PYTHON UNIT 5

Abhay Kumar Singh
www.abhaysingh722@gmail.com

NumPy

- NumPy is a powerful library for numerical computing in Python,
- it provides a wide range of built-in functions for various mathematical operations.

NumPy operation

1.Array Creation and Manipulation

```
# importing the numpy module
import numpy as np
# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])
# Print the array
print("Original Array:", arr)

# Perform some basic operations
print("Sum of Array Elements:", np.sum(arr))
print("Mean of Array Elements:", np.mean(arr))
print("Maximum Element in Array:", np.max(arr))
print("Minimum Element in Array:", np.min(arr))
# Reshape the array
reshaped_arr = arr.reshape(1, 5)
print("Reshaped Array:", reshaped_arr)
```

```
output:
Original Array: [1 2 3 4 5]
Sum of Array Elements: 15
Mean of Array Elements: 3.0
Maximum Element in Array: 5
Minimum Element in Array: 1
Reshaped Array: [[1 2 3 4 5]]
```

2.Array Operations

```
import numpy as np
# Create two NumPy arrays
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

# Perform array operations
print("Element-wise Sum:", np.add(arr1, arr2))
print("Element-wise Subtraction:", np.subtract(arr1, arr2))
print("Element-wise Multiplication:", np.multiply(arr1, arr2))
print("Element-wise Division:", np.divide(arr1, arr2))
```

```
output:
Element-wise Sum: [5 7 9]
Element-wise Subtraction: [-3 -3 -3]
Element-wise Multiplication: [ 4 10 18]
Element-wise Division: [0.25 0.4  0.5 ]
```

3.Statistical Operations

```
import numpy as np

# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])

# Perform statistical operations
print("Standard Deviation:", np.std(arr))
print("Variance:", np.var(arr))
print("Median:", np.median(arr))
print("Percentile (50th):", np.percentile(arr, 50))
```

```
output:
Standard Deviation: 1.4142135623730951
Variance: 2.0
Median: 3.0
Percentile (50th): 3.0
```

3.Linear Algebra Operations

```
import numpy as np
# Define two matrices
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])

# Perform matrix multiplication
result = np.dot(matrix1, matrix2)
print("Matrix Multiplication Result:")
print(result)

# Compute matrix determinant
det = np.linalg.det(matrix1)
print("Determinant of Matrix1:", det)

# Compute matrix inverse
inverse = np.linalg.inv(matrix1)
print("Inverse of Matrix1:")
print(inverse)
```

```
output:
Matrix Multiplication Result:
[[19 22]
 [43 50]]
Determinant of Matrix1: -2.0000000000000004
Inverse of Matrix1:
[[-2.   1. ]
 [ 1.5 -0.5]]
```

Pandas

- Pandas is a powerful library for data manipulation and analysis in Python.
- It provides data structures like DataFrame and Series, as well as functions to manipulate and analyze them efficiently.

```
import pandas as pd
# Creating a DataFrame from a dictionary
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [25, 30, 35, 40],
        'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']}
df = pd.DataFrame(data)

# Display the DataFrame
print("DataFrame:")
print(df)
```

output:

```
DataFrame:
     Name  Age         City
0   Alice   25     New York
1     Bob   30  Los Angeles
2 Charlie   35      Chicago
3   David   40      Houston
```

```
# Accessing specific columns
print("\nAccessing 'Name' column:")
print(df['Name'])
output:
```

```
DataFrame:
     Name  Age         City
0   Alice   25     New York
1     Bob   30  Los Angeles
2 Charlie   35      Chicago
3   David   40      Houston
```

```
# Accessing specific rows
print("\nAccessing first row:")
print(df.iloc[0])
output:
```

```
Accessing first row:
Name        Alice
Age            25
City     New York
Name: 0, dtype: object
```

```
# Adding a new column
df['Gender'] = ['Female', 'Male', 'Male', 'Male']
print("\nDataFrame after adding 'Gender' column:")
print(df)
output:
```

```
DataFrame after adding 'Gender' column:
     Name  Age         City  Gender
0   Alice   25     New York  Female
1     Bob   30  Los Angeles    Male
2 Charlie   35      Chicago    Male
3   David   40      Houston    Male
```

```
# Filtering rows based on a condition
print("\nRows where Age > 30:")
print(df[df['Age'] > 30])
output:
```

```
Rows where Age > 30:
   Name  Age     City Gender
2 Charlie   35  Chicago   Male
3   David   40  Houston   Male
```

