Week 1

Created	@January 11, 2025 7:25 PM

Python :

1. Random module :

- Generating random number from range

```
import random
print(random.randrange(3, 9))
#returns a number between 3 (included) and 9 (not included)
```

- Generating random number from 0 to 1

```
import random
print(random.random())
#returns 0.39581613905688395
```

- Generating random integer including range values

```
import random
print(random.randint(3, 9))
#returns a number between 3 and 9 (both included)
```

- To select a random element from a sequence

📌
```
import random

colors = ['red', 'blue', 'green', 'yellow']
color = random.choice(colors)
print(color)

 # Output: Could be any color from the list
```

2. List :

📌
```
list = []

len(list)  - Length of list

list.append(5) - Add element to list

list.sort() - Sort the list

list.reverse() - Reverse all elements

list.remove(5) - Remove the first occurence

list.insert(index, 5) - Inserts at specified index

list.pop(index) - Deletes the element in specified index
```

3. Function :

Syntax to define a function

📌
```
def sum :
    a = 5
    b = 1
    return a+b
```

4. Dictionary :

Dictionary items are ordered, changeable, and do not allow duplicates.

Dictionary items are presented in key:value pairs, and can be referred to by using the key name.

📌 dict = {}

dict = { 'Ram' : 30, 'Ravi' : 35, 'Raj' : 18}

dict['Swetha'] = 25 - To add items in dictionary

dict['Ram'] = 20 - To change value in dictionary

del dict['Ram'] - To delete an item in dictionary

dict.clear() - To remove all items in dictionary

Some inbuilt dictionary methods :

📌 dict.has_key('Ram') - returns True or False

dict.keys() - returns list of all keys ['Ram', 'Ravi', 'Raj']

dict.values() - retuens list of all values [30, 35, 18]

dict.items() - return list of each key-value pair in tuples [('Ram',30), ('Ravi',35), ('Raj',18)]

5. Matplotlib :

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

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

📌 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) - Plots the x,y values

plt.title("Sports Watch Data") - Gives title to the graph

plt.xlabel("Average Pulse") - Gives name to the X-axis

plt.ylabel("Calorie Burnage") - Gives name to the Y-axis

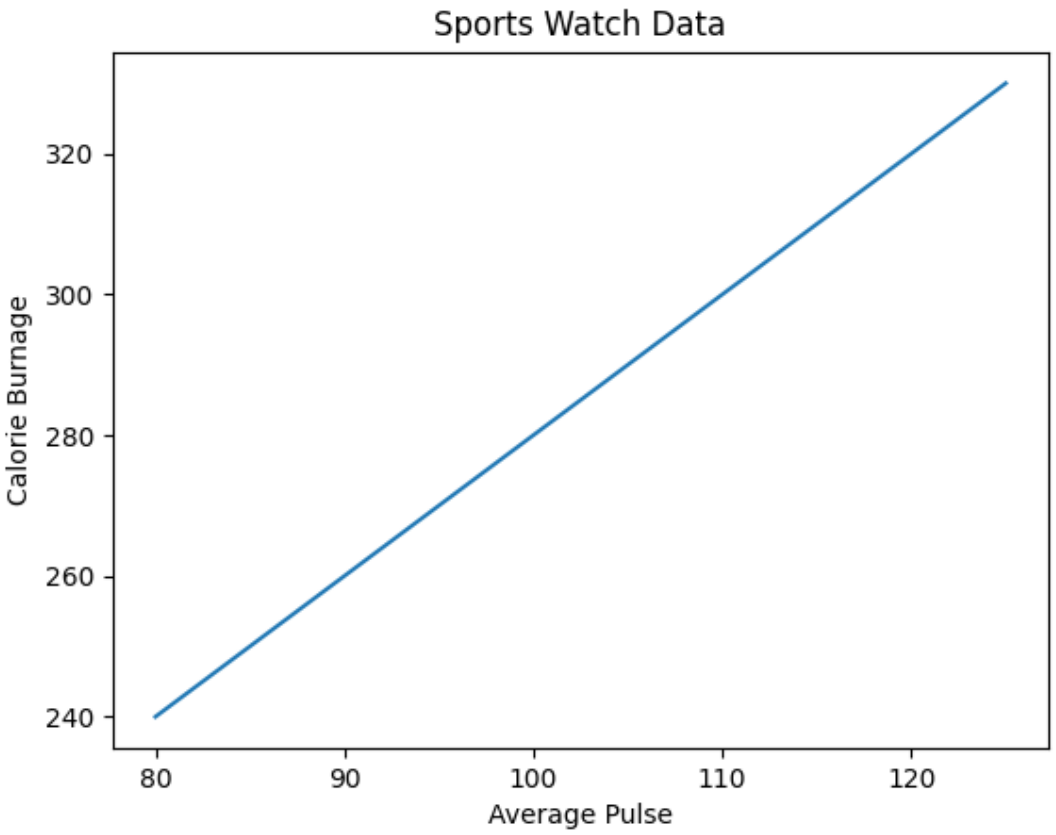
plt.show() - Shows the output

plt.scatter(x, y) - Just keeps a dot at the x,y value

plt.bar(x,y) - Draws bargraph

plt.bar(x,y) - Draws piechart

Output :



6. NetworkX :

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

📌
```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()                    # Creates a Undirected graph
G.add_node(1)                     # Used to add a node to graph
G.add_node(2)
G.nodes()                          # Returns list of all nodes in graph [1,2]
G.add_edge(1,2)                   # Used to add an edge from 1 to 2
G.add_edge(1,2, weight=100)       # Add edge with weight/cost
G.edges()                          # Returns list of all edges in graph [ (1,2)]
nx.draw(G, with_label=1)          # Draw the graph with label
plt.show()                         # Shows the output
```

📌
```
Z = nx.complete_graph(10)
# Creates a complete graph with 10 nodes
# ( each node is connected to all other nodes)

Z.order()  # Returns total number of nodes in graph - 10
Z.size()     # Returns total number of edges in graph - 45
```

📌
```
G = nx.gnp_random_graph(20, 0.5)
# Creates a graph with 20 nodes and
# each edge with 0.5 probability.

G = nx.DiGraph()          # Creates a directed graph
G.number_of_edges()    # Returns total number of edges
G.has_edge(1,2)            # Returns True or False

nx.is_connected(G)   # Returns true if graph is connected else False

# Connected graph - Every node is reachable from other node


nx.has_path(G,1,3)   # Returns True if there is a path from 1 to 3 else False
```

To get a good visualization of graph use :

```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.Graph()

pos = nx.circular_layout(G)
# Distribute the nodes circularly

nx.draw(G, pos)
plt.show()
```

Some inbuilt Path functions :

Assume graph G has several cities as nodes and edges with costs

1. dijkstra_path() :

```
nx.dijkstra_path(G, 'Chennai', 'Delhi')
# Returns the shortest path['Bangalore', 'Surat', 'Bombay'
```

2. dijkstra_path_length() :

```
nx.dijkstra_path_length(G, 'Chennai', 'Delhi')
# Returns the path cost from Chennai to Delhi - 2000
```