Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./Miss Pitharva Yadav

of IT Department, Semester III with

Roll No. 127 has completed a course of the necessary

experiments in the subject Python Lab under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 20 - 20

16/4/24

Teacher In-Charge

Head of the Department

Date 16/04/2024

Principal

THADOMAL SHAHANI ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY

PYTHON LABORATORY (ITL404) INDEX

Sr No	LAB ASSIGNMENT	LO MAPPED	DATE	SIGN
1	Variables & Data Types	LO1	15/1/24	
2	Operators and String functions	LO1	16/1	
3	List and Tuples operation	LO1, LO2	23/1	
4	Operation on Sets	LO1, LO2	29/1	
5	Operation on Dictionary	LO1, LO2	30/1	
6	Control Loop, Function and anonymous Function	LO1, LO2	5/2	
7	Classes and Objects	LO1, LO3	6/2	
8	Inheritance & its Types	LO1, LO3	12/2	
9	Polymorphism (Method Overloading/Overriding, Operator Overloading)	LO1, LO3	13/2	
10	Arrays operations	LO1, LO2	20/2	
11	File and Exception handling	LO1, LO5	26/2	
12	Linear Algebra using SciPy	LO1, LO6	27/2	
13	Data Visualization using Matplotlib	LO1, LO5	4/3	
14	Data Analysis using Pandas	LO1, LO5	5/3	
15	User Interface using Tkinter a. Login Form b. Scientific Calculator	LO1, LO5	18/3 19/3	
16	Written Assignment 1	LO1, LO2, LO3	28/3	
17	Written Assignment 2	LO4, LO5, LO6	28/3	

Smruty 16/4/24

Smruty 16/4/24

Smruty 16/4/24

Assignment 1

Variables and DataTypes

Executable Code:

```
#  Integer Variable
age = 25
print("Age:", age)

#  Floating-Point Variable
height = 5.9
print("Height:", height)

#  String Variable
name = "Om"
print("Name:", name)

#  Boolean Variable
is_student = True
print("Is Student?", is_student)

#  List - Ordered, Mutable
grades = [90, 85, 92, 88]
print("Grades:", grades)

#  Tuple - Ordered, Immutable
coordinates = (3, 4)
print("Coordinates:", coordinates)

#  Set - Unordered, Mutable, Unique Elements
hobbies = {"reading", "traveling", "coding"}
print("Hobbies:", hobbies)

#  Dictionary - Key-Value Pairs
person = {"name": "Alice", "age": 30, "city": "New York"}
print("Person:", person)

#  None Type
result = None
print("Result:", result)

#  Type Conversion
x = 5
y = str(x)
print("x as string:", y)
```

```python
#  Input from User
user_name = input("Enter your name: ")
print("Hello,", user_name)

#  Boolean Comparison
is_adult = age >= 18
print("Is Adult?", is_adult)

#  Arithmetic Operations
num1 = 10
num2 = 3
sum_result = num1 + num2
print("Sum:", sum_result)

#  String Concatenation
greeting = "Hello"
sentence = greeting + " " + name
print("Greeting Sentence:", sentence)

#  Formatting Strings
formatted_sentence = f"{name} is {age} years old."
print("Formatted Sentence:", formatted_sentence)

#  Basic If-Else Statement
if is_adult:
    print(f"{user_name} is an adult.")
else:
    print(f"{user_name} is not an adult.")
```

Output:

Age: 25

Height: 5.9

Name: Om

Is Student? True

Grades: [90, 85, 92, 88]

Coordinates: (3, 4)

Hobbies: {'reading', 'traveling', 'coding'}

Person: {'name': 'Alice', 'age': 30, 'city': 'New York'}

Result: None

x as string: 5

Enter your name:

Assignment 2

Strings and Operators

Executable Code:

