

Unit-5



Syllabus

- Module -5 (Data Processing) (9 hours)
- The os and sys modules, NumPy - Basics, Creating arrays, Arithmetic, Slicing, Matrix
- Operations, Random numbers. Plotting and visualization. Matplotlib - Basic plot, Ticks, Labels,
- and Legends. Working with CSV files. – Pandas - Reading, Manipulating, and Processing Data.
- Introduction to Micro services using Flask.

- The `os` module (and `sys`, and `path`)
- The `os` and `sys` modules provide numerous tools to deal with filenames, paths, directories. The `os` module contains two sub-modules `os.sys` (same as `sys`) and `os.path` that are dedicated to the system and directories; respectively.
- Whenever possible, you should use the functions provided by these modules for file, directory, and path manipulations. These modules are wrappers for platform-specific modules, so functions like `os.path.split` work on UNIX, Windows, Mac OS, and any other platform supported by Python.

- https://www.geeksforgeeks.org/os-module-python-examples/#:~:text=The%20OS%20module%20in%20Python,%20*os*%20and%20*os.

- Handling the Current Working Directory
- Consider Current Working Directory(CWD) as a folder, where the Python is operating. Whenever the files are called only by their name, Python assumes that it starts in the CWD which means that name-only reference will be successful only if the file is in the Python's CWD.
- Note: The folder where the Python script is running is known as the Current Directory. This is not the path where the Python script is located.
- Getting the Current working directory
- To get the location of the current working directory `os.getcwd()` is used.

To get the location of the current working directory `os.getcwd()` is used.

- `# Python program to explain os.getcwd() method`
-
- `# importing os module`
- `import os`
-
- `# Get the current working`
- `# directory (CWD)`
- `cwd = os.getcwd()`
-
- `# Print the current working`
- `# directory (CWD)`
- `print("Current working directory:", cwd)`

- you can build multi-platform path using the proper separator symbol:
- `>>> import os`
- `>>> import os.path`
- `>>> os.path.join(os.sep, 'home', 'user', 'work')`
- `'/home/user/work'`

- `>>> os.path.split('/usr/bin/python')`
- `('/usr/bin', 'python')`

- Functions
- The `os` module has lots of functions. We will not cover all of them thoroughly but this could be a good start to use the module.
- 7.2.1. Manipulating Directories
- The `getcwd()` function returns the current directory (in unicode format with `getcwdu()`).
- The current directory can be changed using `chdir()`:
 - `os.chdir(path)`
- The `listdir()` function returns the content of a directory. Note,

The sys module

- The `sys` module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter. Let's consider the below example.

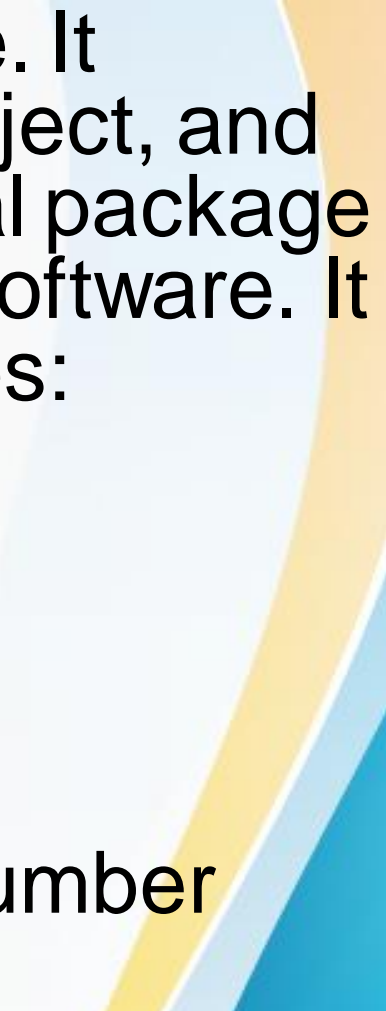
- `import sys`
`print(sys.version)`




- <https://www.geeksforgeeks.org/python-sys-module/#:~:text=The%20sys%20module%20in%20Python,interact%20strongly%20with%20the%20interpreter.>

