PYTHON LIBRARIES

PYTHON LIBRARY

1. A Python library is a collection of pre-written code and functionalities that can be used to extend the capabilities of Python programming language. It consists of a set of modules or packages that provide specific functionality, making it easier for developers to implement complex tasks without having to write the code from scratch.
2. Python libraries are designed to be reusable, modular, and typically focused on a specific domain or purpose. They can range from general-purpose libraries that provide common functions and utilities, such as math or file manipulation, to specialized libraries tailored for specific tasks like data analysis, machine learning, web development, or scientific computing
3. Libraries in Python are typically distributed as packages, which contain modules and sub-modules that organize the code in a structured manner. These packages can be installed from external sources using package managers like pip, or they can be included within a project's source code.
4. To use a Python library, you need to import the desired modules or classes into your code and access their functions and methods. Libraries can significantly simplify the development process by providing ready-to-use solutions and reducing the amount of code you need to write, leading to increased productivity and faster development cycles.

A FEW EXAMPLES

pandas - A powerful library **for data manipulation and analysis. It provides data structures like DataFrames** for efficient handling of structured data, along with functions for filtering, aggregating, and transforming data.
Example: Analyzing and manipulating a dataset of sales records.

matplotlib - A plotting library **for creating visualizations.** It offers a wide range of chart types and customization options, making it ideal for creating graphs, histograms, scatter plots, and more.
Example: Plotting the trends in stock market prices over time.

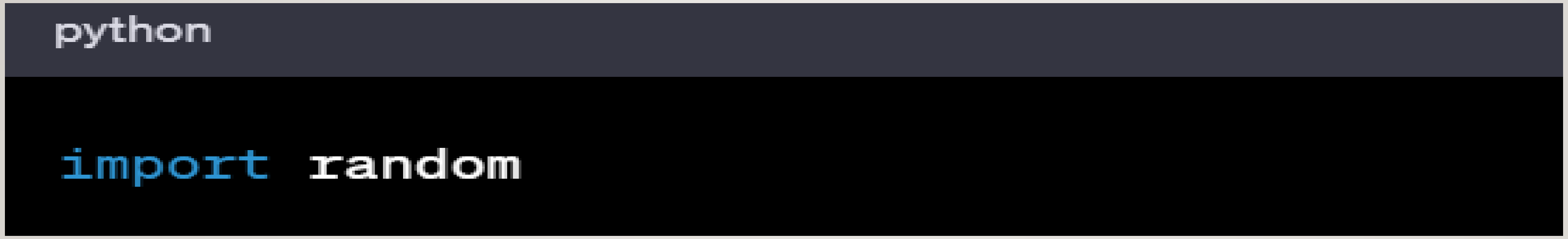
scikit-learn - A machine learning library that provides various algorithms for **classification, regression, clustering, and other tasks. It also includes utilities for data preprocessing,** model evaluation, and feature selection.
Example: Building a predictive model to classify emails as spam or not.

5. numpy – A fundamental library for scientific computing. It offers support for large, **multidimensional array** and a collection of mathematical functions for array operations.
Example: Performing matrix operations or calculating statistical measures on numerical data.

PYTHON PROGRAM

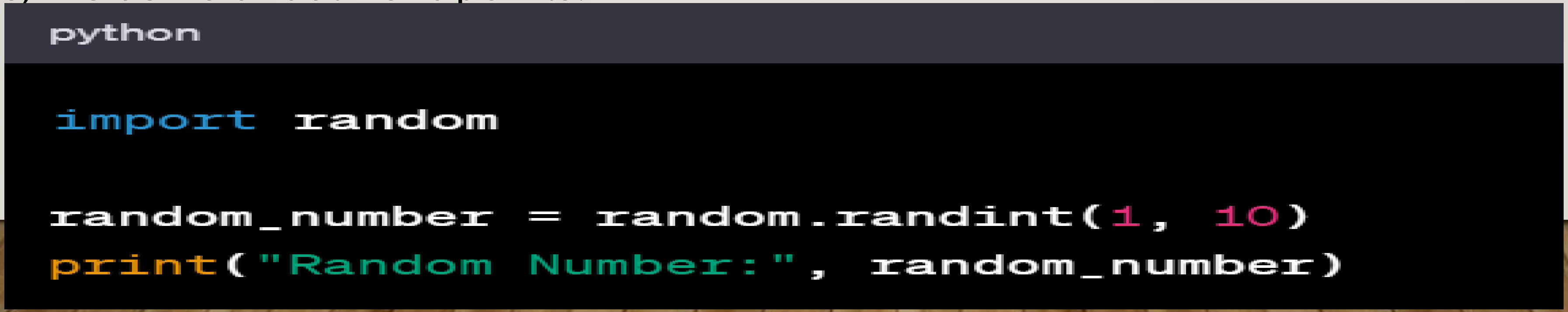
- **‘random’**

1. Importing the library: In python script, import **‘random’** module to access the random number generation function:



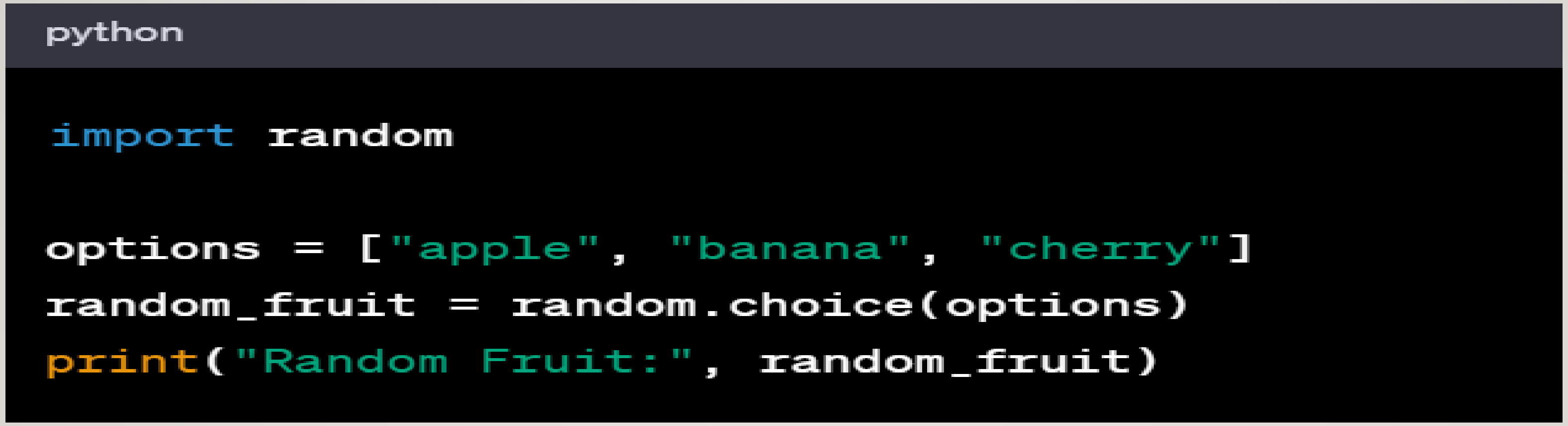
```python
import random
```

2. For Generating a Random Integer, **‘random.randint( )’** function is used within a specified range, inclusive of both endpoints:



```python
import random

random_number = random.randint(1, 10)
print("Random Number:", random_number)
```

3. Random Element Selection: The ‘random.choice( )’ function allows to select a random element from a sequence (e.g. list,tuple,string):



python

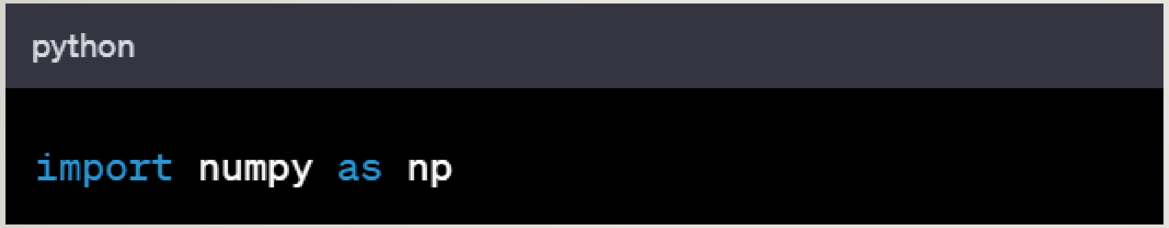
```
import random

options = ["apple", "banana", "cherry"]
random_fruit = random.choice(options)
print("Random Fruit:", random_fruit)
```

The **‘random’** library offers many more functions and capabilities for random number generation and selection. By importing and utilizing this library, we can add randomness and unpredictability to our data Set used in Python programs.

• ‘numpy’

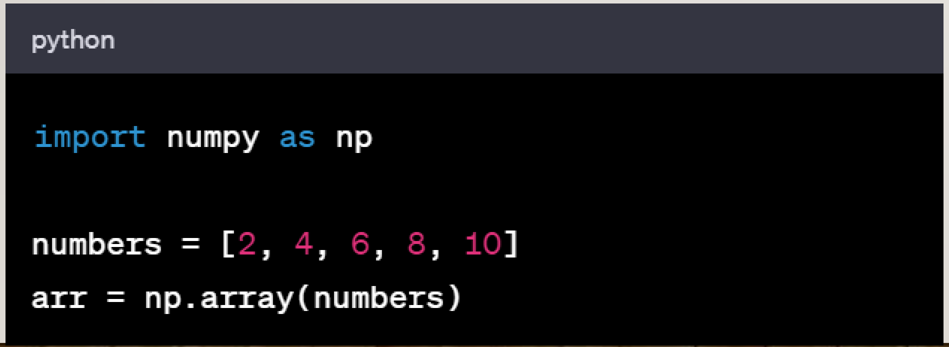
Computing the Mean of an Array :