Operators:

```
#Arithmetic Operators
x = 5
y= 3

print(x + y)
print(x - y)
print(x * y)
print(x / y)
print(x ** y)
print(x // y)
print(x % y)

#Assignment operators

x+=5
print(x)

x-=5
print(x)

x*=5
print(x)

x/=5
print(x)

x%=5
print(x)

x//=5
print(x)

x**=5
print(x)

x<<=5
```

```
print(x)

x>>=5
print(x)

#Comparision Operators

print(x == y)
print(x != y)
print(x > y)
print(x < y)
print(x >= y)
print(x <= y)

#Identity Operators

print(x is y)
print(x is not y)

#Membership operators

list1 = [10,20,30,40,50]

if x not in list1:
    print("X is not in list")
else:
    print("X is in list")

#Ternary Operator
min = a if a<b else b
print(min)
```

Output:

8

2

15

1.6666666666666667

125

1

2

10

5

25

5.0

0.0

0.0

0.0

False

True

False

True

False

True

False

True

X is not in list

10

Strings:

```python
#String Functions

s1 = "hello WOrld"

x = s1.capitalize()
print(f"Converts first char to capital: {x}")

x = s1.casefold()
print(f"Converts string to lower case: {x}")

x = s1.center(0)
print(f"Returns centered string: {x}")

x = s1.encode()
print(f"Encoded String: {x}")

x = s1.count("l")
print(f"Counts occurence of string: {x}")

x = s1.endswith("d")
print(f"Checks end of string: {x}")

x = s1.expandtabs()
print(f"Sets tab size of string: {x}")

x = s1.find("d")
print(f"Finds index of string: {x}")

x = s1.format("d")
print(f"Checks end of string: {x}")

x = s1.index("d")
print(f"Checks index of a char of string: {x}")

x = s1.isalnum()
print(f"Returns true if string is alphanumeric: {x}")

x = s1.isalpha()
print(f"Returns true if string is alphabetic: {x}")

x = s1.isascii()
print(f"Returns true if string is ascii: {x}")

x = s1.isalnum()
print(f"Returns true if string is alphanumeric: {x}")
```

```
x = s1.isdecimal()
print(f"Returns true if string is decimal: {x}")

x = s1.isidentifier()
print(f"Returns true if string is a identifier: {x}")
```

Output:

Converts first char to capital: Hello world

Converts string to lower case: hello world

Returns centered string: hello WOrld

Encoded String: b'hello WOrld'

Counts occurence of string: 3

Checks end of string: True

Sets tab size of string: hello WOrld

Finds index of string: 10

Checks end of string: hello WOrld

Checks index of a char of string: 10

Returns true if string is alphanumeric: False

Returns true if string is alphabetic: False

Returns true if string is ascii: True

Returns true if string is alphanumeric: False

Returns true if string is decimal: False

Returns true if string is a identifier: False

Process finished with exit code 0

Assignment 3

List and Tuples

Executable Code:

Lists:

```
# Example: Combining Lists
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]
list3 = list1 + list2
print(list3)

# Example: Appending Elements
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)

# Example: Clearing List
fruits = ["apple", "banana", "cherry"]
fruits.clear()
print(fruits)

# Example: Copying List
fruits = ["apple", "banana", "cherry"]
x = fruits.copy()
print(x)

# Example: Counting Elements
fruits = ["apple", "banana", "cherry", "cherry"]
x = fruits.count("cherry")
print(x)

# Example: Extending List
fruits = ["apple", "banana", "cherry"]
points = (1, 4, 5, 9)
fruits.extend(points)
print(fruits)

# Example: Index of Element
fruits = [4, 55, 64, 32, 16, 32]
x = fruits.index(32)
print(x)

# Example: Inserting Element
```

```
fruits = ['apple', 'banana', 'cherry']
fruits.insert(1, "orange")
print(fruits)

# Example: Popping Element
fruits = ['apple', 'banana', 'cherry']
x = fruits.pop(1)
print(x)

# Example: Removing Element
fruits = ['apple', 'banana', 'cherry']
fruits.remove("banana")
print(fruits)

# Example: Reversing List
fruits = ['apple', 'banana', 'cherry']
fruits.reverse()
print(fruits)

# Example: Sorting List
cars = ['Ford', 'BMW', 'Volvo']
cars.sort()
print(cars)
```

