  16.5
```

In [ ]:

```
#assigning an array of values to debt
import pandas as pd
import numpy as np
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
 'year': [2000, 2001, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],
                      index=['one', 'two', 'three', 'four', 'five'])
frame2['debt'] = np.arange(0,5)
print(frame2)
```

```
       year   state  pop  debt
one    2000    Ohio  1.5     0
two    2001    Ohio  1.7     1
three  2002    Ohio  3.6     2
four   2001  Nevada  2.4     3
five   2002  Nevada  2.9     4
```

In [ ]:

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session.
Please rerun this cell to enable.

```
Saving order.csv to order.csv
```

In [ ]:

```
#Selection by label
# Import pandas package
import pandas as pd
# making data frame from csv file
data = pd.read_csv("/content/order.csv")
print(data)
data = pd.read_csv("/content/order.csv",index_col="Item")
print(data)
```

slno	OrderDate	Region	Rep	Item	Units	Unit Cost	Total
1	1-6-21	East	Jones	Pencil	95	1.99	189.05
2	1-23-21	Central	Kivell	Binder	50	19.99	999.50
3	2-9-21	Central	Jardine	Pencil	36	4.99	179.64
4	2-26-21	Central	Gill	Pen	27	19.99	539.73
5	3-15-21	West	Sorvino	Pencil	56	2.99	167.44
6	4-1-21	East	Jones	Binder	60	4.99	299.40
7	4-18-21	Central	Andrews	Pencil	75	1.99	149.25
8	5-5-21	Central	Jardine	Pencil	90	4.99	449.10
9	5-22-21	West	Thompson	Pencil	32	1.99	63.68
10	6-8-21	East	Jones	Binder	60	8.99	539.40
11	6-25-21	Central	Morgan	Pencil	90	4.99	449.10
12	7-12-21	East	Howard	Binder	29	1.99	57.71
13	7-29-21	East	Parent	Binder	81	19.99	1,619.19
14	8-15-21	East	Jones	Pencil	35	4.99	174.65
15	9-1-21	Central	Smith	Desk	2	125.00	250.00
16	9-18-21	East	Jones	Pen Set	16	15.99	255.84
17	10-5-21	Central	Morgan	Binder	28	8.99	251.72
18	10-22-21	East	Jones	Pen	64	8.99	575.36
19	11-8-21	East	Parent	Pen	15	19.99	299.85
20	11-25-21	Central	Kivell	Pen Set	96	4.99	479.04
21	12-12-21	Central	Smith	Pencil	67	1.29	86.43
22	12-29-21	East	Parent	Pen Set	74	15.99	1,183.26
23	1-15-22	Central	Gill	Binder	46	8.99	413.54
24	2-1-22	Central	Smith	Binder	87	15.00	1,305.00
25	2-18-22	East	Jones	Binder	4	4.99	19.96
26	3-7-22	West	Sorvino	Binder	7	19.99	139.93
27	3-24-22	Central	Jardine	Pen Set	50	4.99	249.50
28	4-10-22	Central	Andrews	Pencil	66	1.99	131.34
29	4-27-22	East	Howard	Pen	96	4.99	479.04
30	5-14-22	Central	Gill	Pencil	53	1.29	68.37

7
30 31 5-31-22 Central Gill Binder 80 8.99 719.2
0
31 32 6-17-22 Central Kivell Desk 5 125.00 625.0
0
32 33 7-4-22 East Jones Pen Set 62 4.99 309.3
8
33 34 7-21-22 Central Morgan Pen Set 55 12.49 686.9
5
34 35 8-7-22 Central Kivell Pen Set 42 23.95 1,005.9
0
35 36 8-24-22 West Sorvino Desk 3 275.00 825.0
0
36 37 9-10-22 Central Gill Pencil 7 1.29 9.0
3
37 38 9-27-22 West Sorvino Pen 76 1.99 151.2
4
38 39 10-14-22 West Thompson Binder 57 19.99 1,139.4
3
39 40 10-31-22 Central Andrews Pencil 14 1.29 18.0
6
40 41 11-17-22 Central Jardine Binder 11 4.99 54.8
9
41 42 12-4-22 Central Jardine Binder 94 19.99 1,879.0
6
42 43 12-21-22 Central Andrews Binder 28 4.99 139.7
2

Item	slno	OrderDate	Region	Rep	Units	Unit Cost	Total
Pencil	1	1-6-21	East	Jones	95	1.99	189.05
Binder	2	1-23-21	Central	Kivell	50	19.99	999.50
Pencil	3	2-9-21	Central	Jardine	36	4.99	179.64
Pen	4	2-26-21	Central	Gill	27	19.99	539.73
Pencil	5	3-15-21	West	Sorvino	56	2.99	167.44
Binder	6	4-1-21	East	Jones	60	4.99	299.40
Pencil	7	4-18-21	Central	Andrews	75	1.99	149.25
Pencil	8	5-5-21	Central	Jardine	90	4.99	449.10
Pencil	9	5-22-21	West	Thompson	32	1.99	63.68
Binder	10	6-8-21	East	Jones	60	8.99	539.40
Pencil	11	6-25-21	Central	Morgan	90	4.99	449.10
Binder	12	7-12-21	East	Howard	29	1.99	57.71
Binder	13	7-29-21	East	Parent	81	19.99	1,619.19
Pencil	14	8-15-21	East	Jones	35	4.99	174.65
Desk	15	9-1-21	Central	Smith	2	125.00	250.00
Pen Set	16	9-18-21	East	Jones	16	15.99	255.84
Binder	17	10-5-21	Central	Morgan	28	8.99	251.72
Pen	18	10-22-21	East	Jones	64	8.99	575.36
Pen	19	11-8-21	East	Parent	15	19.99	299.85
Pen Set	20	11-25-21	Central	Kivell	96	4.99	479.04
Pencil	21	12-12-21	Central	Smith	67	1.29	86.43
Pen Set	22	12-29-21	East	Parent	74	15.99	1,183.26
Binder	23	1-15-22	Central	Gill	46	8.99	413.54
Binder	24	2-1-22	Central	Smith	87	15.00	1,305.00
Binder	25	2-18-22	East	Jones	4	4.99	19.96
Binder	26	3-7-22	West	Sorvino	7	19.99	139.93
Pen Set	27	3-24-22	Central	Jardine	50	4.99	249.50
Pencil	28	4-10-22	Central	Andrews	66	1.99	131.34
Pen	29	4-27-22	East	Howard	96	4.99	479.04
Pencil	30	5-14-22	Central	Gill	53	1.29	68.37
Binder	31	5-31-22	Central	Gill	80	8.99	719.20
Desk	32	6-17-22	Central	Kivell	5	125.00	625.00

Pen Set	33	7-4-22	East	Jones	62	4.99	309.38
Pen Set	34	7-21-22	Central	Morgan	55	12.49	686.95
Pen Set	35	8-7-22	Central	Kivell	42	23.95	1,005.90
Desk	36	8-24-22	West	Sorvino	3	275.00	825.00
Pencil	37	9-10-22	Central	Gill	7	1.29	9.03
Pen	38	9-27-22	West	Sorvino	76	1.99	151.24
Binder	39	10-14-22	West	Thompson	57	19.99	1,139.43
Pencil	40	10-31-22	Central	Andrews	14	1.29	18.06
Binder	41	11-17-22	Central	Jardine	11	4.99	54.89
Binder	42	12-4-22	Central	Jardine	94	19.99	1,879.06
Binder	43	12-21-22	Central	Andrews	28	4.99	139.72

In [ ]:

```
#Selection by label
# Import pandas package
import pandas as pd
# making data frame from csv file
data = pd.read_csv("/content/order.csv",
                   index_col ="slno")
print("_____________________________")
print("retrieving row by loc method")
print(data.loc[2])
print("_____________________________")
print(data.loc[:, ["OrderDate", "Units"] ])
print("_____________________________")
print(data.loc[1:3, ["OrderDate", "Units"] ])
print("_____________________________")
print(data.loc[1:5:2, : ])
print("----------------------")
```

```
_____________________________
retrieving row by loc method
OrderDate    1-23-21
Region       Central
Rep           Kivell
Item          Binder
Units             50
Unit Cost      19.99
Total          999.5
Name: 2, dtype: object
_____________________________
       OrderDate  Units
slno
1     01-06-2021     95
2        1-23-21     50
3     02-09-2021     36
4        2-26-21     27
5        3-15-21     56
6     04-01-2021     60
7        4-18-21     75
8     05-05-2021     90
9        5-22-21     32
10    06-08-2021     60
11       6-25-21     90
12    07-12-2021     29
13       7-29-21     81
14       8-15-21     35
15    09-01-2021      2
16       9-18-21     16
17    10-05-2021     28
18      10-22-21     64
19    11-08-2021     15
20      11-25-21     96
21    12-12-2021     67
22      12-29-21     74
23       1-15-22     46
24    02-01-2022     87
25       2-18-22      4
26    03-07-2022      7
27       3-24-22     50
28    04-10-2022     66
29       4-27-22     96
30       5-14-22     53
31       5-31-22     80
32       6-17-22      5
33    07-04-2022     62
34       7-21-22     55
35    08-07-2022     42
36       8-24-22      3
37    09-10-2022      7
38       9-27-22     76
39      10-14-22     57
40      10-31-22     14
41      11-17-22     11
42    12-04-2022     94
43      12-21-22     28
_____________________________
       OrderDate  Units
slno
1     01-06-2021     95
2        1-23-21     50
```

```
3     02-09-2021     36
```

```
       OrderDate   Region      Rep    Item  Units  Unit Cost   Total
slno
1     01-06-2021     East    Jones  Pencil     95       1.99  189.05
3     02-09-2021  Central  Jardine  Pencil     36       4.99  179.64
5        3-15-21     West  Sorvino  Pencil     56       2.99  167.44
```

In [ ]:

```
#Selection by label
# Import pandas package
import pandas as pd
# making data frame from csv file
data = pd.read_csv("/content/ord.csv",index_col ="slno")

#print("all rows")
print(data.loc[:])
print("-------------------------")
#print(all rows and 2 columns)
print(data.loc[:,["Region","Item"]])
print("-------------------------")
#print(from 3rd label)
print(data.loc[5:,["Region","Item"]])
print("-------------------------")
```

```
       OrderDate   Region      Rep    Item  Units  Unit Cost
slno
1     01-06-2021     East    Jones  Pencil     95       1.99
2        1-23-21  Central   Kivell  Binder     50      19.99
3     02-09-2021  Central  Jardine  Pencil     36       4.99
4        2-26-21  Central     Gill     Pen     27      19.99
5        3-15-21     West  Sorvino  Pencil     56       2.99
6     04-01-2021     East    Jones  Binder     60       4.99
7        4-18-21  Central  Andrews  Pencil     75       1.99
8     05-05-2021  Central  Jardine  Pencil     90       4.99
-------------------------
       Region    Item
slno
1        East  Pencil
2     Central  Binder
3     Central  Pencil
4     Central     Pen
5        West  Pencil
6        East  Binder
7     Central  Pencil
8     Central  Pencil
-------------------------
       Region    Item
slno
5        West  Pencil
6        East  Binder
7     Central  Pencil
8     Central  Pencil
-------------------------
```

In [ ]:

```
#missing values
import pandas as pd
import numpy as np
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
 'year': [2000, np.nan, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, np.nan, 2.9]}
frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop',np.nan],
                      index=['one', 'two', 'three', 'four', 'five'])
df=pd.DataFrame(data)
print(df)
print("----------------")
print("is null")
print(df.isnull())
print("_____________")
print("not null")
print(df.notnull())
print("-----------------")
# filling missing value using fillna()
print(df.fillna(0))
print("____________________")
print("filling missing value using mean value()")
print(df.fillna(df.mean()))
print("____________________")
#filling the NaN values by interpolation
print(df.interpolate())
print("-----------------------")
#replace missing values with -1
print(df.replace(np.nan,-1))
print("-------------------------")
```

```
    state    year  pop
0    Ohio  2000.0  1.5
1    Ohio     NaN  1.7
2    Ohio  2002.0  3.6
3  Nevada  2001.0  NaN
4  Nevada  2002.0  2.9
----------------
is null
   state   year    pop
0  False  False  False
1  False   True  False
2  False  False  False
3  False  False   True
4  False  False  False
_______________
not null
   state   year    pop
0   True   True   True
1   True  False   True
2   True   True   True
3   True   True  False
4   True   True   True
-----------------
    state    year  pop
0    Ohio  2000.0  1.5
1    Ohio     0.0  1.7
2    Ohio  2002.0  3.6
3  Nevada  2001.0  0.0
4  Nevada  2002.0  2.9
____________________
filling missing value using mean value()
    state     year    pop
0    Ohio  2000.00  1.500
1    Ohio  2001.25  1.700
2    Ohio  2002.00  3.600
3  Nevada  2001.00  2.425
4  Nevada  2002.00  2.900
____________________
    state    year   pop
0    Ohio  2000.0  1.50
1    Ohio  2001.0  1.70
2    Ohio  2002.0  3.60
3  Nevada  2001.0  3.25
4  Nevada  2002.0  2.90
-----------------------
    state    year  pop
0    Ohio  2000.0  1.5
1    Ohio    -1.0  1.7
2    Ohio  2002.0  3.6
3  Nevada  2001.0 -1.0
4  Nevada  2002.0  2.9
-------------------------

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: FutureWar
ning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_
only=None') is deprecated; in a future version this will raise TypeError.
Select only valid columns before calling the reduction.
```

In [ ]:

```
#selection by index
# Import pandas package
import pandas as pd
# making data frame from csv file
data = pd.read_csv("/content/order.csv", index_col ="Region")
print("_____________________________")
print(data.iloc[1])
print("_____________________")
print("row from 0 to 2 and columns from 0 and 1")
print(data.iloc[0:3,[0,2]])
print("_____________________")
print(data.iloc[0:3,0:2])
print("_____________________")
print("row with index 1 2 and 4,column with 0 1 and 2 ")
print(data.iloc[[1,2,4],[0,2]])
```

```
_____________________________
slno                 2
OrderDate     1-23-21
Rep            Kivell
Item           Binder
Units              50
Unit Cost       19.99
Total          999.50
Name: Central, dtype: object
_____________________
row from 0 to 2 and columns from 0 and 1
         slno      Rep
Region
East        1    Jones
Central     2   Kivell
Central     3  Jardine
_____________________
         slno OrderDate
Region
East        1    1-6-21
Central     2   1-23-21
Central     3    2-9-21
_____________________
row with index 1 2 and 4,column with 0 1 and 2
         slno      Rep
Region
Central     2   Kivell
Central     3  Jardine
West        5  Sorvino
```

In [ ]:

```
#dropping the rows containing null values
print(df.dropna())
```

```
    state    year  pop
0    Ohio  2000.0  1.5
2    Ohio  2002.0  3.6
4  Nevada  2002.0  2.9
```

In order to iterate over rows, we can use three function iteritems(), iterrows(), itertuples() .

In [ ]:

```
import pandas as pd
# dictionary of lists
dict = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
 'year': [2000, np.nan, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, np.nan, 2.9]}
# creating a dataframe from a dictionary
df = pd.DataFrame(dict)
print(df)
for i in df.itertuples(): # this will get each row as a tuple
    print(i)
    print()
```

```
    state    year  pop
0    Ohio  2000.0  1.5
1    Ohio     NaN  1.7
2    Ohio  2002.0  3.6
3  Nevada  2001.0  NaN
4  Nevada  2002.0  2.9
Pandas(Index=0, state='Ohio', year=2000.0, pop=1.5)

Pandas(Index=1, state='Ohio', year=nan, pop=1.7)

Pandas(Index=2, state='Ohio', year=2002.0, pop=3.6)

Pandas(Index=3, state='Nevada', year=2001.0, pop=nan)

Pandas(Index=4, state='Nevada', year=2002.0, pop=2.9)
```

In [ ]:

```
#iterrows and iteritems
import pandas as pd
# dictionary of lists
dict = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
 'year': [2000, np.nan, 2002, 2001, 2002],
 'pop': [1.5, 1.7, 3.6, np.nan, 2.9]}
# creating a dataframe from a dictionary
df = pd.DataFrame(dict)
print("iterrows")
for i,j in df.iterrows(): # this will get each index and each row values
    print(i,j)
    print("_____________________")
print("iteritems")
for i,j in df.iteritems():# this will extract each field seperately
    print(i,j)
    print("----------------------")
```