1.Reading and Writing Data

```
import pandas as pd
# Reading data from a CSV file
df = pd.read_csv('data.csv')
# Display the DataFrame
print("DataFrame read from CSV:")
print(df)
# Writing data to a CSV file
df.to_csv('output.csv', index=False)
print("DataFrame written to CSV.")
```

output:

```
DataFrame read from CSV:
     Name  Age         City
0   Alice   25     New York
1     Bob   30  Los Angeles
2 Charlie   35      Chicago
3   David   40      Houston
DataFrame written to CSV.
```

2.Data Analysis

```
import pandas as pd
# Creating a DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [25, 30, 35, 40],
        'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']}
df = pd.DataFrame(data)
# Descriptive statistics
print("Descriptive statistics:")
print(df.describe())
```

output:

```
Descriptive statistics:
             Age
count   4.000000
mean   32.500000
std     6.454972
min    25.000000
25%    28.750000
50%    32.500000
75%    36.250000
max    40.000000
```

```
# Grouping
print("\nMean age by city:")
print(df.groupby('City')['Age'].mean())
```

output:

```
Mean age by city:
City
Chicago        35.0
Houston        40.0
Los Angeles    30.0
New York       25.0
Name: Age, dtype: float64
```

Matplotlib

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- Matplotlib makes easy things easy and hard things possible.
- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.

```
import matplotlib
```

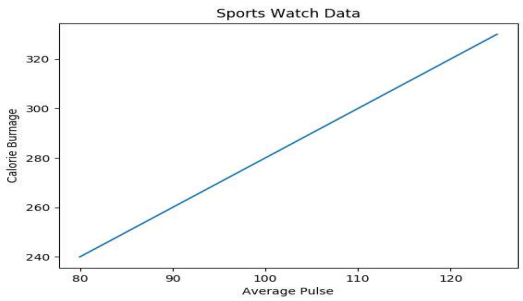
1.Pyplot

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias

```
import matplotlib.pyplot as plt
```

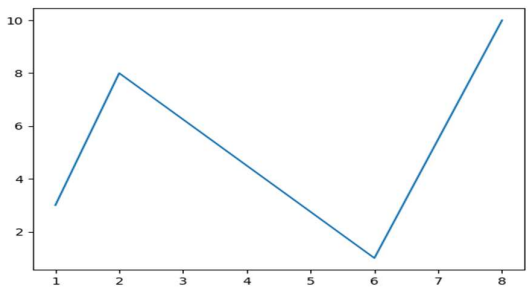
2.line plot:

```
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.plot(x, y)

plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.show()
```



3.Multiple Points

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

xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])
plt.plot(xpoints, ypoints)
plt.show()
```

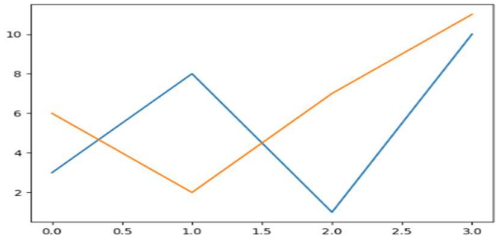


4.Markers

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

5.Multiple Lines

```
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)
```

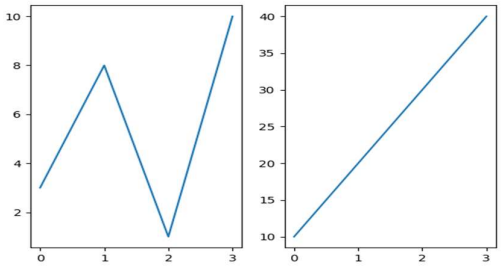