Output:

['a', 'b', 'c', 1, 2, 3]

['apple', 'banana', 'cherry', 'orange']

[]

['apple', 'banana', 'cherry']

2

['apple', 'banana', 'cherry', 1, 4, 5, 9]

3

['apple', 'orange', 'banana', 'cherry']

banana

['apple', 'cherry']

['cherry', 'banana', 'apple']

['BMW', 'Ford', 'Volvo']

Tuples:

```
#Tuple
# Example: Creating a Tuple
subjects = ("Math", "Physics", "Chemistry", "Biology",
"History")
print(subjects)

# Example: Accessing Tuple Items
print(subjects[1])
print(subjects[-3])
print(subjects[2:4])
print(subjects[:3])

# Example: Checking If Item is Present
if "Biology" in subjects:
    print("Yes, 'Biology' is a subject")

# Example: Converting Tuple to List, Modifying, and Converting
Back
subjects = ("Math", "Physics", "Chemistry", "Biology",
"History")
print(subjects)
subjects_list = list(subjects)
subjects_list[1] = "Geography"
subjects = tuple(subjects_list)
print(subjects)

# Example: Adding Items to Tuple (Convert to List, Append,
Convert Back)
subjects = ("Math", "Physics", "Chemistry", "Biology",
"History")
print(subjects)
subjects_list = list(subjects)
subjects_list.append("English")
subjects = tuple(subjects_list)
print(subjects)

# Example: Removing Item from Tuple (Convert to List, Remove,
Convert Back)
```

```python
subjects = ("Math", "Physics", "Chemistry", "Biology",
"History")
print(subjects)
subjects_list = list(subjects)
subjects_list.remove("Physics")
subjects = tuple(subjects_list)
print(subjects)

# Example: Unpacking Tuples
colors = ("Red", "Green", "Blue")
(first, second, third) = colors
print(first)
print(second)
print(third)

# Example: Unpacking with *
colors = ("Red", "Green", "Blue", "Yellow", "Purple")
(first, *rest, last) = colors
print(first)
print(rest)
print(last)

# Example: Looping Through Tuple
subjects = ("Math", "Physics", "Chemistry", "Biology",
"History")
for subject in subjects:
    print(subject)
print("Second way to access")
for i in range(len(subjects)):
    print(subjects[i])

# Example: While Loop with Tuple
subjects = ("Math", "Physics", "Chemistry", "Biology",
"History")
i = 0
while i < len(subjects):
    print(subjects[i])
    i += 1

# Example: Joining Tuples
tuple1 = ("a", "b", "c")
tuple2 = (1, 2, 3)
tuple3 = tuple1 + tuple2
print(tuple3)
print("Other Way")
fruits = ("Apple", "Banana", "Cherry")
combined_tuple = fruits * 2
print(combined_tuple)

# Example: Tuple Methods
numbers = (2, 7, 8, 3, 2, 5, 7, 4, 6, 8, 5)
```

```
count_of_5 = numbers.count(5)
print(count_of_5)
print("Second Method")
index_of_8 = numbers.index(8)
print(index_of_8)
```

Output:

('Math', 'Physics', 'Chemistry', 'Biology', 'History')

Physics

Chemistry

('Chemistry', 'Biology')

('Math', 'Physics', 'Chemistry')

Yes, 'Biology' is a subject

('Math', 'Physics', 'Chemistry', 'Biology', 'History')

('Math', 'Geography', 'Chemistry', 'Biology', 'History')

('Math', 'Physics', 'Chemistry', 'Biology', 'History')

('Math', 'Physics', 'Chemistry', 'Biology', 'History', 'English')

('Math', 'Physics', 'Chemistry', 'Biology', 'History')

('Math', 'Chemistry', 'Biology', 'History')

Red

Green

Blue

Red

['Green', 'Blue', 'Yellow']