1. Importing the ‘numpy’ Library in python script to access its functions and classes:



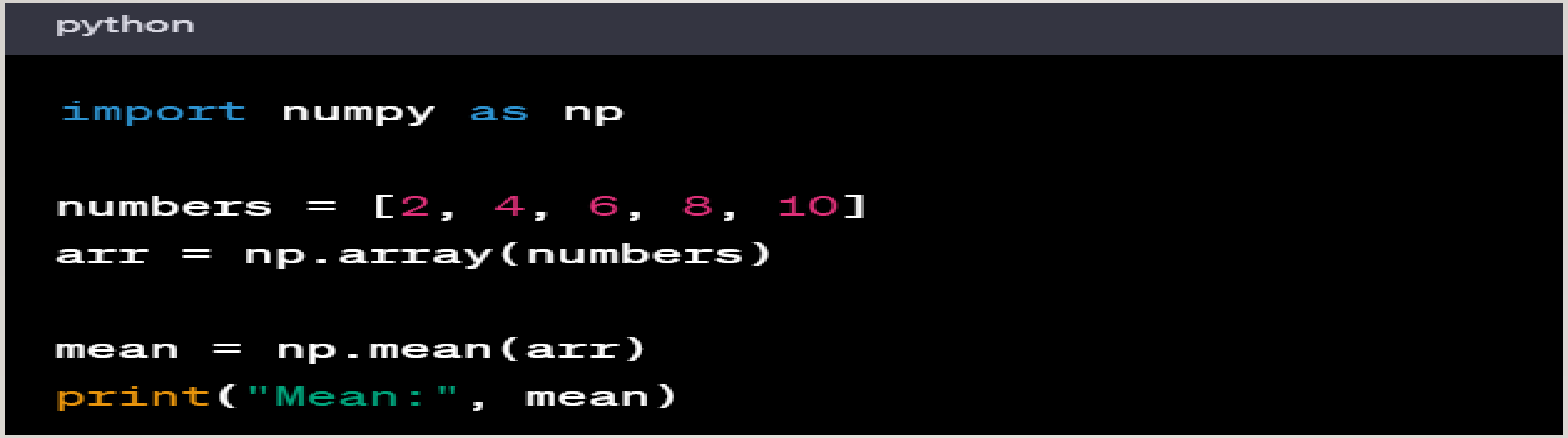
```python
import numpy as np
```

2. Creating an array:



```python
import numpy as np

numbers = [2, 4, 6, 8, 10]
arr = np.array(numbers)
```

3. Computing the Mean: numpy provides various mathematical functions that operate on arrays efficiently, using **‘np.mean( )’** function to calculate the mean of the array:

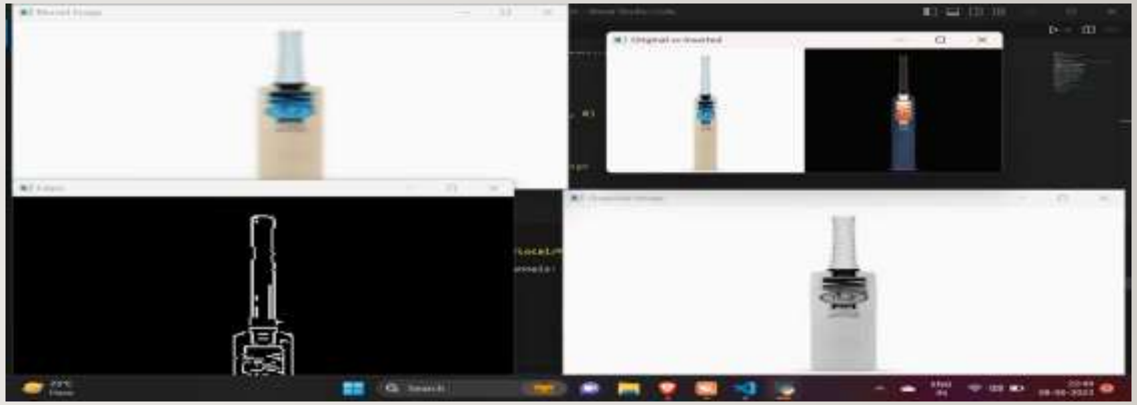


```python
import numpy as np

numbers = [2, 4, 6, 8, 10]
arr = np.array(numbers)

mean = np.mean(arr)
print("Mean:", mean)
```

The **‘numpy’** library offers so many functionalities, including array manipulation, element-wise operations, statistical functions, linear algebra operations, and much more, It enables efficient numerical computations and data processing, making it indispensable for tasks like scientific simulations, statistical analysis, machine learning, and signal processing.

OpenCV - Open Source Computer Vision Library

1. OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library.
2. OpenCV supports various programming languages, including C++, Python, Java, and MATLAB, making it accessible and widely used across different platforms and applications.
3. OpenCV has a large and active community, with extensive documentation, tutorials, and resources available online.
4. The library offers a wide range of functionalities, including image and video processing, feature detection and extraction, object detection and tracking, camera calibration, machine learning integration, and more.



1. Image Processing: OpenCV provides a rich set of functions for image processing, including operations like image resizing, cropping, rotation, filtering, and thresholding. It also supports advanced techniques such as edge detection, contour detection, and image morphological operations.

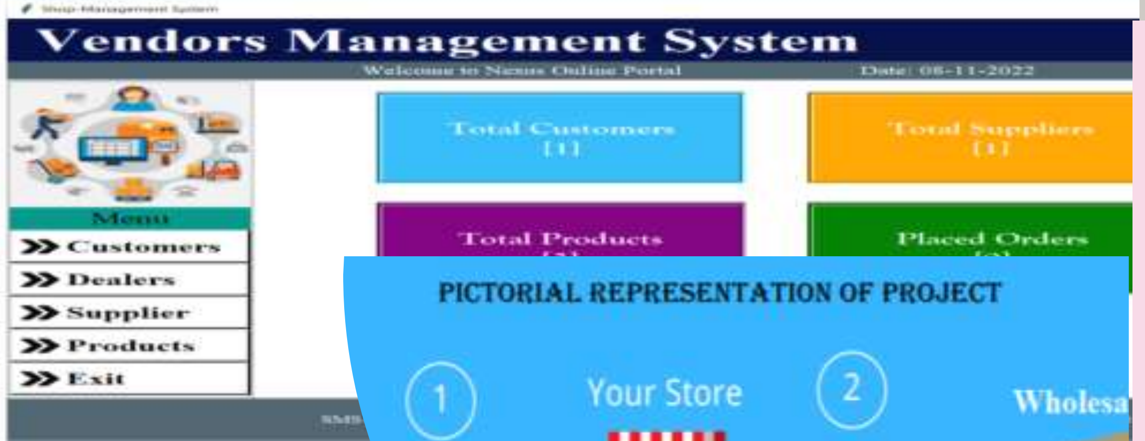
2. Video Processing: OpenCV enables you to read and write videos, capture frames from cameras, and perform video-related operations like frame extraction, video stabilization, and video object tracking.

3. Object Detection and Recognition: OpenCV provides pre-trained models and functions for object detection and recognition. You can use popular algorithms like HAR cascades and deep learning-based approaches (e.g., using frameworks like TensorFlow and PyTorch).

May all who enter as a guest,
leave as a friends !
We are most delightfully welcomes you to
Join Us Today !
Nexus Technologies pvt. Ltd.

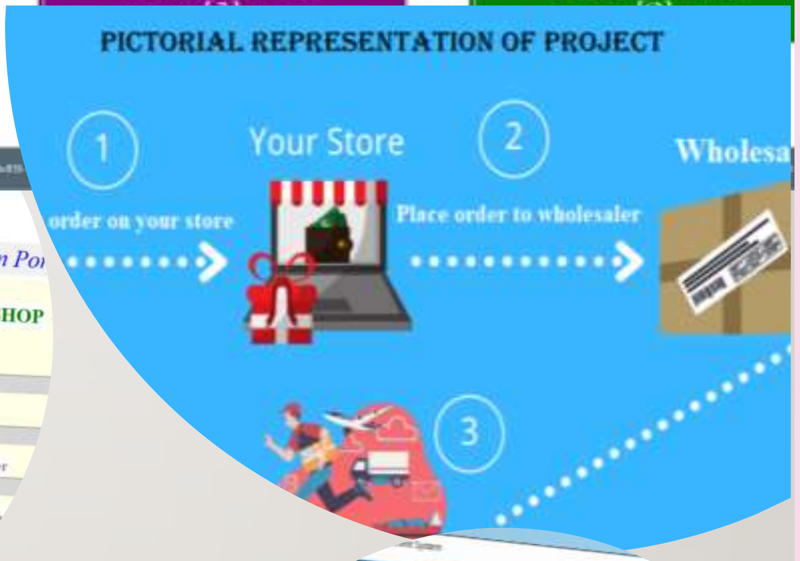
First page of our software!!

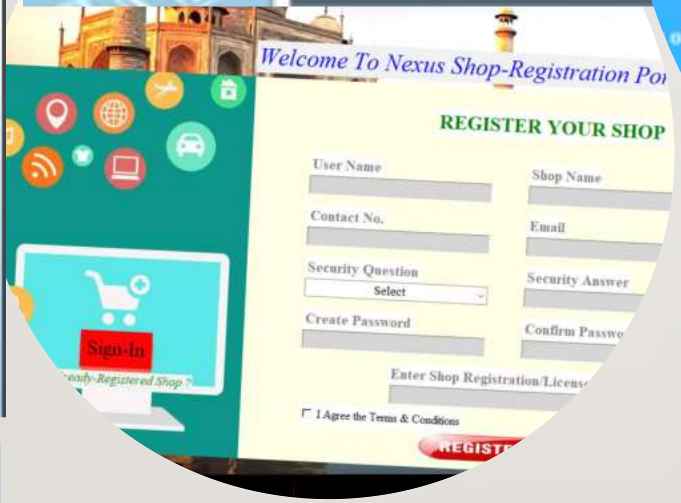
n portal
s able to
omers,
lers and
total
ite and
ers by
portal.”