NumPy in Python



- NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including these important ones:
 - A powerful N-dimensional array object
 - Sophisticated (broadcasting) functions
 - Tools for integrating C/C++ and Fortran code
 - Useful linear algebra, Fourier transform, and random number capabilities
- 

- Arrays in NumPy: NumPy's main object is the homogeneous multidimensional array.
 - It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.
 - In NumPy dimensions are called axes. The number of axes is rank.
 - NumPy's array class is called ndarray. It is also known by the alias array.
- 

- Example :
- $\begin{bmatrix} 1, & 2, & 3 \\ 4, & 2, & 5 \end{bmatrix}$
- Here,
- rank = 2 (as it is 2-dimensional or it has 2 axes)
- first dimension(axis) length = 2, second dimension has length = 3
- overall shape can be expressed as: (2, 3)

- # Python program to demonstrate
- # basic array characteristics
- import numpy as np
- # Creating array object
- arr = np.array([[1, 2, 3],
- [4, 2, 5]])
- # Printing type of arr object
- print("Array is of type: ", type(arr))
- # Printing array dimensions (axes)
- print("No. of dimensions: ", arr.ndim)
- # Printing shape of array
- print("Shape of array: ", arr.shape)
- # Printing size (total number of elements) of array
- print("Size of array: ", arr.size)
-
- # Printing type of elements in array
- print("Array stores elements of type: ", arr.dtype)

- Output :
- Array is of type:
- No. of dimensions: 2
- Shape of array: (2, 3)
- Size of array: 6
- Array stores elements of type: int64



- Array creation: There are various ways to create arrays in NumPy.
- For example, you can create an array from a regular Python list or tuple using the `array` function. The type of the resulting array is deduced from the type of the elements in the sequences.
- Often, the elements of an array are originally unknown, but its size is known. Hence, NumPy offers several functions to create arrays with initial placeholder content. These minimize the necessity of growing arrays, an expensive operation. For example: `np.zeros`, `np.ones`, `np.full`, `np.empty`, etc.
- To create sequences of numbers, NumPy provides a function analogous to `range` that returns arrays instead of lists.

- `arange`: returns evenly spaced values within a given interval. step size is specified.
- `linspace`: returns evenly spaced values within a given interval. num no. of elements are returned.
- Reshaping array: We can use `reshape` method to reshape an array. Consider an array with shape $(a_1, a_2, a_3, \dots, a_N)$. We can reshape and convert it into another array with shape $(b_1, b_2, b_3, \dots, b_M)$. The only required condition is: $a_1 \times a_2 \times a_3 \dots \times a_N = b_1 \times b_2 \times b_3 \dots \times b_M$. (i.e original size of array remains unchanged.)
- Flatten array: We can use `flatten` method to get a copy of array collapsed into one dimension. It accepts `order` argument. Default value is 'C' (for row-major order). Use 'F' for column major order.


- **Array Indexing:** Knowing the basics of array indexing is important for analysing and manipulating the array object. NumPy offers many ways to do array indexing.
- **Slicing:** Just like lists in python, NumPy arrays can be sliced. As arrays can be multidimensional, you need to specify a slice for each dimension of the array.
- **Integer array indexing:** In this method, lists are passed for indexing for each dimension. One to one mapping of corresponding elements is done to construct a new arbitrary array.
- **Boolean array indexing:** This method is used when we want to pick elements from array which satisfy some condition.