Purple

Math

Physics

Chemistry

Biology

History

Second way to access

Math

Physics

Chemistry

Biology

History

Math

Physics

Chemistry

Biology

History

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

Other Way

('Apple', 'Banana', 'Cherry', 'Apple', 'Banana', 'Cherry')

2

Second Method

2

Process finished with exit code 0

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

Sets and Dictionary

CODE:

```
students = {"Atharva", "shivam", "Aniket", "Kunal", "Vamshi",
"Atharva", True, 1, 0, False} print(students) print(type(students))
print(len(students)) set1 = set(("raspberry", "banana", "Orange",
"Mango", "Apple", "cherry")) print(set1) for x in set1:     print(x)
set2 = {"raspberry", "banana", "Orange", "Mango", "Apple", 10,
30} print("banana" in set2) students = {"Atharva", "shivam",
"Aniket", "Kunal", "Vamshi"} students.add("atharva")
print(students) set1 = {"banana", "Orange", "Mango", "Apple",
10 , True} students.update(set1) print(students) students =
{"Atharva", "shivam", "Aniket", "Kunal", "Vamshi"}
students.remove("Kunal")
print(students) students = {"Atharva", "shivam",
"Aniket", "Kunal", "Vamshi"} students.discard("Atharva")
print(students) students.pop() print(students)
students.clear() print(students) del students students =
{"Atharva", "shivam", "Aniket", "Kunal", "Vamshi"} set1
= {"banana", "Orange", "Mango", "Apple", 10 , True}
```

```
final_set = students.union(set1) print(final_set) set3 =
{"banana", "Orange", "Mango"} set4 = {"Atharva",
"shivam", "Mango"} set3.intersection_update(set4)
print(set3) set3 = {"banana", "Orange", "Mango"} set4 =
{"Atharva", "shivam", "Mango"} inter_set =
set3.intersection(set4) print(inter_set) set3 =
{"banana", "Orange", "Mango"} set4 = {"Atharva",
"shivam", "Mango"}
set3.symmetric_difference_update(set4) print(set3) set3
= {"banana", "Orange", "Mango"} set4 = {"Atharva",
"shivam", "Mango"} sym_set =
set3.symmetric_difference(set4) print(sym_set) fruits =
{"banana", "Orange", "Mango"} x = fruits.copy() print(x)
set3 = {"banana", "Orange", "Mango"} set4 = {"Atharva",
"shivam", "Mango"} z = set3.difference(set4) print(z)
set3 = {"banana", "Orange", "Mango"} set4 = {"Atharva",
"shivam", "Mango"} set3.difference_update(set4)
print(set3) set3 = {"banana", "Orange", "Mango"} set4 =
{"Atharva", "shivam", "apple"} z = set3.isdisjoint(set4)
print(z) set3 = {"banana", "Orange", "Mango", "Apple",
"cherry"} set4 = {"banana", "Orange", "Mango"}
```

```
z = set4.issubset(set3) print(z) set3 = {"banana",

"Orange", "Mango", "Apple", "cherry"} set4 = {"banana",

"Orange", "Mango"} z = set3.issuperset(set4) print(z)
```