```
plt.show()
```

6.Display Multiple Plots

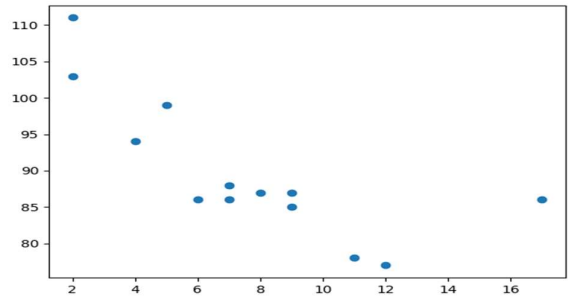
```
import matplotlib.pyplot as plt
import numpy as np

#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)

#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.show()
```

```
plt.subplot(1, 2, 1)
#the figure has 1 row, 2 columns, and this plot is the first plot.

plt.subplot(1, 2, 2)
#the figure has 1 row, 2 columns, and this plot is the second plot.
```



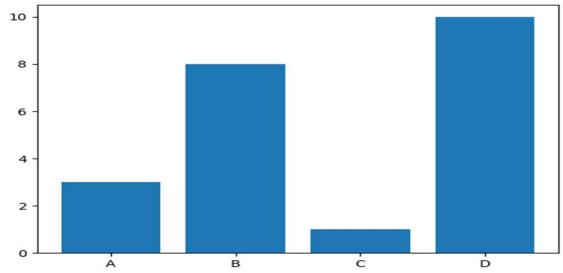
7.scatterplot

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

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)
plt.show()
```



8.bar plot

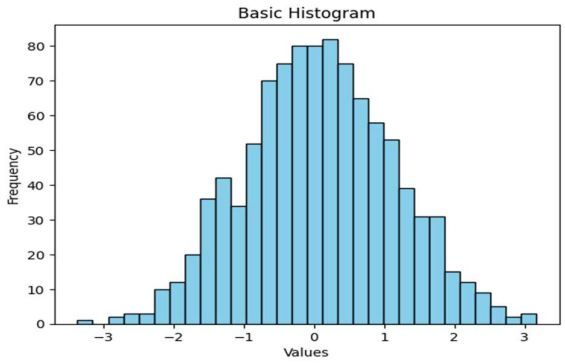
```
import matplotlib.pyplot as plt
import numpy as np

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

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



9.Histogram

- histogram represents data provided in the form of some groups.
- It is an accurate method for the graphical representation of numerical data distribution.

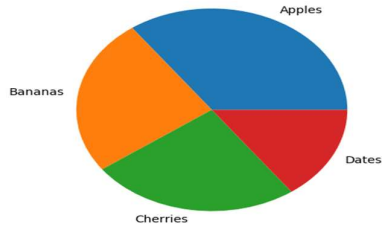
```
import matplotlib.pyplot as plt
import numpy as np

# Generate random data for the histogram
data = np.random.randn(1000)
# Plotting a basic histogram
plt.hist(data, bins=30, color='skyblue', edgecolor='black')

# Adding labels and title
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Basic Histogram')
# Display the plot
plt.show()
```



10.Pie Charts

- With Pyplot, you can use the pie() function to draw 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)
plt.show()
```



1.Draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10):

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

GUI programming

- GUI programming involves creating visual interfaces for user interaction.
- It utilizes graphical elements like windows, buttons, menus, and textboxes.
- Users interact with applications through these graphical components.
- Libraries or frameworks are used to design and implement GUIs.
- GUIs provide a more intuitive and user-friendly way to interact with software.
- They are used in desktop applications, mobile apps, games, and data visualization tools.
- GUI programming enhances the user experience and accessibility of software.

- It is essential for creating visually appealing and interactive applications.

Tkinter

- Tkinter is Python's built-in GUI toolkit for creating desktop applications.
- It offers a wide range of GUI widgets such as buttons, labels, and textboxes.
- GUI elements are arranged using layout managers like pack, grid, and place.
- Tkinter follows an event-driven programming paradigm, responding to user actions with specific functions.
- It allows customization of widget appearance and behavior.
- Tkinter applications are highly portable across various operating systems.
- Suitable for small to medium-sized GUI projects and rapid prototyping.
- Comprehensive documentation and community support are available.
- While Tkinter is sufficient for many applications, more advanced libraries like PyQt and wxPython offer additional features and scalability for larger projects.