```
iterrows
0 state      Ohio
year     2000.0
pop         1.5
Name: 0, dtype: object
__________________
1 state    Ohio
year      NaN
pop       1.7
Name: 1, dtype: object
__________________
2 state      Ohio
year     2002.0
pop         3.6
Name: 2, dtype: object
__________________
3 state    Nevada
year     2001.0
pop         NaN
Name: 3, dtype: object
__________________
4 state    Nevada
year     2002.0
pop         2.9
Name: 4, dtype: object
__________________
iteritems
state 0      Ohio
1      Ohio
2      Ohio
3    Nevada
4    Nevada
Name: state, dtype: object
----------------------
year 0    2000.0
1       NaN
2    2002.0
3    2001.0
4    2002.0
Name: year, dtype: float64
----------------------
pop 0    1.5
1    1.7
2    3.6
3    NaN
4    2.9
Name: pop, dtype: float64
----------------------
```

In [ ]:

```
Write Python program to write the data
given below to a CSV file.(university question)
SN Name                          Country          Contribution              Year
1     Linus Torvalds          Finland         Linux Kernel              1991
2     Tim Berners-Lee       England        World Wide Web       1990
3     Guido van Rossum    Netherlands   Python                        1991
```

Write Python program to write the data given below to a CSV file.(university question) SN Name Country Contribution Year 1 Linus Torvalds Finland Linux Kernel 1991 2 Tim Berners-Lee England World Wide Web 1990 3 Guido van Rossum Netherlands Python 1991

In [ ]:

```
# importing pandas as pd
import pandas as pd
# dictionary of lists
# creating a dataframe from a dictionary
df = pd.DataFrame([[1,' Linus Torvalds','Finland','Linux Kernel ',1991],
      [2,'Tim Berners-Lee','England','World Wide Web',1990],
      [3,'Guido van Rossum','Netherlands','Python',1991]],
columns=['SN','Name','Country','Contribution','Year'])
print("data frame with defaut index=",df)
df=df.set_index('SN')
print("data frame with SN as index=",df)
print(df)
df.to_csv('invenrs.csv')
```

```
data frame with defaut index=    SN              Name      Country    Cont
ribution  Year
0   1    Linus Torvalds      Finland   Linux Kernel   1991
1   2   Tim Berners-Lee      England  World Wide Web  1990
2   3  Guido van Rossum  Netherlands          Python  1991
data frame with SN as index=                 Name      Country    Contribu
tion  Year
SN
1     Linus Torvalds      Finland   Linux Kernel   1991
2    Tim Berners-Lee      England  World Wide Web  1990
3   Guido van Rossum  Netherlands          Python  1991
                Name      Country    Contribution  Year
SN
1     Linus Torvalds      Finland   Linux Kernel   1991
2    Tim Berners-Lee      England  World Wide Web  1990
3   Guido van Rossum  Netherlands          Python  1991
```

Given a file “auto.csv” of automobile data with the fields index, company, body-style, wheel-base, length, engine-type, num-of-cylinders, horsepower, average-mileage, and price, write Python codes using Pandas to read the csv file and do the following 1) From the given dataset print the first and last five rows

In [ ]:

```
import pandas as pd
df = pd.read_csv("/content/Automobile_data.csv")
print(df.head(5))
print("last rows")
print(df.tail(5))
```

```
   index      company   body-style  wheel-base  length engine-type  \
0      0  alfa-romero  convertible        88.6   168.8        dohc
1      1  alfa-romero  convertible        88.6   168.8        dohc
2      2  alfa-romero    hatchback        94.5   171.2        ohcv
3      3         audi        sedan        99.8   176.6         ohc
4      4         audi        sedan        99.4   176.6         ohc

  num-of-cylinders  horsepower  average-mileage    price
0             four         111               21  13495.0
1             four         111               21  16500.0
2              six         154               19  16500.0
3             four         102               24  13950.0
4             five         115               18  17450.0
last rows
    index      company body-style  wheel-base  length engine-type  \
56     81  volkswagen      sedan        97.3   171.7         ohc
57     82  volkswagen      sedan        97.3   171.7         ohc
58     86  volkswagen      sedan        97.3   171.7         ohc
59     87       volvo      sedan       104.3   188.8         ohc
60     88       volvo      wagon       104.3   188.8         ohc

   num-of-cylinders  horsepower  average-mileage    price
56             four          85               27   7975.0
57             four          52               37   7995.0
58             four         100               26   9995.0
59             four         114               23  12940.0
60             four         114               23  13415.0
```

Clean the dataset and update the CSV file Replace all column values which contain ?, n.a, or NaN.

In [ ]:

```
import pandas as pd
df = pd.read_csv("/content/Automobile_data.csv",na_values={
'price':["?","n.a",'NaN'," "],
'stroke':["?","n.a",'NaN'," "],
'horsepower':["?","n.a",'NaN'],
'peak-rpm':["?","n.a",'NaN',' '],
'average-mileage':["?","n.a",'NaN']})
DF2=df.fillna(0)
DF2.to_csv("/content/Automobil4.csv")
```

In [ ]:

```
import pandas as pd
df = pd.read_csv("/content/Automobile_data.csv")
DF2=df.fillna(0)
DF2.to_csv("/content/Automobil5.csv")
```

In [ ]:

```
#Find the most expensive car company name
import pandas as pd
df = pd.read_csv("/content/Automobile_data.csv")
company = df [['company','price']][df.price==df['price'].max()]
print(company)
```

```
          company    price
35  mercedes-benz  45400.0
```

In [ ]:

```
#Find the most expensive car company name
import pandas as pd
df = pd.read_csv("Automobile_data.csv")
DF1=df.sort_values(by=['price'],
ascending=False)[['company','price']]
DF1.head(1)
```

Out[ ]:

	company	price
35	mercedes-benz	45400.0

In [ ]:

```
#maximum price of each companies
import pandas as pd
df = pd.read_csv("/content/Automobile_data.csv")
df1=df.groupby('company')['body-style','horsepower','price'].max()
print(df1)
```

```
              body-style  horsepower    price
company
alfa-romero    hatchback         154  16500.0
audi               wagon         115  18920.0
bmw                sedan         182  41315.0
chevrolet          sedan          70   6575.0
dodge          hatchback          68   6377.0
honda              wagon         101  12945.0
isuzu              sedan          78   6785.0
jaguar             sedan         262  36000.0
mazda              sedan         101  18344.0
mercedes-benz      wagon         184  45400.0
mitsubishi         sedan          88   8189.0
nissan             wagon         152  13499.0
porsche        hatchback         288  37028.0
toyota             wagon         156  15750.0
volkswagen         sedan         100   9995.0
volvo              wagon         114  13415.0

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: FutureWarn
ing: Indexing with multiple keys (implicitly converted to a tuple of keys)
will be deprecated, use a list instead.
  after removing the cwd from sys.path.
```

In [ ]:

```
# Print All Toyota Cars details
import pandas as pd
df = pd.read_csv("/content/Automobile_data.csv")
company =print(df[df['company']=='toyota'])
print(company)
```

```
    index company body-style  wheel-base  length engine-type num-of-cylind
ers  \
48     66  toyota  hatchback        95.7   158.7         ohc             f
our
49     67  toyota  hatchback        95.7   158.7         ohc             f
our
50     68  toyota  hatchback        95.7   158.7         ohc             f
our
51     69  toyota      wagon        95.7   169.7         ohc             f
our
52     70  toyota      wagon        95.7   169.7         ohc             f
our
53     71  toyota      wagon        95.7   169.7         ohc             f
our
54     79  toyota      wagon       104.5   187.8        dohc
six

    horsepower  average-mileage    price
48          62               35   5348.0
49          62               31   6338.0
50          62               31   6488.0
51          62               31   6918.0
52          62               27   7898.0
53          62               27   8778.0
54         156               19  15750.0
None
```

In [ ]:

```
import pandas as pd
df = pd.read_csv("Automobile_data.csv")
car_Manufacturers = df.groupby('company')
toyotaDf = car_Manufacturers.get_group('toyota')
print(toyotaDf)
```

```
    index company body-style  wheel-base  length engine-type num-of-cylind
ers  \
48     66  toyota  hatchback        95.7   158.7         ohc             f
our
49     67  toyota  hatchback        95.7   158.7         ohc             f
our
50     68  toyota  hatchback        95.7   158.7         ohc             f
our
51     69  toyota      wagon        95.7   169.7         ohc             f
our
52     70  toyota      wagon        95.7   169.7         ohc             f
our
53     71  toyota      wagon        95.7   169.7         ohc             f
our
54     79  toyota      wagon       104.5   187.8        dohc
six

    horsepower  average-mileage    price
48          62               35   5348.0
49          62               31   6338.0
50          62               31   6488.0
51          62               31   6918.0
52          62               27   7898.0
53          62               27   8778.0
54         156               19  15750.0
```

In [ ]:

```
#Print total cars of all companies
import pandas as pd
df = pd.read_csv("Automobile_data.csv")
df1=df.groupby('company')
for company,company_df in df1:
  print(company,company_df)
```

```
alfa-romero    index      company   body-style  wheel-base  length engine-
type  \
0      0  alfa-romero  convertible        88.6   168.8        dohc
1      1  alfa-romero  convertible        88.6   168.8        dohc
2      2  alfa-romero    hatchback        94.5   171.2        ohcv

  num-of-cylinders  horsepower  average-mileage    price
0             four         111               21  13495.0
1             four         111               21  16500.0
2              six         154               19  16500.0
audi    index company body-style  wheel-base  length engine-type num-of-cy
linders  \
3      3    audi      sedan        99.8   176.6         ohc             fo
ur
4      4    audi      sedan        99.4   176.6         ohc             fi
ve
5      5    audi      sedan        99.8   177.3         ohc             fi
ve
6      6    audi      wagon       105.8   192.7         ohc             fi
ve

   horsepower  average-mileage    price
3         102               24  13950.0
4         115               18  17450.0
5         110               19  15250.0
6         110               19  18920.0
bmw     index company body-style  wheel-base  length engine-type num-of-cy
linders  \
7       9     bmw      sedan       101.2   176.8         ohc              f
our
8      10     bmw      sedan       101.2   176.8         ohc              f
our
9      11     bmw      sedan       101.2   176.8         ohc
six
10     13     bmw      sedan       103.5   189.0         ohc
six
11     14     bmw      sedan       103.5   193.8         ohc
six
12     15     bmw      sedan       110.0   197.0         ohc
six

    horsepower  average-mileage    price
7          101               23  16430.0
8          101               23  16925.0
9          121               21  20970.0
10         182               16  30760.0
11         182               16  41315.0
12         182               15  36880.0
chevrolet     index    company body-style  wheel-base  length engine-type
\
13     16  chevrolet  hatchback        88.4   141.1           l
14     17  chevrolet  hatchback        94.5   155.9         ohc
15     18  chevrolet      sedan        94.5   158.8         ohc

   num-of-cylinders  horsepower  average-mileage   price
13            three          48               47  5151.0
14             four          70               38  6295.0
15             four          70               38  6575.0
dodge     index company body-style  wheel-base  length engine-type num-of-
cylinders  \
16     19   dodge  hatchback        93.7   157.3         ohc              f
```