Output:

```
{0, True, 'Vamshi', 'shivam', 'Aniket', 'Atharva', 'Kunal'}
<class 'set'>
7
{'banana', 'Apple', 'raspberry', 'Orange', 'Mango',
'cherry'} banana Apple raspberry Orange Mango
cherry
True
{'atharva', 'Vamshi', 'Aniket', 'shivam', 'Atharva', 'Kunal'}
{'atharva', 'Vamshi', True, 'banana', 'Aniket', 'shivam', 'Apple', 10,
'Kunal', 'Orange', 'Mango', 'Atharva'}
{'Vamshi', 'Aniket', 'shivam', 'Atharva'}
{'Vamshi', 'Aniket', 'shivam', 'Kunal'}
{'Aniket', 'shivam', 'Kunal'} set()
{True, 'Vamshi', 'banana', 'Aniket', 'shivam', 'Apple', 10, 'Kunal',
'Orange', 'Mango', 'Atharva'}
{'Mango'}
{'Mango'}
{'banana', 'shivam', 'Orange', 'Atharva'}
{'banana', 'shivam', 'Orange', 'Atharva'}
{'banana', 'Orange', 'Mango'}
{'banana', 'Orange'}
{'banana', 'Orange'}
True
True
True

=== Code Execution Successful ===
```

```
Functions: student = {"name": "Atharva",
        "age": 19,
        "gender": "male", "branch": "IT"}
print(student) print(len(student))
print(type(student)) print(student["name"])
x = student.get("name") print(x) student1 =
dict(name="Aniket", gender="male")
print(student1) x = student.keys() print(x)
student["DOB"] = 2005 print(student)
        x = student.values() print(x)
        student["name"] = "Atharva Pawaskar"
        print(student) x = student.items()
        print(x) student["name"] = "Atharva"
        print(x)   student   =   {"name":
        "Atharva", "age": 19,
        "gender":  "male",  "branch":  "IT"}
        student.update({"name": "Atharva
Pawaskar"}) print(student)
        student.update({"DOB": 2005})
        print(student) student.pop("DOB")
        print(student) student = {"name":
        "Atharva", "age": 19,
        "gender": "male",
```

```
"branch": "IT"}
student.popitem()  print(student)

student  =  {"name":  "Atharva",

"age": 19,

"gender": "male", "branch":

"IT"} print(student) del

student["branch"] print(student)

student.clear() print(student)

del student student = {"name":

"Atharva", "age": 19,

"gender": "male", "branch":

"IT"} for x in student:

print(x) for x in student:

print(student[x]) for x in

student.keys():     print(x)

for x in student.values():

print(x) for x in

student.items():     print(x)

new_dict = student.copy()

print(new_dict) new_dict1 =

dict(student) print(new_dict1)

all_student = {"student1":

{
```

```
        "name": "Atharva",
        "age": 19,
        "gender": "male",
        "branch": "IT"
        },
        "student2":
        {
        "name": "Aniket",
        "age": 18,
        "gender": "male",
        "branch": "IT"
        }
        }
        print(all_student) print(all_student["student1"]["name
"])
        x = ("key1", "key2", "key3") y = 0
        new_dict = dict.frAtharvakeys(x, y)
        print(new_dict) student = {"name":
        "Atharva", "age": 19, "gender":
        "male"} x =
        student.setdefault("branch",
"IT")
      print(x)
```

Output:

```
{'name': 'Atharva', 'age': 19, 'gender': 'male', 'branch': 'IT'}
4
<class 'dict'>
Atharva
Atharva
{'name': 'Aniket', 'gender': 'male'}
dict_keys(['name', 'age', 'gender', 'branch'])
{'name': 'Atharva', 'age': 19, 'gender': 'male', 'branch': 'IT',
'DOB': 2005} dict_values(['Atharva', 19, 'male', 'IT', 2005])
{'name': 'Atharva Pawaskar', 'age': 19, 'gender': 'male', 'branch':
'IT', 'DOB':
2005} dict_items([('name', 'Atharva Pawaskar'), ('age', 19),
('gender', 'male'), ('branch',
'IT'), ('DOB', 2005)]) dict_items([('name', 'Atharva'), ('age',
19), ('gender', 'male'), ('branch', 'IT'), ('DOB', 2005)])
{'name': 'Atharva Pawaskar', 'age': 19, 'gender': 'male', 'branch':
'IT'} {'name': 'Atharva Pawaskar', 'age': 19, 'gender': 'male',
'branch': 'IT', 'DOB': 2005}
{'name': 'Atharva Pawaskar', 'age': 19, 'gender': 'male', 'branch':
'IT'}
{'name': 'Atharva', 'age': 19, 'gender': 'male'}
{'name': 'Atharva', 'age': 19, 'gender': 'male', 'branch': 'IT'}
{'name': 'Atharva', 'age': 19, 'gender': 'male'}
{}
name age gender
branch
Atharva
19 male
IT
name age gender
branch
Atharva
19 male
IT
('name', 'Atharva')
('age', 19)
('gender', 'male')
('branch', 'IT')
{'name': 'Atharva', 'age': 19, 'gender': 'male', 'branch':
'IT'} {'name': 'Atharva', 'age': 19, 'gender': 'male',
'branch': 'IT'}
{'student1': {'name': 'Atharva', 'age': 19, 'gender': 'male',
'branch': 'IT'},
'student2': {'name': 'Aniket', 'age': 18, 'gender': 'male', 'branch':
'IT'}}
Atharva
{'key1': 0, 'key2': 0, 'key3': 0}
IT
```

```
=== Code Execution Successful ===
```