Shop-Management System

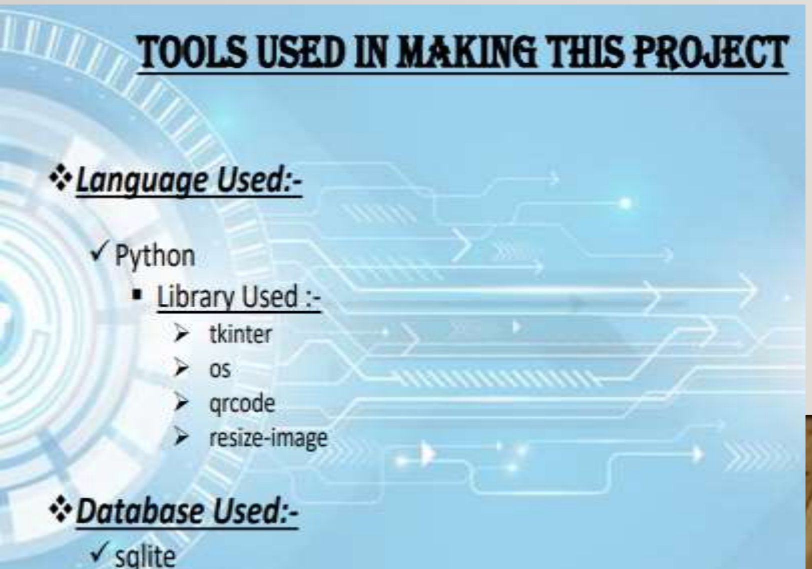
Vendors Management System

Welcome to Nexus Online Portal — Date: 08-11-2022

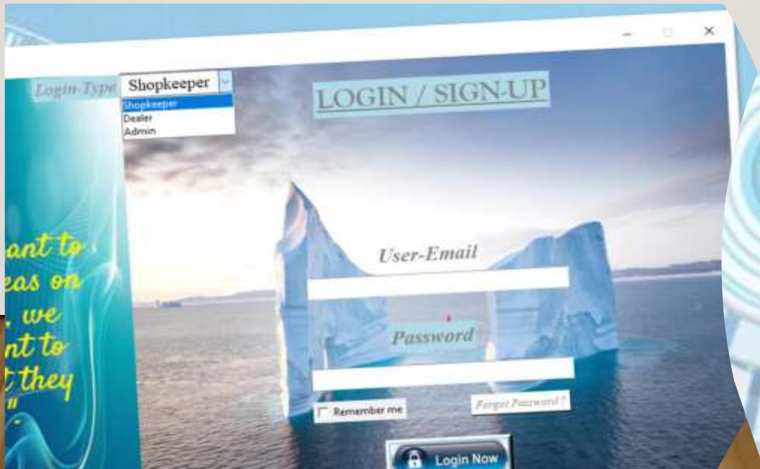

PICTORIAL REPRESENTATION OF PROJECT


Welcome To Nexus Shop-Registration Po

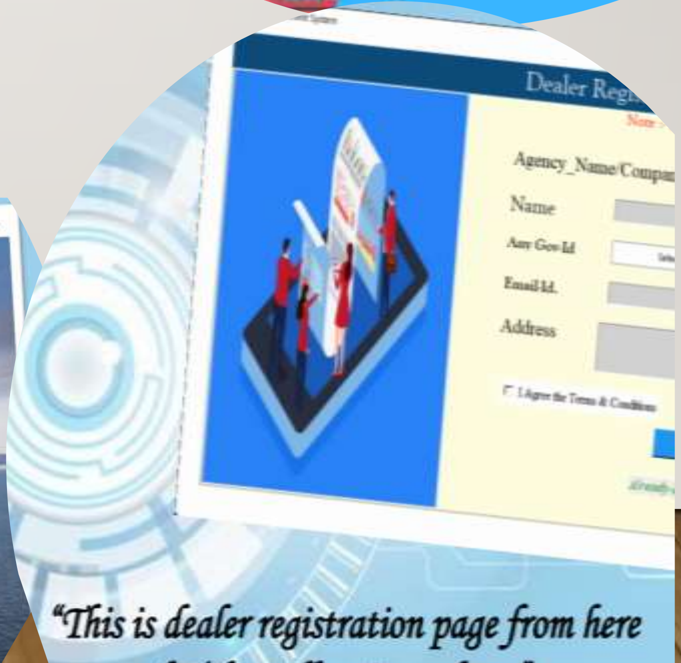
REGISTER YOUR SHOP


TOOLS USED IN MAKING THIS PROJECT

- ***Language Used:-***
  - ✓ Python
    - Library Used :-
      - tkinter
      - os
      - qrcode
      - resize-image
- ***Database Used:-***
  - ✓ sqlite



ant to
eas on
, we
nt to
t they
”



“This is dealer registration page from here

TKINTER FRAMEWORK

Python has a lot of GUI frameworks, Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux.

Introduction to Pandas

Pandas Data Structures

Data Cleaning with Pandas

Data Manipulation with Pandas

Data Analysis with Pandas

Conclusion

Introduction to Pandas

1. Provides easy-to-use data structures and data analysis tools for handling large datasets.
2. Is built on top of NumPy and provides two primary data structures: Series and DataFrames.

Pandas Data Structures

1. A **Series** is a one-dimensional array-like object that can hold any data type such as integers, strings, or even other Python objects.
2. A **DataFrame**, on the other hand, is a two-dimensional table-like structure that consists of rows and columns. It is similar to a spreadsheet or a SQL table.

Data Cleaning with Pandas

1. It involves identifying and handling missing values, duplicates, and other inconsistencies in the data.
2. provides powerful string manipulation functions for cleaning text data, regular expression functions for pattern matching, and datetime functions for handling date and time data.

Data Manipulation with Pandas

1. Provides several functions for data manipulation, such as sorting, filtering, and grouping data.
2. Provides pivot tables for summarizing data, merging and joining functions for combining multiple datasets, and time series functions for working with time-based data.

Data Analysis with Pandas

1. provides several functions for data analysis, such as descriptive statistics, correlation analysis, and regression analysis.
2. Provides visualization tools for creating charts and graphs to visualize your data.

Conclusion

In conclusion, Pandas is a powerful library for data manipulation and analysis in Python. It provides easy-to-use data structures and data analysis tools for handling large datasets. With Pandas, you can easily load, manipulate, and analyze data from various sources such as CSV files, Excel spreadsheets, SQL databases, and more.

USE OF NUMPY MATPLOTLIB IN PYTHON

NUMPY

A library for numerical computing

Efficient handling of large arrays and matrices

Mathematical functions for array operations

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)
```

PANDAS

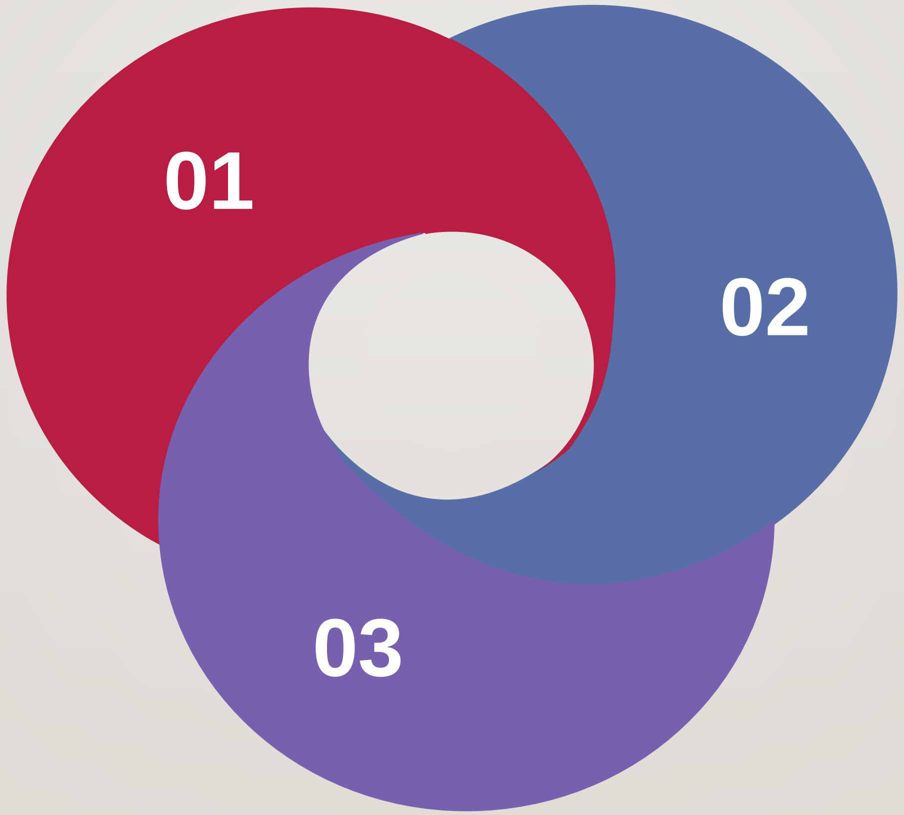
- A library for data manipulation and analysis
- Data structures for easy handling of structured data
- Efficient querying and filtering of data

```python
import pandas as pd

df = pd.read_csv('data.csv')

print(df.to_string())
```

```python
import pandas

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

myvar = pandas.DataFrame(mydataset)