Tkinter widgets

1.Button: A **clickable** button that triggers an action when pressed.

```
import tkinter as tk

def button_clicked():
    print("Button clicked!")

root = tk.Tk()
button = tk.Button(root, text="Click Me", command=button_clicked)
button.pack()
root.mainloop()
```

2.Label: **Static** text or image that provides information or instructions to the user.

```
import tkinter as tk

root = tk.Tk()
label = tk.Label(root, text="Hello, Tkinter!")
label.pack()
root.mainloop()
```

- **Entry**: Single-line text entry field where users can input text or numbers.

```
import tkinter as tk

def get_entry_text():
    print(entry.get())

root = tk.Tk()
entry = tk.Entry(root)
entry.pack()
button = tk.Button(root, text="Get Text", command=get_entry_text)
button.pack()
root.mainloop()
```

- **Text**: Multi-line text entry field for longer text input or display.

```
import tkinter as tk

def get_text_content():
    print(text.get("1.0", "end-1c"))

root = tk.Tk()
text = tk.Text(root)
text.pack()
button = tk.Button(root, text="Get Text",
command=get_text_content)
button.pack()
root.mainloop()
```

- **Checkbutton**: Checkbox widget that allows users to select or deselect an option.

```
import tkinter as tk

root = tk.Tk()
var = tk.IntVar()
checkbutton = tk.Checkbutton(root, text="Check me",
variable=var)
checkbutton.pack()
root.mainloop()
```

- **Radiobutton**: Group of radio buttons where only one option can be selected at a time.

```
import tkinter as tk

root = tk.Tk()
var = tk.StringVar()
radiobutton1 = tk.Radiobutton(root, text="Option 1",
variable=var, value="Option 1")
radiobutton2 = tk.Radiobutton(root, text="Option 2",
variable=var, value="Option 2")
radiobutton1.pack()
radiobutton2.pack()
root.mainloop()
```

- **Listbox:** Widget for displaying a list of items from which users can select one or more.

```
import tkinter as tk

root = tk.Tk()
listbox = tk.Listbox(root)
for item in ["Item 1", "Item 2", "Item 3"]:
    listbox.insert(tk.END, item)
listbox.pack()
root.mainloop()
```

- **Frame:** Container widget used to organize and group other widgets together.

```
import tkinter as tk
root = tk.Tk()
root.title("Frame Example")

# Create a frame
frame = tk.Frame(root, width=200, height=100, bg="lightblue")
frame.pack()

# Add widgets to the frame
label = tk.Label(frame, text="This is a frame", font=("Arial", 18))
label.pack(pady=10)

button = tk.Button(frame, text="Click Me", command=lambda:
print("Button clicked!"))
button.pack()

root.mainloop()
```

- **Menu**: Menu bar, dropdown menus, and context menus for creating menu-driven interfaces.

```
import tkinter as tk
def menu_command():
    print("Menu command executed")
root = tk.Tk()
# Create a menu
menu_bar = tk.Menu(root)

# Create a file menu
file_menu = tk.Menu(menu_bar, tearoff=0)
file_menu.add_command(label="New",
command=menu_command)
file_menu.add_command(label="Open",
command=menu_command)
file_menu.add_separator()
file_menu.add_command(label="Exit", command=root.quit)

# Add the file menu to the menu bar
menu_bar.add_cascade(label="File", menu=file_menu)

# Create an edit menu
edit_menu = tk.Menu(menu_bar, tearoff=0)
edit_menu.add_command(label="Cut",
command=menu_command)
edit_menu.add_command(label="Copy",
command=menu_command)
edit_menu.add_command(label="Paste",
command=menu_command)

# Add the edit menu to the menu bar
menu_bar.add_cascade(label="Edit", menu=edit_menu)
# Display the menu bar
root.config(menu=menu_bar)

root.mainloop()
```