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

DICTIONARY IN PYTHON

```
Example = {
 "brand": "Ford",
 "model": "Mustang",
 "year": 1964
}
print(Example["brand"])
```

```
Ford
```

```
#Duplicate values will overwrite existing values:

Example= {
 "brand": "Ford",
 "model": "Mustang",
 "year": 1964,
 "year": 2020
} print(Example)
print(len(Example)
)
print(type(Example
))
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
3
<class 'dict'>
```

```
Example = dict(name = "Nikhilesh", age = 47, country = "India")
print(Example)
Example = dict(name = "SHLOK", age = 14, country =
"India") print(Example)
```

```
   {'name': 'Nikhilesh', 'age': 47, 'country': 'India'}
   {'name': 'SHLOK', 'age': 14, 'country': 'India'}
```

```
Example = {
 "brand": "Ford",
 "model": "Mustang",
 "year": 1964 } x
= Example["model"]
print(x)
#There is also a method called get() that
will give you the same result: x =
Example.get("model")
print(x)
```

```
#Add a new item to the original dictionary, and see
that the keys list gets updated as well:

car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
} x =
car.keys()
print(x) #before the change
car["color"] = "white"
print(x) #after the change
```

```
car = {"brand": "Ford","model": "Mustang","year":
1964} x = car.values()
print(x) #before the
change car["year"] = 2020
print(x) #after the
change
```

```
#Add a new item to the original dictionary, and
see that the values list gets updated as well:
car = { "brand": "Ford","model":
"Mustang","year": 1964} x = car.values()
print(x) #before the
change car["color"] =
"red" print(x) #after the
change

#The items() method will return each item
in a dictionary, as tuples in a list. x =
Example.items() print(x)

#Make a change in the original dictionary, and
see that the items list gets updated as well:
car = {"brand": "Ford","model":
"Mustang","year": 1964} x = car.items()
print(x) #before the
change car["year"] = 2020
print(x) #after the
change

#Add a new item to the original dictionary, and
see that the items list gets updated as well:
car = {"brand": "Ford","model":
"Mustang","year": 1964} x = car.items()
print(x) #before the
change car["color"] =
"red" print(x) #after the
change


#Check if "model" is present in the dictionary:

Example = {"brand": "Ford","model": "Mustang","year": 1964} if
"model" in Example: print("Yes, 'model' is one of the keys in
the Example dictionary")
```

```
Mustang
Mustang
dict_keys(['brand', 'model', 'year'])
dict_keys(['brand', 'model', 'year'])
dict_keys(['brand', 'model', 'year', 'color'])
dict_values(['Ford', 'Mustang', 1964])
dict_values(['Ford', 'Mustang', 2020])
dict_values(['Ford', 'Mustang', 1964])
dict_values(['Ford', 'Mustang', 1964, 'red'])
```

```