print(myvar)
```

MATPLOTLIB

A library for data visualization



A wide range of plot types and customizations

Integration with NumPy and Pandas

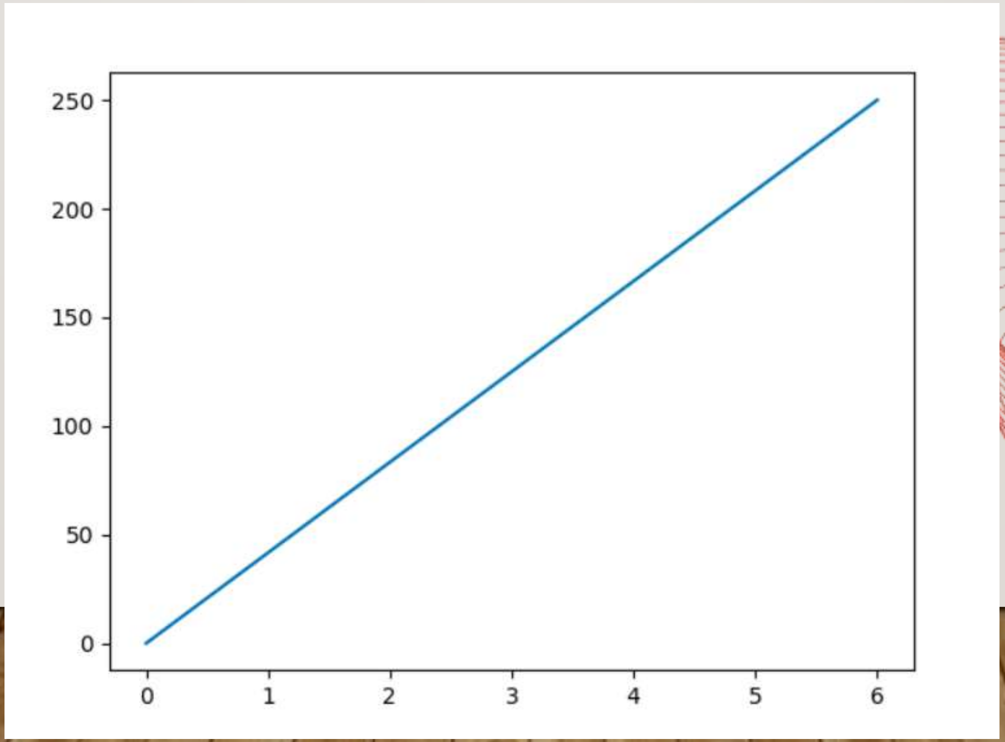
What is Matplotlib?

-
- Matplotlib is a low level graph plotting library in python that serves as a visualization utility.
- Matplotlib was created by John D. Hunter.
- Matplotlib is open source and we can use it freely.
- Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Install it using this command:

pip install matplotlib

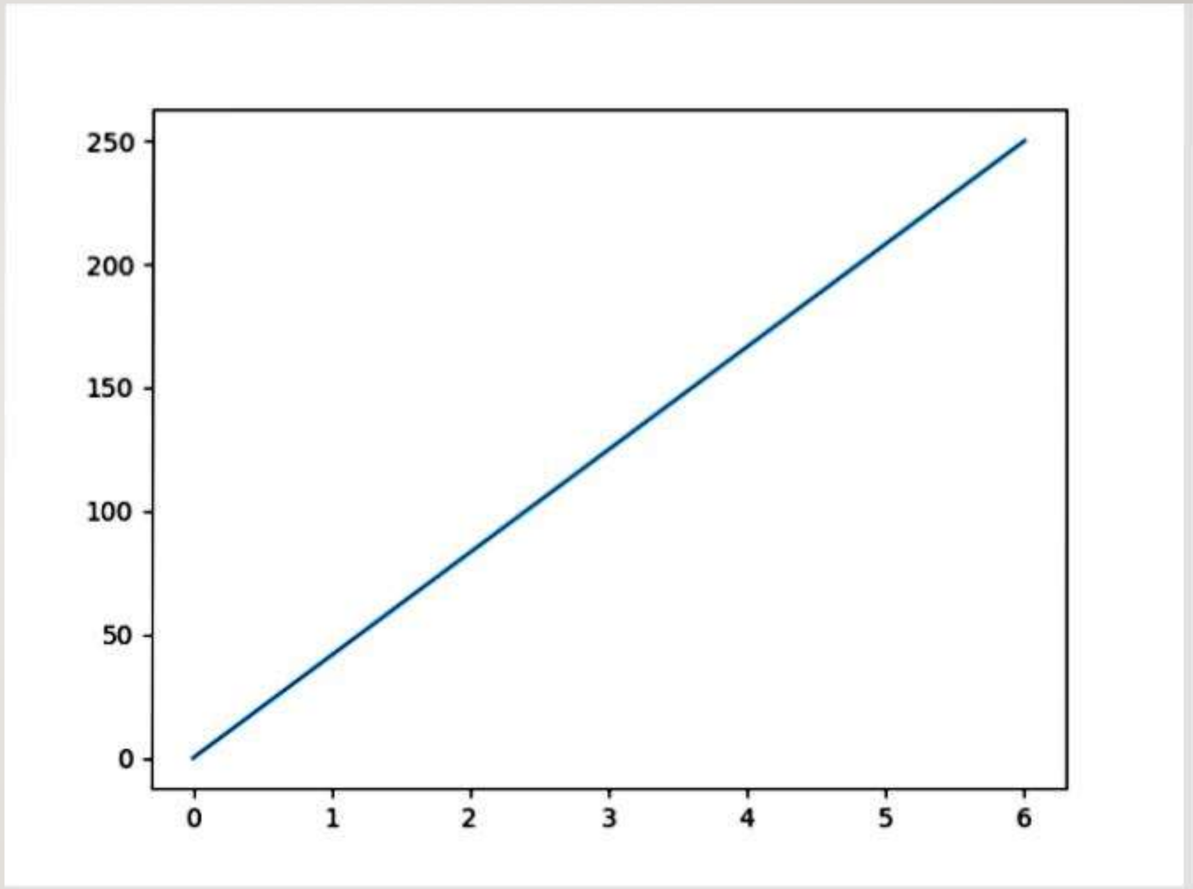
```python
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```



EXAMPLE

Draw a line in a diagram from position (0,0) to position (6,250):

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

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

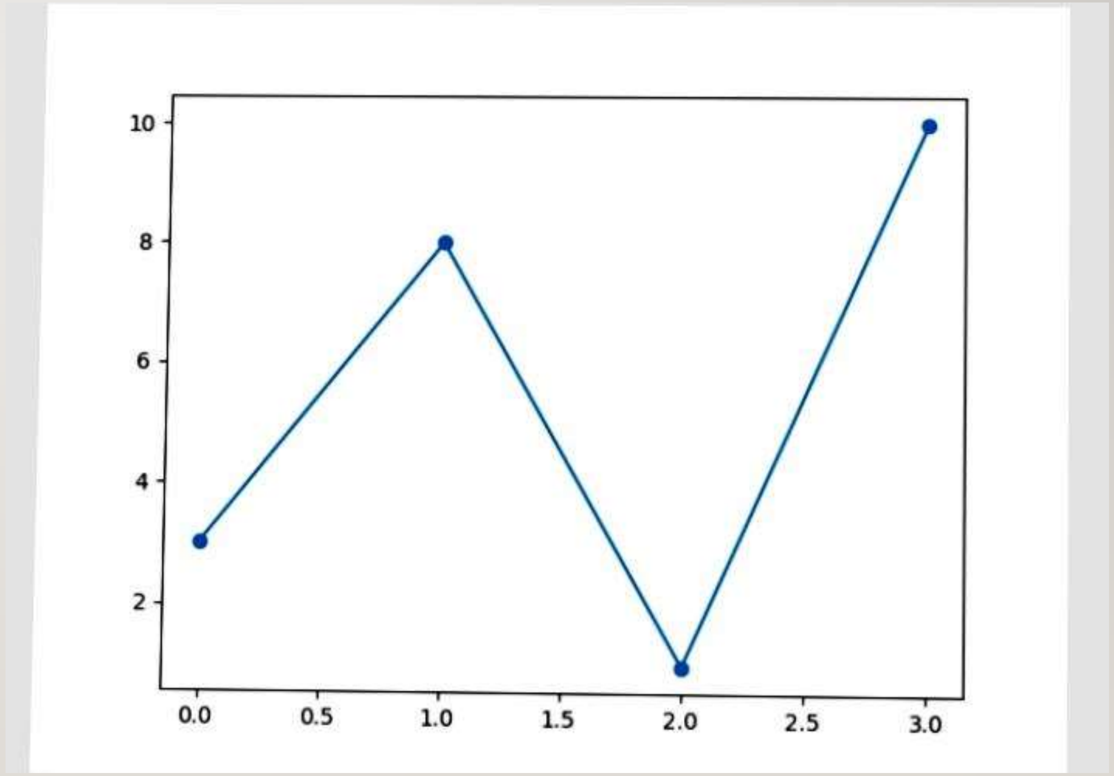


Markers

You can use the keyword argument marker to emphasize each point with a specified marker:

Mark each point with circle here:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = ‘o’)
plt.show()
```

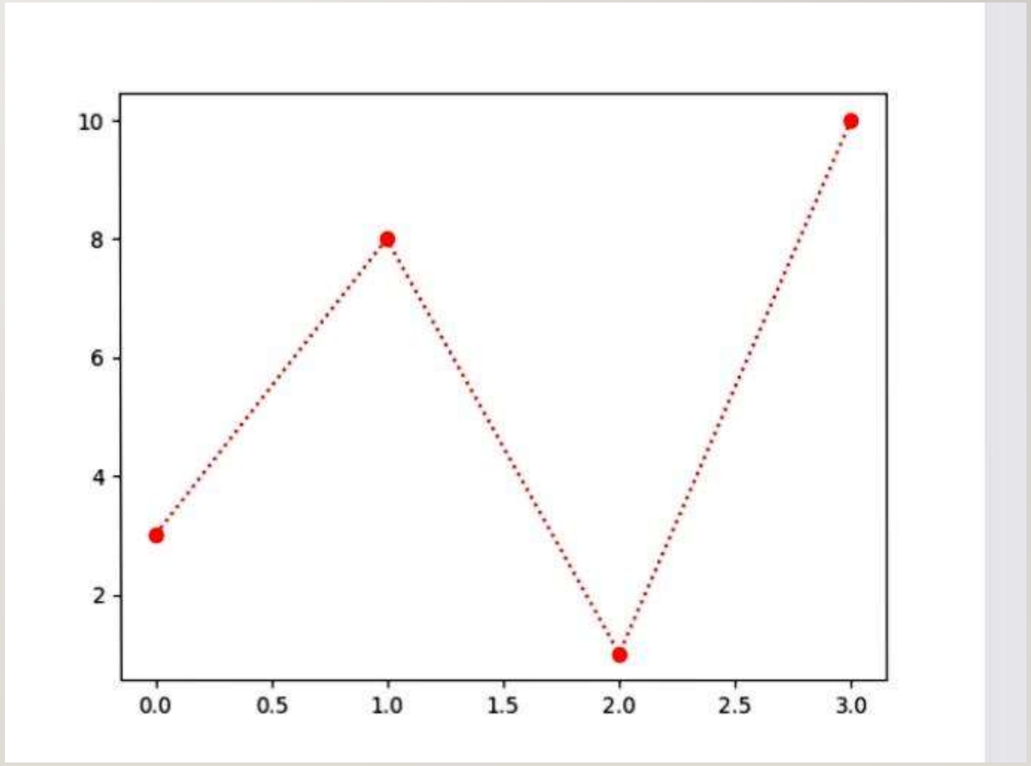


Format Strings fmt

-
- You can use also use the shortcut string notation parameter to specify the marker.
- This parameter is also called fmt, and is written with this syntax:
- **marker|line|color**

Example

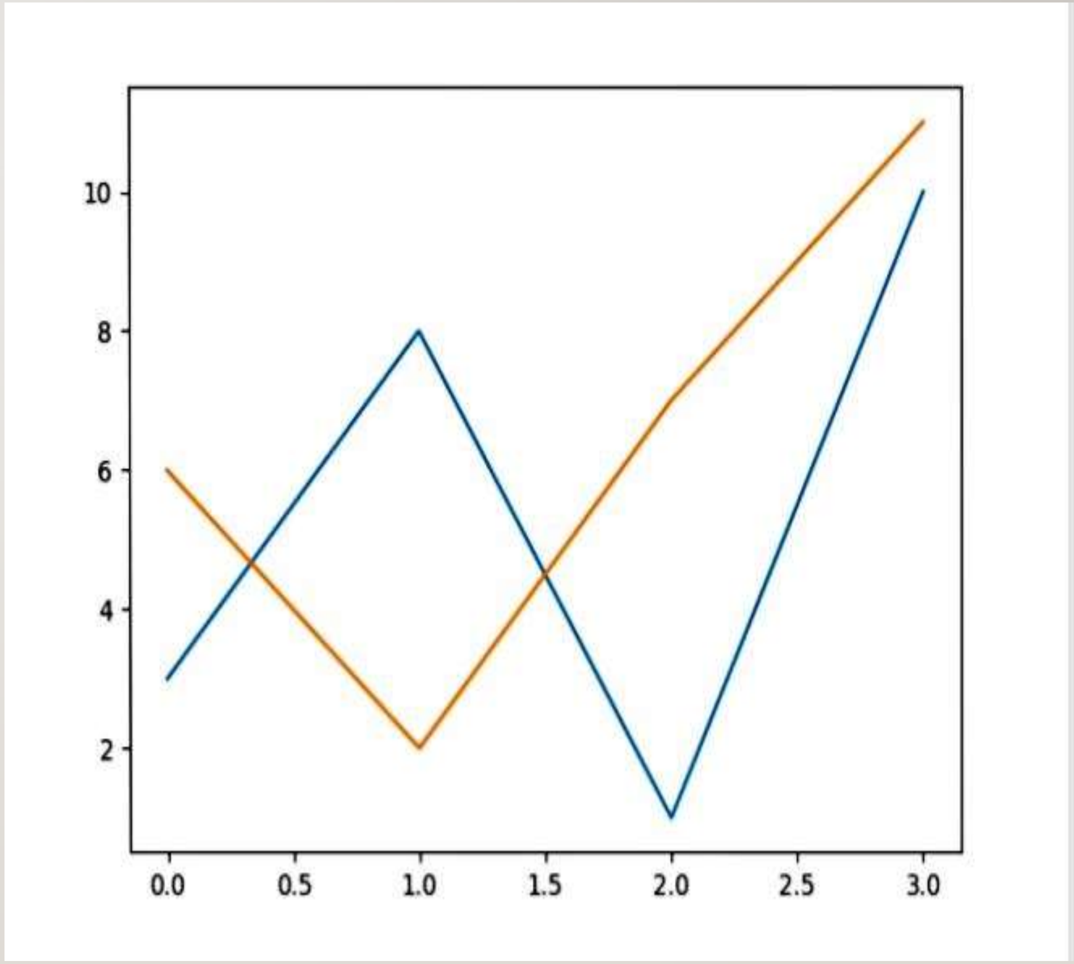
```
Import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, ‘o:r’)
plt.show()
```



Example

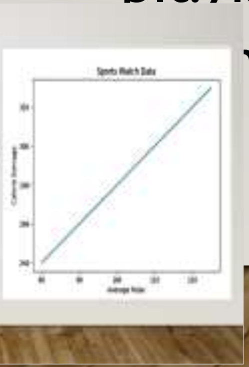
Draw two lines by specifying a plt.plot() function for each line:

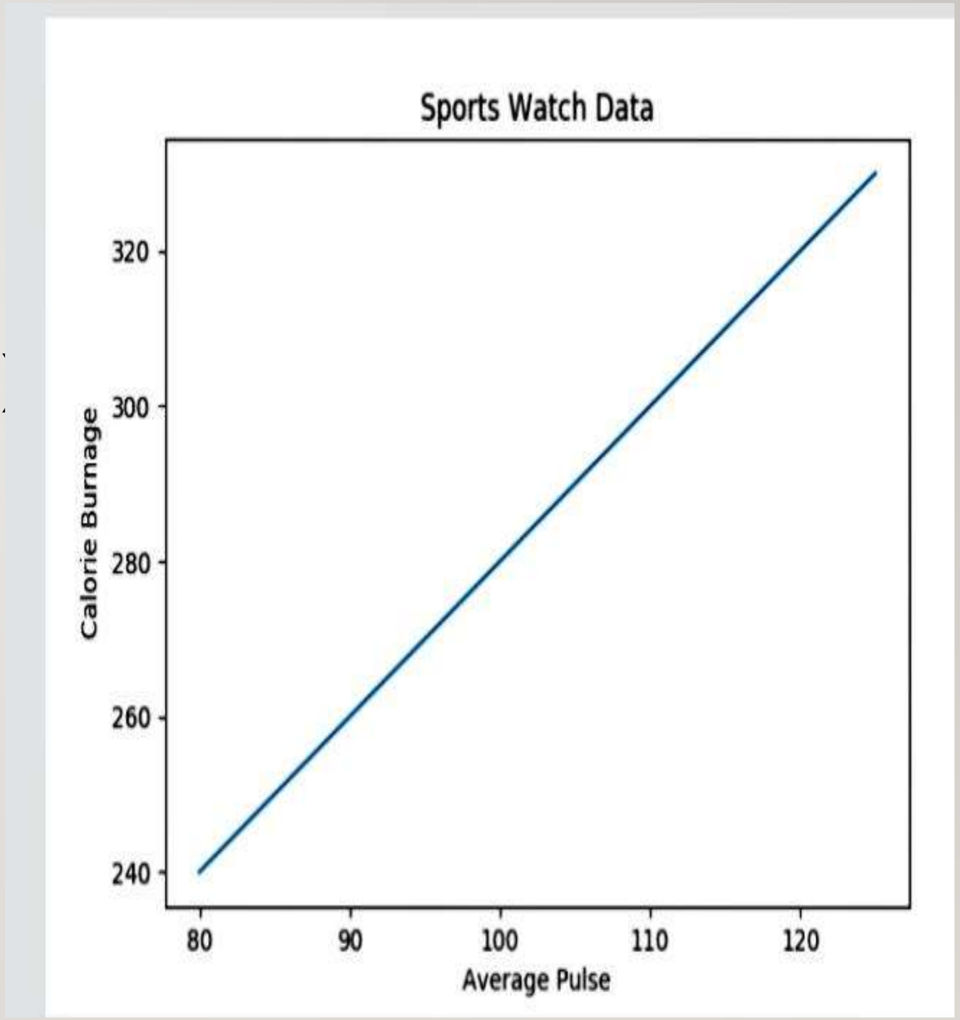
```
import matplotlib.pyplot as plt
import numpy as np

y1 = np.array([3, 8, 1, 10])
y2 = np.array([6, 2, 7, 11])
plt.plot(y1)
plt.plot(y2)
plt.show()
```



Matplotlib Labels and Title

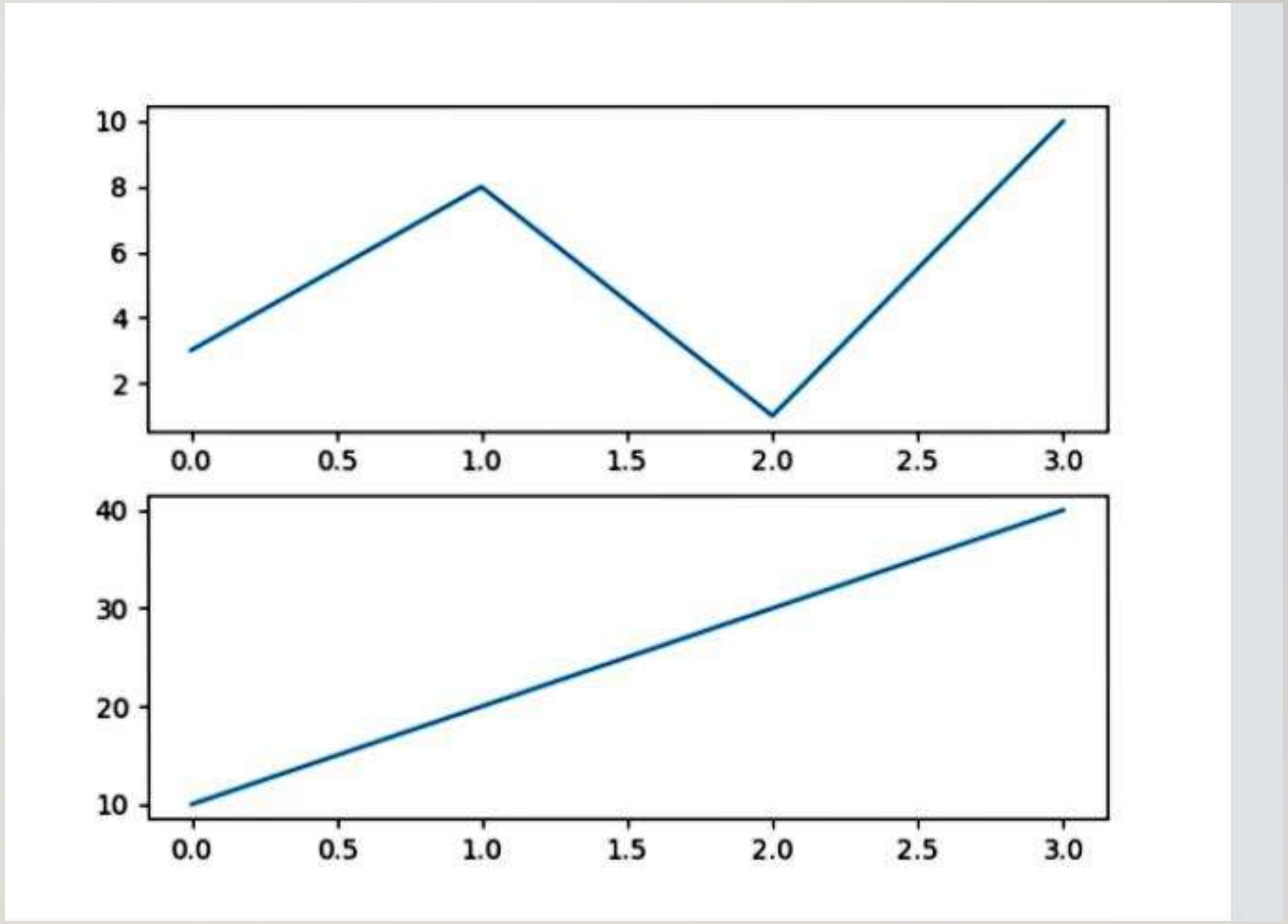
```
Import numpy as np
import matplotlib.pyplot as plt