Introduction to Matplotlib

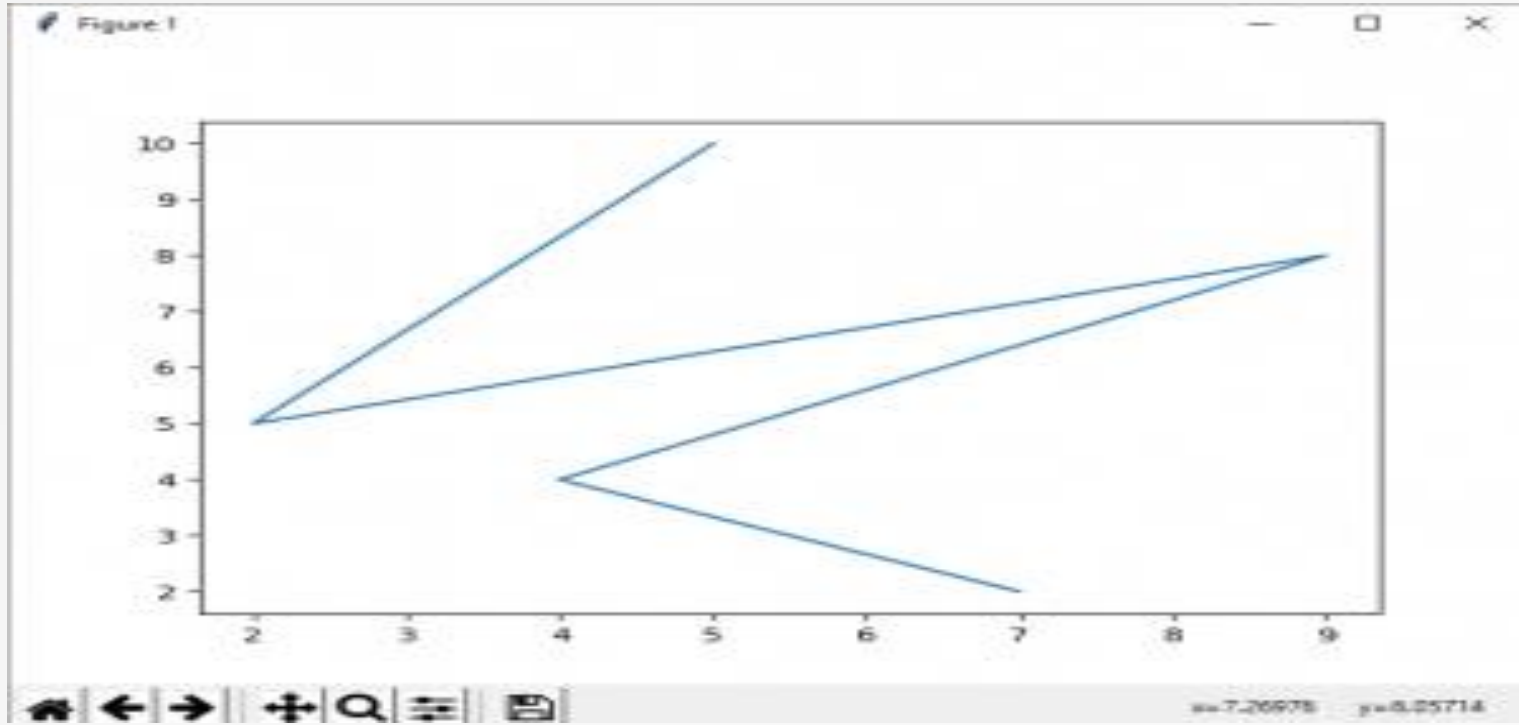
- Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.
- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

- Importing matplotlib :
- `from matplotlib import pyplot as plt`
- or
- `import matplotlib.pyplot as plt`



- Basic plots in Matplotlib :
 - Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information. Some of the sample plots are covered here.
- 

- Line plot :
- # importing matplotlib module
- from matplotlib import pyplot as plt
- # x-axis values
- x = [5, 2, 9, 4, 7]
- # Y-axis values
- y = [10, 5, 8, 4, 2]
- # Function to plot
- plt.plot(x,y)
-
- # function to show the plot



Working with csv files in Python

- What is a CSV ?
- CSV (Comma Separated Values) is a simple file format used to store tabular data, such as a spreadsheet or database. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.
- For working CSV files in python, there is an inbuilt module called csv.