```
our
17     20   dodge  hatchback        93.7   157.3         ohc             f
our

    horsepower  average-mileage   price
16          68               31  6377.0
17          68               31  6229.0
honda     index company body-style  wheel-base  length engine-type num-of-
cylinders  \
18     27   honda      wagon        96.5   157.1         ohc             f
our
19     28   honda      sedan        96.5   175.4         ohc             f
our
20     29   honda      sedan        96.5   169.1         ohc             f
our

    horsepower  average-mileage    price
18          76               30   7295.0
19         101               24  12945.0
20         100               25  10345.0
isuzu     index company body-style  wheel-base  length engine-type num-of-
cylinders  \
21     30   isuzu      sedan        94.3   170.7         ohc             f
our
22     31   isuzu      sedan        94.5   155.9         ohc             f
our
23     32   isuzu      sedan        94.5   155.9         ohc             f
our

    horsepower  average-mileage   price
21          78               24  6785.0
22          70               38     NaN
23          70               38     NaN
jaguar     index company body-style  wheel-base  length engine-type num-of
-cylinders  \
24     33  jaguar      sedan       113.0   199.6        dohc
six
25     34  jaguar      sedan       113.0   199.6        dohc
six
26     35  jaguar      sedan       102.0   191.7        ohcv           twe
lve

    horsepower  average-mileage    price
24         176               15  32250.0
25         176               15  35550.0
26         262               13  36000.0
mazda     index company body-style  wheel-base  length engine-type num-of-
cylinders  \
27     36   mazda  hatchback        93.1   159.1         ohc             f
our
28     37   mazda  hatchback        93.1   159.1         ohc             f
our
29     38   mazda  hatchback        93.1   159.1         ohc             f
our
30     39   mazda  hatchback        95.3   169.0       rotor
two
31     43   mazda      sedan       104.9   175.0         ohc             f
our

    horsepower  average-mileage    price
27          68               30   5195.0
```

```
28          68               31   6095.0
29          68               31   6795.0
30         101               17  11845.0
31          72               31  18344.0
mercedes-benz     index        company body-style  wheel-base  length engi
ne-type  \
32     44  mercedes-benz      sedan       110.0   190.9         ohc
33     45  mercedes-benz      wagon       110.0   190.9         ohc
34     46  mercedes-benz      sedan       120.9   208.1        ohcv
35     47  mercedes-benz    hardtop       112.0   199.2        ohcv

   num-of-cylinders  horsepower  average-mileage    price
32             five         123               22  25552.0
33             five         123               22  28248.0
34            eight         184               14  40960.0
35            eight         184               14  45400.0
mitsubishi     index     company body-style  wheel-base  length engine-typ
e  \
36     49  mitsubishi  hatchback        93.7   157.3         ohc
37     50  mitsubishi  hatchback        93.7   157.3         ohc
38     51  mitsubishi      sedan        96.3   172.4         ohc
39     52  mitsubishi      sedan        96.3   172.4         ohc

   num-of-cylinders  horsepower  average-mileage   price
36             four          68               37  5389.0
37             four          68               31  6189.0
38             four          88               25  6989.0
39             four          88               25  8189.0
nissan     index company body-style  wheel-base  length engine-type num-of
-cylinders  \
40     53  nissan      sedan        94.5   165.3         ohc             f
our
41     54  nissan      sedan        94.5   165.3         ohc             f
our
42     55  nissan      sedan        94.5   165.3         ohc             f
our
43     56  nissan      wagon        94.5   170.2         ohc             f
our
44     57  nissan      sedan       100.4   184.6        ohcv
six

    horsepower  average-mileage    price
40          55               45   7099.0
41          69               31   6649.0
42          69               31   6849.0
43          69               31   7349.0
44         152               19  13499.0
porsche     index  company   body-style  wheel-base  length engine-type  \
45     61  porsche      hardtop        89.5   168.9        ohcf
46     62  porsche  convertible        89.5   168.9        ohcf
47     63  porsche    hatchback        98.4   175.7       dohcv

   num-of-cylinders  horsepower  average-mileage    price
45              six         207               17  34028.0
46              six         207               17  37028.0
47            eight         288               17      NaN
toyota     index company body-style  wheel-base  length engine-type num-of
-cylinders  \
48     66  toyota  hatchback        95.7   158.7         ohc             f
our
49     67  toyota  hatchback        95.7   158.7         ohc             f
```

```
our
50     68  toyota  hatchback        95.7   158.7         ohc             f
our
51     69  toyota      wagon        95.7   169.7         ohc             f
our
52     70  toyota      wagon        95.7   169.7         ohc             f
our
53     71  toyota      wagon        95.7   169.7         ohc             f
our
54     79  toyota      wagon       104.5   187.8        dohc
six

    horsepower  average-mileage    price
48          62               35   5348.0
49          62               31   6338.0
50          62               31   6488.0
51          62               31   6918.0
52          62               27   7898.0
53          62               27   8778.0
54         156               19  15750.0
volkswagen     index     company body-style  wheel-base  length engine-typ
e  \
55     80  volkswagen      sedan        97.3   171.7         ohc
56     81  volkswagen      sedan        97.3   171.7         ohc
57     82  volkswagen      sedan        97.3   171.7         ohc
58     86  volkswagen      sedan        97.3   171.7         ohc

   num-of-cylinders  horsepower  average-mileage   price
55             four          52               37  7775.0
56             four          85               27  7975.0
57             four          52               37  7995.0
58             four         100               26  9995.0
volvo     index company body-style  wheel-base  length engine-type num-of-
cylinders  \
59     87   volvo      sedan       104.3   188.8         ohc             f
our
60     88   volvo      wagon       104.3   188.8         ohc             f
our

    horsepower  average-mileage    price
59         114               23  12940.0
60         114               23  13415.0
```

In [ ]:

```
#Sort all cars by Price column
import pandas as pd
df = pd.read_csv("Automobile_data.csv")
df.sort_values(by=['price'],
ascending=False)[['company','price']]
```

Out[ ]:

	company	price
35	mercedes-benz	45400.0
11	bmw	41315.0
34	mercedes-benz	40960.0
46	porsche	37028.0
12	bmw	36880.0
...	...	...
27	mazda	5195.0
13	chevrolet	5151.0
22	isuzu	NaN
23	isuzu	NaN
47	porsche	NaN

61 rows × 2 columns

In [ ]:

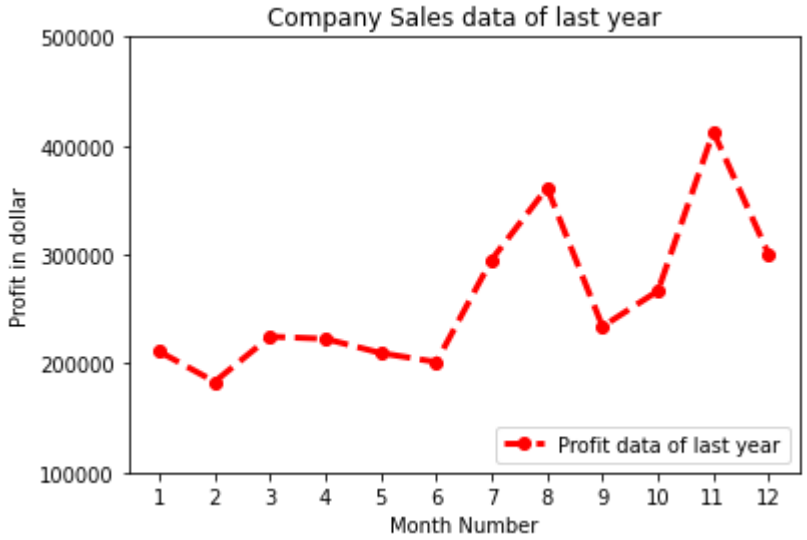
```
Read Total profit of all months and show it using a line plot
```

Generated line plot must include following Style properties: –

Line Style dotted and Line-color should be red Show legend at the lower right location. X label name = Month Number Y label name = Sold units number Add a circle marker. Line marker color as read Line width should be 3

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("/content/company_sales_data.csv")
profitList = df ['total_profit'].tolist()
monthList  = df ['month_number'].tolist()
plt.plot(monthList, profitList,
         label = 'Profit data of last year',
      color='r', marker='o', markerfacecolor='r',
      linestyle='--', linewidth=3)

plt.xlabel('Month Number')
plt.ylabel('Profit in dollar')
plt.legend(loc='lower right')
plt.title('Company Sales data of last year')
plt.xticks(monthList)
plt.yticks([100000, 200000, 300000, 400000, 500000])
plt.show()
```



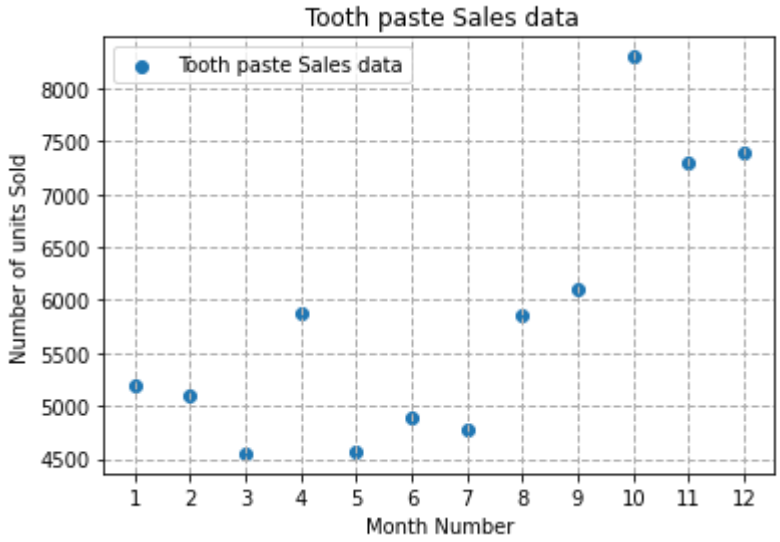
Read all product sales data and show it using a multiline plot Display the number of units sold per month for each product using multiline plots. (i.e., Separate Plotline for each product).

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("/content/company_sales_data.csv")
monthList  = df ['month_number'].tolist()
faceCremSalesData   = df ['facecream'].tolist()
faceWashSalesData   = df ['facewash'].tolist()
toothPasteSalesData = df ['toothpaste'].tolist()
bathingsoapSalesData   = df ['bathingsoap'].tolist()
shampooSalesData   = df ['shampoo'].tolist()
moisturizerSalesData = df ['moisturizer'].tolist()

plt.plot(monthList, faceCremSalesData,   label = 'Face cream Sales Data', marker='o', l
inewidth=3)
plt.plot(monthList, faceWashSalesData,   label = 'Face Wash Sales Data',  marker='o', l
inewidth=3)
plt.plot(monthList, toothPasteSalesData, label = 'ToothPaste Sales Data', marker='o', l
inewidth=3)
plt.plot(monthList, bathingsoapSalesData, label = 'bathingsoapSalesData', marker='o', l
inewidth=3)
plt.plot(monthList, shampooSalesData, label = 'shampooSalesData', marker='o', linewidth
=3)
plt.plot(monthList, moisturizerSalesData, label = 'moisturizerSalesData', marker='o', l
inewidth=3)

plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.legend(loc='upper left')
plt.xticks(monthList)
plt.yticks([1000, 2000, 4000, 6000, 8000, 10000, 12000, 15000, 18000])
plt.title('Sales data')
plt.show()
```

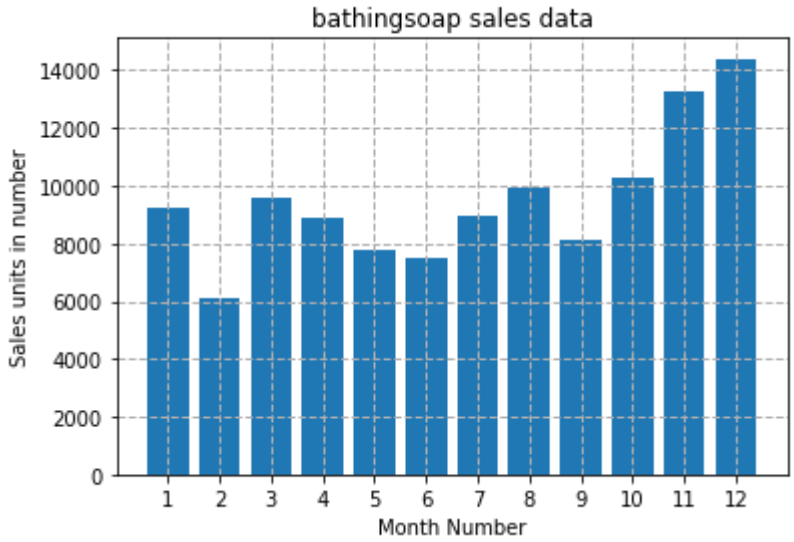
In [ ]:

```
#Read toothpaste sales data of each month and show it
 using a scatter plot
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("/content/company_sales_data.csv")
monthList  = df ['month_number'].tolist()
toothPasteSalesData = df ['toothpaste'].tolist()
plt.scatter(monthList, toothPasteSalesData, label
            = 'Tooth paste Sales data')
plt.xlabel('Month Number')
plt.ylabel('Number of units Sold')
plt.legend(loc='upper left')
plt.title(' Tooth paste Sales data')
plt.xticks(monthList)
plt.grid(True, linewidth= 1, linestyle="--")
plt.show()
```



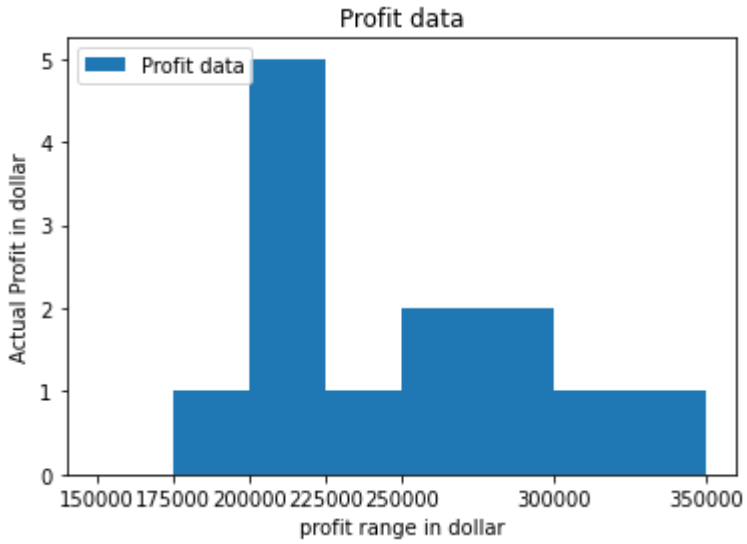
In [ ]:

```
# Read sales data of bathing soap of
#all months and show it using a bar chart.
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("/content/company_sales_data.csv")
monthList  = df ['month_number'].tolist()
bathingsoapSalesData   = df ['bathingsoap'].tolist()
plt.bar(monthList, bathingsoapSalesData)
plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.title(' Sales data')
plt.xticks(monthList)
plt.grid(True, linewidth= 1, linestyle="--")
plt.title('bathingsoap sales data')
plt.show()
```



In [ ]:

```
#Read the total profit of each month and show it using the histogram
 #to see the most common profit ranges
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("/content/company_sales_data.csv")
profitList = df ['total_profit'].tolist()
labels = ['low', 'average', 'Good', 'Best']
profit_range = [150000, 175000, 200000, 225000, 250000, 300000, 350000]
plt.hist(profitList, profit_range, label = 'Profit data')
plt.xlabel('profit range in dollar')
plt.ylabel('Actual Profit in dollar')
plt.legend(loc='upper left')
plt.xticks(profit_range)
plt.title('Profit data')
plt.show()
```



In [ ]:

```
#Calculate total sale data for last year for each product
 #and#show it using a Pie chart
 import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("/content/company_sales_data.csv")
monthList  = df ['month_number'].tolist()

labels = ['FaceCream', 'FaseWash', 'ToothPaste', 'Bathing soap', 'Shampoo', 'Moisturize
r']
salesData   = [df ['facecream'].sum(), df ['facewash'].sum(), df ['toothpaste'].sum(),
         df ['bathingsoap'].sum(), df ['shampoo'].sum(), df ['moisturizer'].sum()]
plt.axis("equal")
plt.pie(salesData, labels=labels,autopct='%1.1f%%')
plt.legend(loc='lower right')
plt.title('Sales data')
plt.show()
```