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.plot(x, y)
plt.title(“Sports Watch Data”)
plt.xlabel(“Average Pulse”)
plt.ylabel(“Calorie Burnage”)
      now()
```





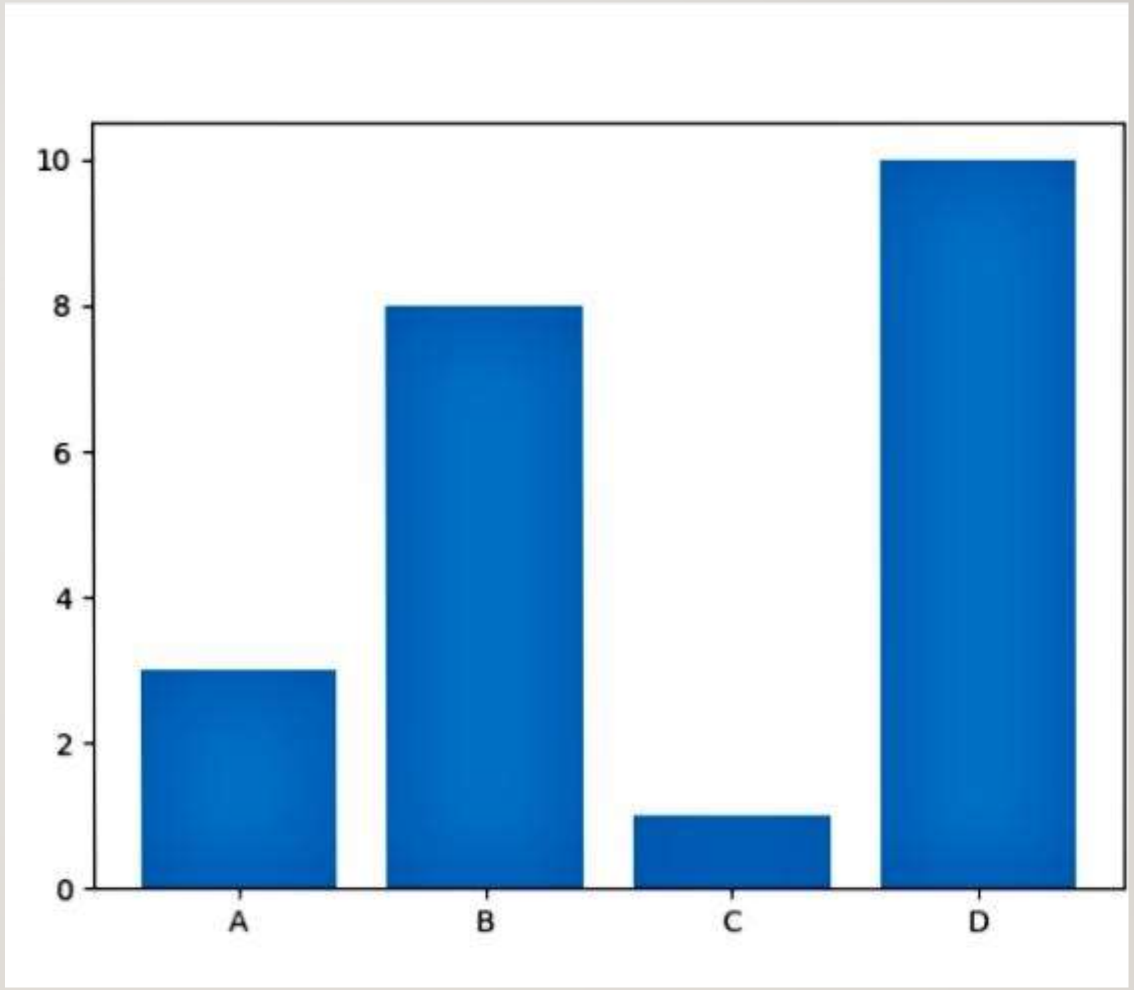
Draw 2 plots on top of each other:

```
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()
```



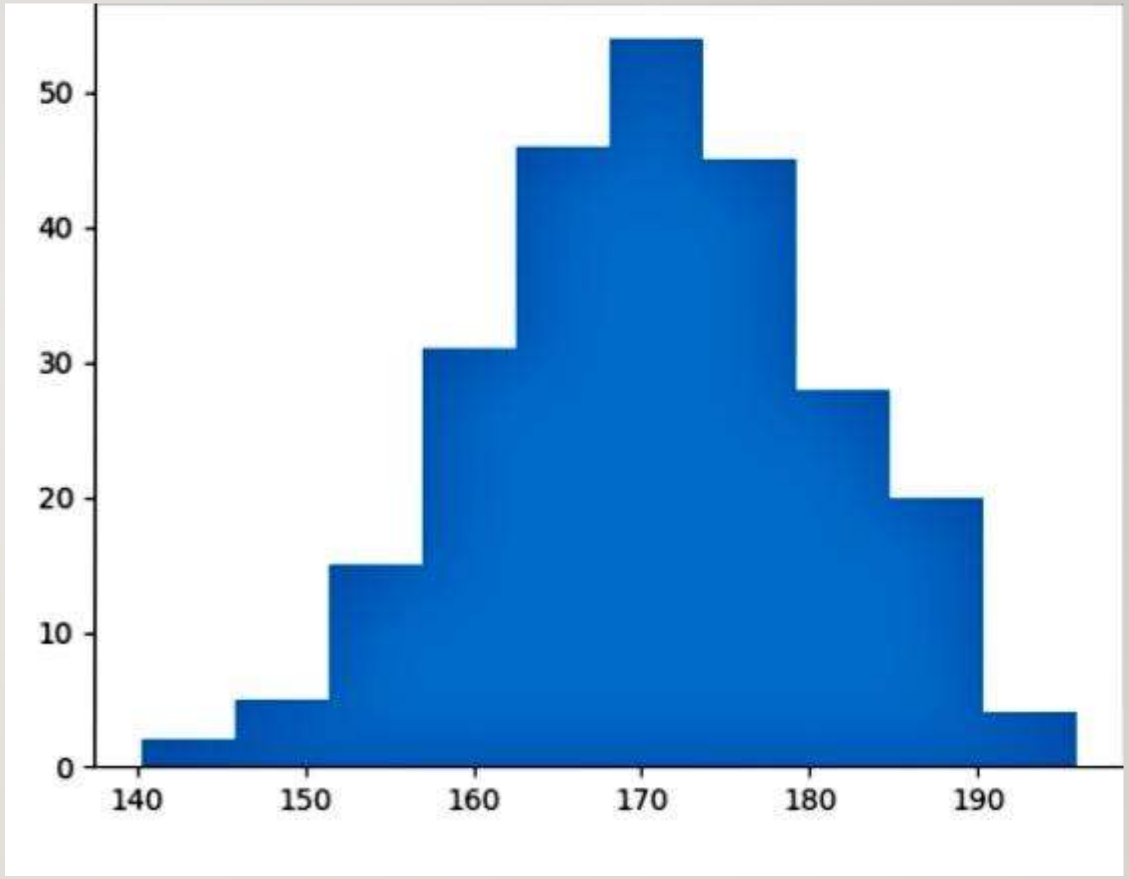
Bars

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

x = np.array([“A”,“B”,“C”,“D”])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
```



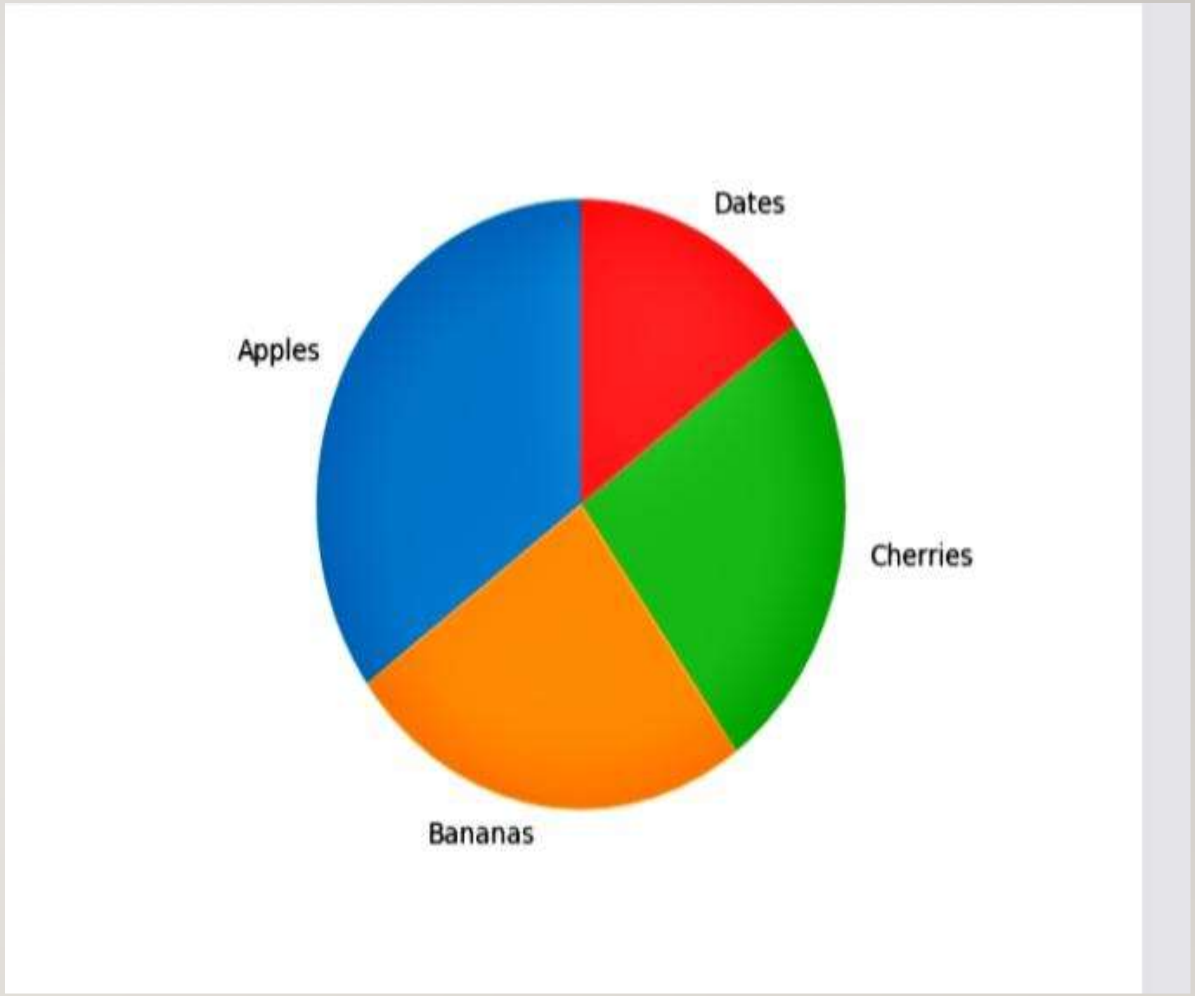
Matplotlib Histograms

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

x = np.random.normal(170, 10, 250)