- # importing csv module
- import csv
- # csv file name
- filename = "aapl.csv"

- # initializing the titles and rows list
- fields = []
- rows = []

- # reading csv file
- with open(filename, 'r') as csvfile:
 - # creating a csv reader object
 - csvreader = csv.reader(csvfile)
 - # extracting field names through first row
 - fields = next(csvreader)
 -
 -

- # extracting each data row one by one
- for row in csvreader:
- rows.append(row)
-
- # get total number of rows
- print("Total no. of rows: %d"%(csvreader.line_num))
-
- # printing the field names
- print('Field names are: ' + ', '.join(field for field in fields))
-
- # printing first 5 rows
- print('\nFirst 5 rows are:\n')
- for row in rows[:5]:
- # parsing each column of a row
- for col in row:
- print("%10s"%col,end=" "),
- print('\n')

Total no. of rows: 252

Field names are:Date, Open, High, Low, Close, Volume

First 5 rows are:

7-Dec-16	109.26	111.19	109.16	111.03	29998719
6-Dec-16	109.50	110.36	109.19	109.95	26195462
5-Dec-16	110.00	110.03	108.25	109.11	34324540
2-Dec-16	109.17	110.09	108.85	109.90	26527997
1-Dec-16	110.36	110.94	109.03	109.49	37086862

Reading and Writing CSV Files in Python

- Reading from CSV file
- Python contains a module called `csv` for the handling of CSV files. The `reader` class from the module is used for reading data from a CSV file. At first, the CSV file is opened using the `open()` method in 'r' mode (specifies read mode while opening a file) which returns the file object then it is read by using the `reader()` method of CSV module that returns the reader object that iterates throughout the lines in the specified CSV document.
- Syntax:
 - `csv.reader(csvfile, dialect='excel', **fmtparams`
 - Note: The 'with' keyword is used along with the `open()` method as it simplifies exception handling and automatically closes the CSV file.

Consider the below CSV file –

	A	B	C
1	name	age	grade
2	Steve	13	A
3	John	14	F
4	Nancy	14	C
5	Ravi	13	B
6			

- `import csv`
-
- `# opening the CSV file`
- `with open('Giants.csv', mode ='r')as file:`
-
- `# reading the CSV file`
- `csvFile = csv.reader(file)`
-
- `# displaying the contents of the CSV file`
- `for lines in csvFile:`
- `print(lines)`

- Output:
- [['Steve', 13, 'A'],
- ['John', 14, 'F'],
- ['Nancy', 14, 'C'],
- ['Ravi', 13, 'B']]



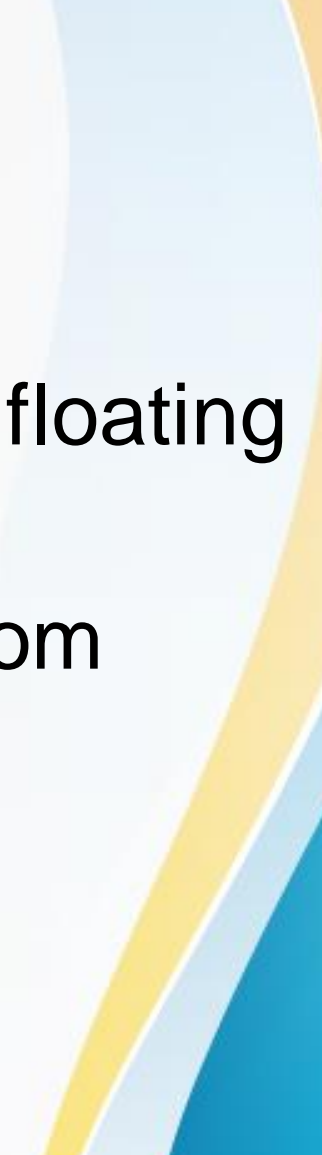
<https://www.geeksforgeeks.org/reading-and-writing-csv-files-in-python/>



Introduction to Pandas in Python

- <https://www.geeksforgeeks.org/introduction-to-pandas-in-python/>

- Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

- Advantages
 - Fast and efficient for manipulating and analyzing data.
 - Data from different file objects can be loaded.
 - Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
 - Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
 - Data set merging and joining.
 - Flexible reshaping and pivoting of data sets
 - Provides time-series functionality.
- 

- Getting Started
- The first step of working in pandas is to ensure whether it is installed in the Python folder or not. If not then we need to install it in our system using pip command. Type cmd command in the search box and locate the folder using cd command where python-pip file has been installed. After locating it, type the command:
 - `pip install pandas`
 - After the pandas have been installed into the system, you need to import the library. This module is generally imported as:
 - `import pandas as pd`

- Pandas generally provide two data structures for manipulating data, They are:
- Series
- DataFrame
- Series: Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called indexes. Pandas Series is nothing but a column in an excel sheet. Labels need not be unique but must be a hashable type. The object supports both integer and label-based indexing and provides a host of methods for performing operations involving the index.

- Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index. Pandas Series is nothing but a column in an excel sheet.
- Labels need not be unique but must be a hashable type. The object supports both integer and label-based indexing and provides a host of methods for performing operations involving the index.

	<i>Name</i>	<i>Team</i>	<i>Number</i>
0	Avery Bradley	Boston Celtics	0.0
1	John Holland	Boston Celtics	30.0
2	Jonas Jerebko	Boston Celtics	8.0
3	Jordan Mickey	Boston Celtics	NaN
4	Terry Rozier	Boston Celtics	12.0
5	Jared Sullinger	Boston Celtics	7.0
6	Evan Turner	Boston Celtics	11.0

`ser = pd.Series (df ['Name'])`

`ser = pd.Series (df ['Team'])`

`ser = pd.Series (df ['Number'])`



- Creating a Pandas Series
- In the real world, a Pandas Series will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, and Excel file. Pandas Series can be created from the lists, dictionary, and from a scalar value etc. Series can be created in different ways, here are some ways by which we create a series:
- Creating a series from array: In order to create a series from array, we have to import a numpy module and have to use `array()` function.

- `# import pandas as pd`
- `import pandas as pd`
-
- `# import numpy as np`
- `import numpy as np`
-
- `# simple array`
- `data = np.array(['g','e','e','k','s'])`
-
- `ser = pd.Series(data)`



```
0  
1  
2  
3  
4
```

0	00
1	00
2	00
3	k
4	s

```
dtype: object
```

- DataFrame
- Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

Columns

Name *Team* *Number* *Position* *Age*

Rows

0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Data



- Creating a DataFrame:
- In the real world, a Pandas DataFrame will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, an Excel file. Pandas DataFrame can be created from the lists, dictionary, and from a list of dictionaries, etc.

- `import pandas as pd`
-
- `# Calling DataFrame constructor`
- `df = pd.DataFrame()`
- `print(df)`
-
- `# list of strings`
- `lst = ['Geeks', 'For', 'Geeks', 'is',`
- `'portal', 'for', 'Geeks']`
-

output

- Empty DataFrame
- Columns: []
- Index: []
- 0
- 0 Geeks
- 1 For
- 2 Geeks
- 3 is
- 4 portal
- 5 for

- Why Pandas is used for Data Science
- Pandas are generally used for data science but have you wondered why? This is because pandas are used in conjunction with other libraries that are used for data science. It is built on the top of the NumPy library which means that a lot of structures of NumPy are used or replicated in Pandas. The data produced by Pandas are often used as input for plotting functions of Matplotlib, statistical analysis in SciPy, and machine learning algorithms in Scikit-learn.

- Pandas program can be run from any text editor but it is recommended to use Jupyter Notebook for this as Jupyter given the ability to execute code in a particular cell rather than executing the entire file. Jupyter also provides an easy way to visualize pandas data frames and plots.

- How do you add two matrices in Python using Numpy?

```
import numpy as np

# The first matrix will be

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

#The second matrix will be

Q = np.array([[4, 5], [6, 7]])

print("Elements of the first matrix")

print(P)

print("Elements of the second matrix")

print(Q)

# adding two matrix

print("The sum of the two matrices is")

print(np.add(P, Q))
```

- Output:
- Elements of the first matrix
- $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
- Elements of the second matrix
- $\begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix}$
- The sum of the two matrices is
- $\begin{bmatrix} 5 & 7 \\ 9 & 11 \end{bmatrix}$



- Introducing Flask
- Flask is a micro framework for Python web development. A framework, in the
- simplest terms, is a library or collection of libraries that aims to solve a part of a generic
- problem instead of a complete specific one. When building web applications, there
- are some problems that will always need to be solved, such as routing from URLs to
- resources, inserting dynamic data into HTML, and interacting with an end user

- Flask is a micro framework because it implements only core functionality (including
- routing) but leaves more advanced functionality (including authentication and
- database ORMs) to extensions. The result of this is less initial setup for the first-time
- user and more choice and flexibility for the experienced user. This is in contrast
- with "fuller" frameworks, such as Django, which dictate their own ORM and
- authentication technologies.