   dict_items([('brand',  'Ford'),  ('model',
   'Mustang'), ('year', 1964)]) dict_items([('brand',
   'Ford'), ('model', 'Mustang'), ('year', 1964)])
   dict_items([('brand',  'Ford'),  ('model',
   'Mustang'), ('year', 2020)]) dict_items([('brand',
   'Ford'), ('model', 'Mustang'), ('year', 1964)])
   dict_items([('brand',  'Ford'),  ('model',
   'Mustang'), ('year', 1964), ('color', 'red')])
   Yes, 'model' is one of the keys in the Example dictionary

#Change Values
#Change the "year" to 2018:
Example = { "brand": "Ford", "model": "Mustang", "year":
1964} print (Example) Example["year"] = 2018
print(Example)

#update Method
#Update the "year" of the car by using the update()
method: Example = {"brand": "Ford","model":
"Mustang","year": 1964} print(Example)
Example.update({"year": 2020})
print(Example)


   {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
   {'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
   {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
   {'brand': 'Ford', 'model': 'Mustang', 'year': 2020}

#Remove Items
#The pop() method removes the item with the specified key name:

Example = {"brand": "Ford","model": "Mustang","year":
1964} Example.pop("model") print(Example)

  {'brand': 'Ford', 'year': 1964}


#The popitem() method removes the last inserted item
Example = {"brand": "Ford","model": "Mustang","year":
1964} Example.popitem() print(Example)

   {'brand': 'Ford', 'model': 'Mustang'}

#The del keyword removes the item with the specified key
name: Example = {"brand": "Ford","model":
"Mustang","year": 1964} del Example["model"]
print(Example)
```

```
   {'brand': 'Ford', 'year': 1964}


#The clear() method empties the dictionary:
Example = {"brand": "Ford","model": "Mustang","year":
1964} Example.clear()
print(Example)
   {}

#Loop through
#Print all key names in the dictionary, one by one:
Example = {"brand": "Ford","model": "Mustang","year":
1964} for x in Example: print(x)

#Print all values in the dictionary, one by one
print("PART TWO")
Example = {"brand": "Ford","model": "Mustang","year":
1964} for x in Example:
 print(Example[x])

#You can also use the values() method to return values of a dictionary:
print("PART THREE")
for x in Example.values():
 print(x)

#You can use the keys() method to return the keys of a dictionary:
print("PART FOUR")
for x in Example.keys():
 print(x)

#Loop through both keys and values, by using the items() method:
print("PART FIVE")
Example = {"brand": "Ford","model": "Mustang","year":
1964} for x, y in Example.items(): print(x, y)


   brand
   model
   year
   PART TWO
   Ford
   Mustang
   1964
   PART THREE
   Ford
   Mustang
   1964 PART
   FOUR brand
```

```
   model year
   PART FIVE
   brand Ford
   model
   Mustang year
   1964
```