plt.hist(x)
plt.show()
```



Matplotlib Pie Charts

```
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, startangle = 90)
plt.show()
```



CONCLUSION

01 **These libraries provide the user with the necessary tools to solve complex problems.**

02 **Python offers a wide range of libraries for various purposes such as numerical computing, data analysis, visualization and machine learning.**

INTRODUCTION TO MORE LIBRARIES AND APPLICATION

Python is a general-purpose programming language that is becoming increasingly popular for a wide variety of applications. It is known for its clear syntax, its powerful libraries, and its large community of developers.

Python libraries are used in vast varieties of applications as web development, data science, machine learning, automation, and game development.

WEB DEVELOPMENT

Python is widely used in web development, thanks to its simplicity and extensive libraries. Some key Python web development frameworks and tools include:

- **Django:** A high-level web framework for building robust and scalable web applications.
- **Flask**: A lightweight framework for creating smaller web applications and APIs.
- **Pyramid:** A flexible framework suitable for projects of all sizes.
- **FastAPI:** A modern, fast, and easy-to-use web framework for building APIs.
- **Beautiful Soup:** A library for web scraping and extracting data from HTML and XML

DATA SCIENCE AND MACHINE LEARNING

Python has become the go-to language for data science and machine learning due to its powerful libraries and ecosystem. Some prominent Python libraries for data science and machine learning include:

- **NumPy:** A fundamental library for numerical computing, providing powerful multi-dimensional arrays and mathematical functions.
- **Pandas:** A versatile library for data manipulation, analysis, and preprocessing.
- **scikit-learn**: A comprehensive library for machine learning, featuring various algorithms and tools for classification, regression, clustering, and more.
- **TensorFlow and Keras:** Keras is the high-level API of the TensorFlow platform. It provides an approachable, highly-productive interface for solving machine learning (ML) problems, with a focus on modern deep learning. Popular deep learning libraries for building and training neural networks.
- **PyTorch:** A flexible deep learning library known for its dynamic computation graphs

SCIENTIFIC COMPUTING

Python excels in scientific computing and simulations. Some notable Python libraries for scientific computing include:

- **SciPy:** A library for scientific and technical computing, offering functions for optimization, interpolation, linear algebra, and more.
- **Matplotlib:** A powerful plotting library for creating static, animated, and interactive visualizations.
- **Plotly:** A library for creating interactive and dynamic visualizations.
- **SymPy:** A symbolic mathematics library for symbolic computation and algebraic manipulations.
- **OpenCV:** A computer vision library with tools for image and video processing, feature detection, and more

GAME DEVELOPMENT

Python can also be used for game development, thanks to its simplicity and various game development libraries. Some popular Python game development libraries and frameworks include:

- **Pygame:** A cross-platform library for creating 2D games.
- **Panda3D:** A game engine and framework for creating 3D games.
- **Pyglet:** A library for creating games, interactive applications, and media-driven applications.
- **Arcade:** A simple and beginner-friendly library for creating 2D games.

OTHER APPLICATIONS

Apart from the mentioned applications, Python is utilized in many other domains, including:

- Scripting and automation
- Desktop application development
- Network programming
- Internet of Things (IoT)
- Natural language processing (NLP)
- Robotics

CONCLUSION

Python's versatility and rich ecosystem make it an ideal choice for a wide range of applications. From web development to machine learning and scientific computing, Python empowers developers to create efficient, scalable, and innovative solutions.

AI-POWERED PDF SUMMARY GENERATOR

ABSTRACT:-

Creating a Python application that uses Natural Language Processing (NLP) techniques to analyze and summarize the content of PDF documents. The application will read PDF files, extract text content, process the text using NLP, and generate concise summaries of the documents. The summaries will be saved as PDF

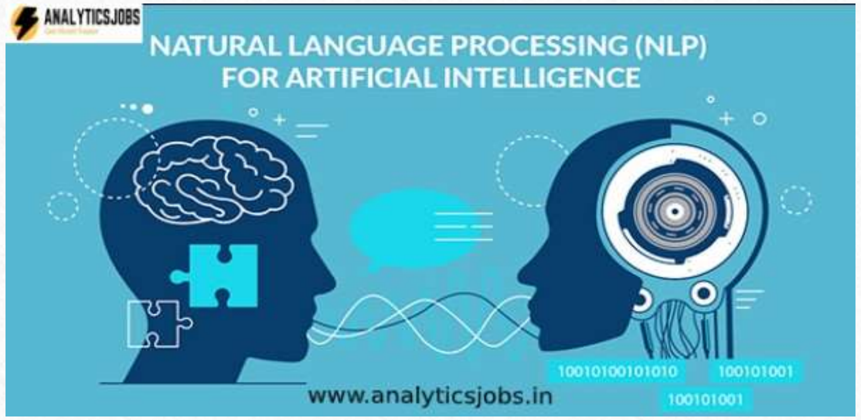
INTRODUCTION

- A PDF summary generator powered by AI is sophisticated application that utilise ai and natural language processing technique to automatically create concise and coherent summaries from lengthy pdf document. This technology aim to assists user in efficiently extracting key information and insight from large text ,improving productivity , etc. Traditional method of summarization often involve mannual effort in subjective decision. AI powered pdf summary generator on the other hand employ advance algorithm to analyse the content of PDF document, identify important passages, and condense them into coherent summarize while preseving the essential meaning.

Natural Language Processing – Definition Uses & techniques

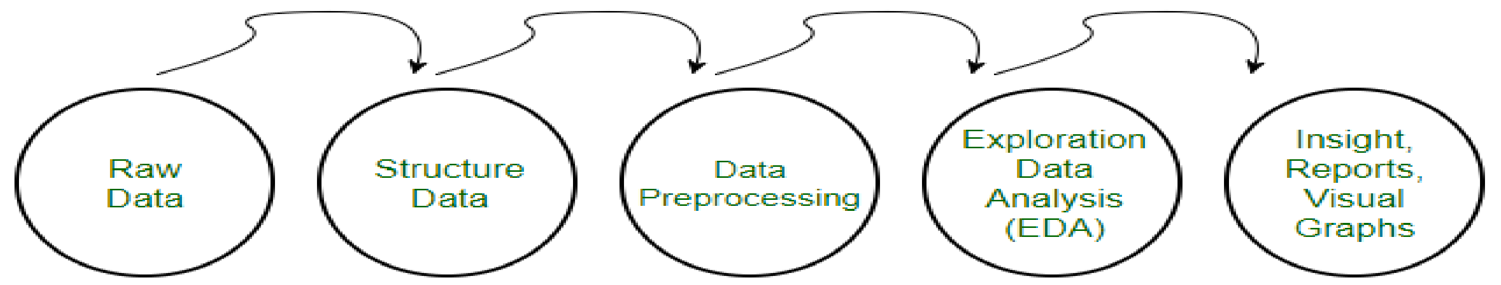
What is Natural Language Processing (NLP)?

Natural Language Processing, generally abbreviated as NLP, is a part of artificial intelligence that manages the association among PCs and humans using the natural language. A definitive target of NLP is to peruse, decode, comprehend, and understand the human language in a way that is important.



Data Preprocessing

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.


Raw Data
Structure Data
Data Preprocessing
Exploration Data Analysis (EDA)
Insight, Reports, Visual Graphs

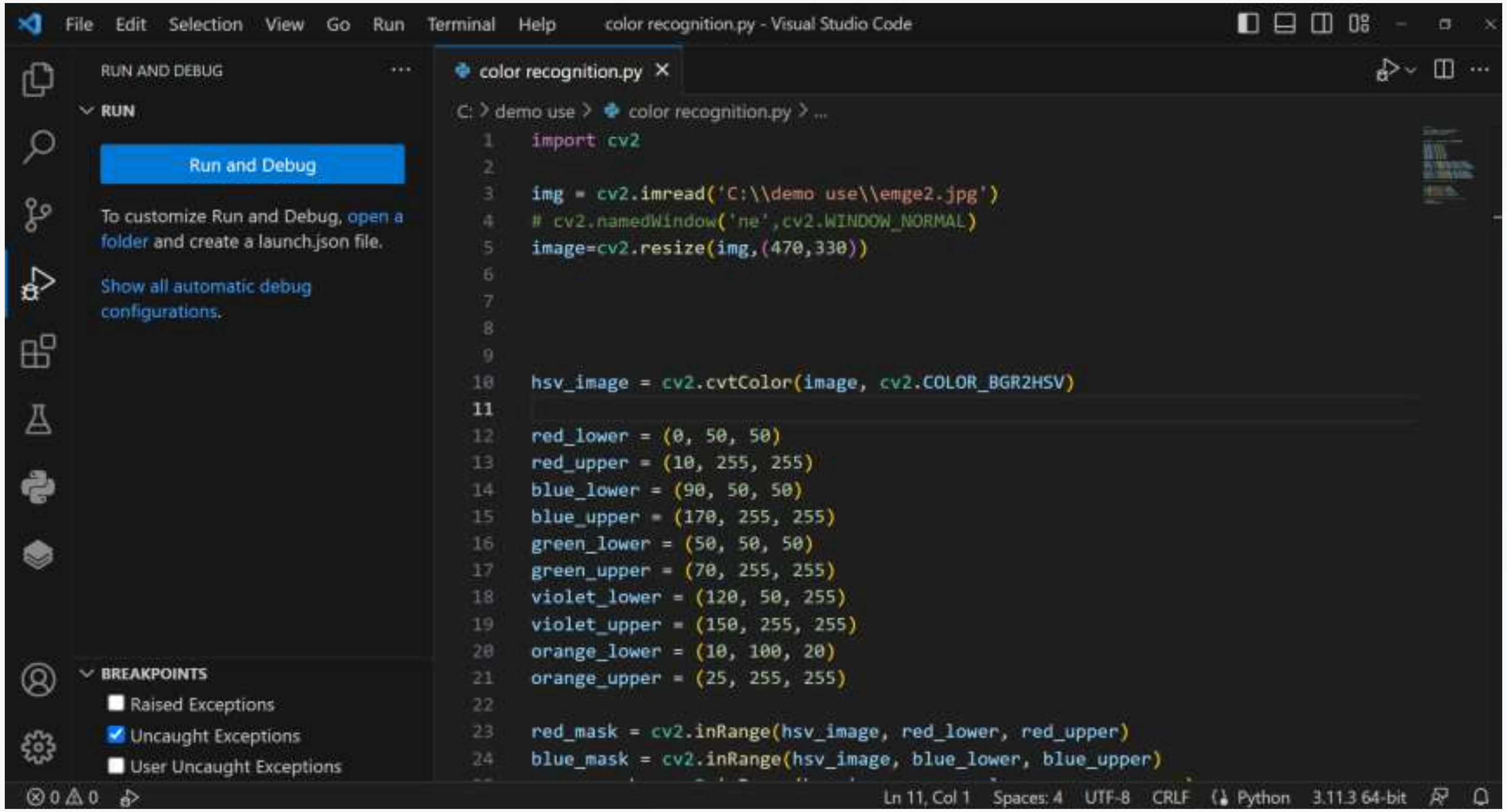
Color Recognition

OpenCV Library

For python program,we use OpenCV (Open Source Computer Vision Library) which is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

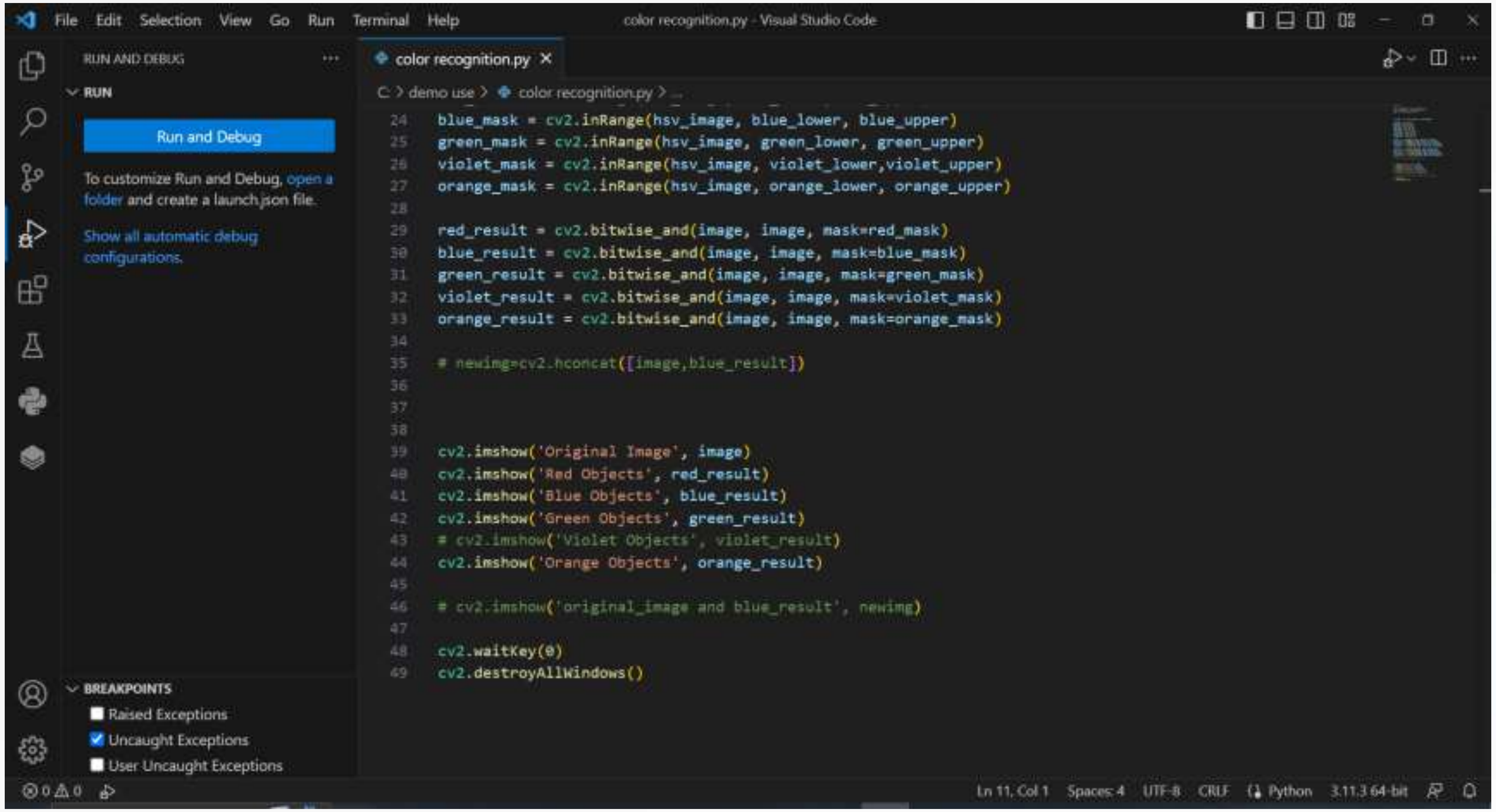
It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Color Recognition


File Edit Selection View Go Run Terminal Help
color recognition.py - Visual Studio Code
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
color recognition.py
C: > demo use > color recognition.py > ...
Ln 11, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit

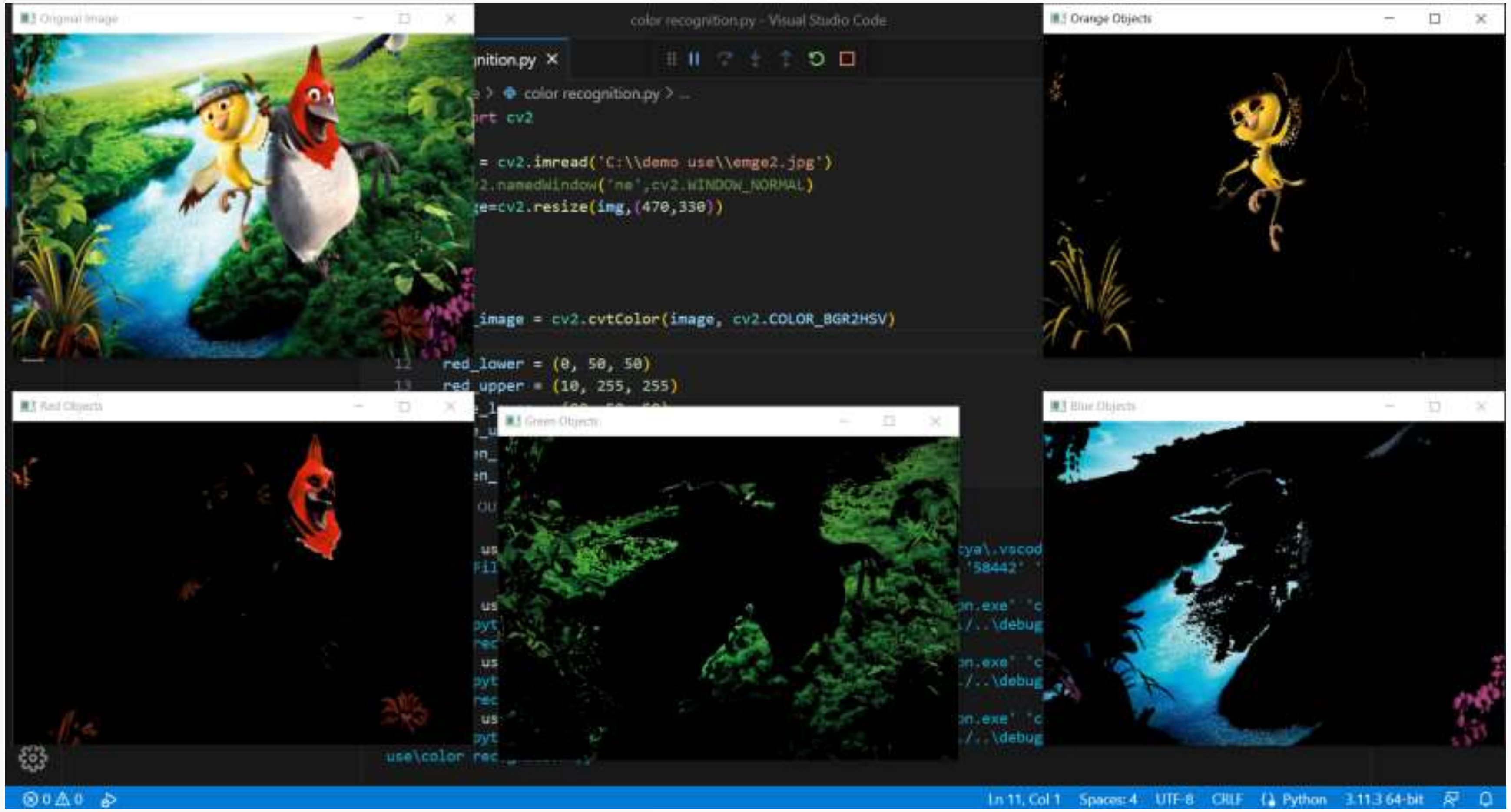
```python
import cv2