- Writing "Hello, World!"
- Now, we'll create a basic web page and serve it using Flask's built-in server to
- localhost. This means that we'll run a web server on our local machine that we can
- easily make requests to from our local machine. This is very useful for development
- but not suited for production applications. Later on, we'll take a look at how to serve
- Flask web applications using the popular Apache web server.

- Writing the code
- Our application will be a single Python file. Create a directory in your home directory
- called firstapp and a file inside this called hello.py. In the hello.py file, we'll write
- code to serve a web page comprising the static string "Hello, World!". The code looks
- as follows:
- `from flask import Flask`
- `app = Flask(__name__)`


- `@app.route("/")`
- `def index():`
- `return "Hello, World!"`
- `if __name__ == '__main__':`
- `app.run(port=5000, debug=True)`

- Let's break down what this does. The first line should be familiar; it simply imports
- Flask from the package flask. The second line creates an instance of the Flask object
- using our module's name as a parameter. Flask uses this to resolve resources, and in
- complex cases, one can use something other than `__name__` here. For our purposes,
- we'll always use `__name__`, which links our module to the Flask object.

- Line 3 is a Python decorator. Flask uses decorators for URL routing, so this line of
- code means that the function directly below it should be called whenever a user
- visits the main root page of our web application (which is defined by the single
- forward slash). If you are not familiar with decorators, these are beautiful Python
- shortcuts that seem a bit like black magic at first. In essence, they call a function that
- takes the function defined under the decorator (in our case,

- The next two lines should also seem familiar. They define a very simple function
- that returns our message. As this function is called by Flask when a user visits our
- application, the return value of this will be what is sent in response to a user who
- requests our landing page.

- Line 6 is a Python idiom with which you are probably familiar. This is a simple
- conditional statement that evaluates to True if our application is run directly.
- It is used to prevent Python scripts from being unintentionally run when they
- are imported into other Python files.
- The final line kicks off Flask's development server on our local machine. We set it to
- run on port 5000 (we'll use port 80 for production) and set debug to True, which

- Running the code
 - To run our development web server, simply fire up a terminal and run the `hello.py`
 - file. If you used the same structure outlined in the previous section, the commands
 - will be as follows:
 - `cd firstapp/hello`
 - `python hello.py`
- 

output

The screenshot shows a PDF viewer window with two tabs: 'syllabus.pdf' and 'Flask_Building Python Web Serv'. The active document is 'C:/Users/Sujith/Desktop/python-s6/text%20books/Flask_%20Building%20Python%20Web%20Services%201st%20Edition.Pdf'. The viewer interface includes a search bar (27 of 770), navigation icons, and a toolbar with options like Page view, Read aloud, Add text, Draw, Highlight, and Erase.

The PDF content features a horizontal line with '[7]' centered below it. Below this is a paragraph:

Hello, World!

You should see the "Hello, World!" string displayed in your browser as in the following screenshot. Congratulations, you've built your first web application using Flask!

The browser screenshot shows the address bar with 'http://localhost:5000/' and the page content 'Hello, World!'.

Deploying our application to production

It's great to have an application that runs, but inherent to the idea of a web application is the idea that we want others to be able to use it. As our application is Python-based, we are a bit limited in how we can run our application on a web server (many traditional web hosts are only configured to run PHP and/or .NET applications). Let's consider how to serve Flask applications using a **Virtual Private Server (VPS)** running Ubuntu Server, Apache, and WSGI.

From this point on, we'll maintain *two* environments. The first is our development

The Windows taskbar at the bottom shows the search bar, taskbar icons for Edge, File Explorer, and other apps, and the system tray with the time 10:15 and date 27-07-2022.

Thank You