img = cv2.imread('C:\\demo use\\emge2.jpg')
# cv2.namedWindow('ne',cv2.WINDOW_NORMAL)
image=cv2.resize(img,(470,330))




hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

red_lower = (0, 50, 50)
red_upper = (10, 255, 255)
blue_lower = (90, 50, 50)
blue_upper = (170, 255, 255)
green_lower = (50, 50, 50)
green_upper = (70, 255, 255)
violet_lower = (120, 50, 255)
violet_upper = (150, 255, 255)
orange_lower = (10, 100, 20)
orange_upper = (25, 255, 255)

red_mask = cv2.inRange(hsv_image, red_lower, red_upper)
blue_mask = cv2.inRange(hsv_image, blue_lower, blue_upper)
```

Color Recognition


File Edit Selection View Go Run Terminal Help
color recognition.py - Visual Studio Code
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
color recognition.py
C > demo use > color recognition.py > ...
Ln 11, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit

```
blue_mask = cv2.inRange(hsv_image, blue_lower, blue_upper)
green_mask = cv2.inRange(hsv_image, green_lower, green_upper)
violet_mask = cv2.inRange(hsv_image, violet_lower,violet_upper)
orange_mask = cv2.inRange(hsv_image, orange_lower, orange_upper)

red_result = cv2.bitwise_and(image, image, mask=red_mask)
blue_result = cv2.bitwise_and(image, image, mask=blue_mask)
green_result = cv2.bitwise_and(image, image, mask=green_mask)
violet_result = cv2.bitwise_and(image, image, mask=violet_mask)
orange_result = cv2.bitwise_and(image, image, mask=orange_mask)

# newimg=cv2.hconcat([image,blue_result])



cv2.imshow('Original Image', image)
cv2.imshow('Red Objects', red_result)
cv2.imshow('Blue Objects', blue_result)
cv2.imshow('Green Objects', green_result)
# cv2.imshow('Violet Objects', violet_result)
cv2.imshow('Orange Objects', orange_result)

# cv2.imshow('original_image and blue_result', newimg)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Color Recognition


Original Image
color recognition.py - Visual Studio Code
Orange Objects
Red Objects
Green Objects
Blue Objects
Ln 11, Col 1   Spaces: 4   UTF-8   CRLF   Python   3.11.3 64-bit

```
rt cv2

 = cv2.imread('C:\\demo use\\emge2.jpg')
2.namedWindow('ne',cv2.WINDOW_NORMAL)
e=cv2.resize(img,(470,330))



_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

red_lower = (0, 50, 50)
red_upper = (10, 255, 255)
```

Here color detection algorithm works by identifing pixels(A pixel is the smallest unit of a digital image or graphic that can be displayed and represented on a digital display device. Image pixels are numerical values that represent color intensities in images) in an image that match a specified color or color range. The color of detected pixels can then be changed to distinguish them from the rest of the image.

The keywords and functions used in the program are:

- ❑cv2.imread
- ❑cv2.resize
- ❑cv2.cvtColor
- ❑cv2.inRange
- ❑cv2.bitwise
- ❑cv2.imshow
- ❑cv2.waitKEY
- ❑cv2.destroyAllWindows