Copy a Dictionary

```
Example = {"brand": "Ford","model": "Mustang","year":
1964} print(Example)
mydict = Example.copy()
print(mydict)

#Make a copy of a dictionary with the dict() function:
Example = {"brand": "Ford","model": "Mustang","year":
1964} mydict = dict(Example)
print(mydict)
```

```
   {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
   {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
   {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Nested Dictionaries

```
myfamily = {"child1" : { "name" : "Jay","year" : 2004},"child2" : {"name" :
"Titan","year" : 2007}, "child3" : {"name" : "Leo","year" : 2011
print(myfamily) print(myfamily["child2"]["name"])
```

```
   {'child1': {'name': 'Jay', 'year': 2004}, 'child2': {'name': 'Titan',
   'year': 2007}, 'child3': {'name': 'Leo', 'year': 2011}} Titan
```

Dictionary Methods

```
#Remove all elements from the car list: car =
{"brand": "Ford","model": "Mustang","year": 1964}
print(car) car.clear() print(car) print("SECOND
PART")

#Copy the car dictionary:
car = {"brand": "Ford","model": "Mustang","year": 1964}
x = car.copy()
print(x)

print("THIRD

PART")
```

```
car = {"brand": "Ford","model": "Mustang","year": 1964}
x =
car.get("model")
print(x)
print("PART FIVE")
#Return the
dictionary's key-
value pairs: car =
{"brand":
"Ford","model":
"Mustang","year":
1964}
x = car.items()
print(x)
print("PART SIX")
#When an item in the dictionary changes value, the view object also gets
updated: car = {"brand": "Ford","model": "Mustang","year": 1964}
x = car.items()
car["year"] =
2018 print(x)
print("PART
SEVEN")

car {"brand": "Ford" "model": "Mustang" "year": 1964}
car = { brand : Ford , model : Mustang , year :
1964} x = car.setdefault("color", "white")
print(x)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
{}
SECOND PART
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
THIRD PART
  Mustang PART FIVE
dict_items([('brand', 'Ford'),
('model', 'Mustang'), ('year',
1964)]) PART SIX
dict_items([('brand', 'Ford'),
('model', 'Mustang'), ('year',
2018)]) PART SEVEN white
```

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

Sets and Dictionary

CODE:

```
students = {"Atharva", "shivam", "Aniket", "Kunal", "Vamshi",
"Atharva", True, 1, 0, False} print(students) print(type(students))
print(len(students)) set1 = set(("raspberry", "banana", "Orange",
"Mango", "Apple", "cherry")) print(set1) for x in set1:     print(x)
set2 = {"raspberry", "banana", "Orange", "Mango", "Apple", 10,
30} print("banana" in set2) students = {"Atharva", "shivam",
"Aniket", "Kunal", "Vamshi"} students.add("atharva")
print(students) set1 = {"banana", "Orange", "Mango", "Apple",
10 , True} students.update(set1) print(students) students =
{"Atharva", "shivam", "Aniket", "Kunal", "Vamshi"}
students.remove("Kunal")
print(students) students = {"Atharva", "shivam",
"Aniket", "Kunal", "Vamshi"} students.discard("Atharva")
print(students) students.pop() print(students)
students.clear() print(students) del students students =
{"Atharva", "shivam", "Aniket", "Kunal", "Vamshi"} set1
= {"banana", "Orange", "Mango", "Apple", 10 , True}
```

```
final_set = students.union(set1) print(final_set) set3 =
{"banana", "Orange", "Mango"} set4 = {"Atharva",
"shivam", "Mango"} set3.intersection_update(set4)
print(set3) set3 = {"banana", "Orange", "Mango"} set4 =
{"Atharva", "shivam", "Mango"} inter_set =
set3.intersection(set4) print(inter_set) set3 =
{"banana", "Orange", "Mango"} set4 = {"Atharva",
"shivam", "Mango"}
set3.symmetric_difference_update(set4) print(set3) set3
= {"banana", "Orange", "Mango"} set4 = {"Atharva",
"shivam", "Mango"} sym_set =
set3.symmetric_difference(set4) print(sym_set) fruits =
{"banana", "Orange", "Mango"} x = fruits.copy() print(x)
set3 = {"banana", "Orange", "Mango"} set4 = {"Atharva",
"shivam", "Mango"} z = set3.difference(set4) print(z)
set3 = {"banana", "Orange", "Mango"} set4 = {"Atharva",
"shivam", "Mango"} set3.difference_update(set4)
print(set3) set3 = {"banana", "Orange", "Mango"} set4 =
{"Atharva", "shivam", "apple"} z = set3.isdisjoint(set4)
print(z) set3 = {"banana", "Orange", "Mango", "Apple",
"cherry"} set4 = {"banana", "Orange", "Mango"}
```

```
z = set4.issubset(set3) print(z) set3 = {"banana",

"Orange", "Mango", "Apple", "cherry"} set4 = {"banana",

"Orange", "Mango"} z = set3.issuperset(set4) print(z)
```

Output:

```
{0, True, 'Vamshi', 'shivam', 'Aniket', 'Atharva', 'Kunal'}
<class 'set'>
7
{'banana', 'Apple', 'raspberry', 'Orange', 'Mango',
'cherry'} banana Apple raspberry Orange Mango
cherry
True
{'atharva', 'Vamshi', 'Aniket', 'shivam', 'Atharva', 'Kunal'}
{'atharva', 'Vamshi', True, 'banana', 'Aniket', 'shivam', 'Apple', 10,
'Kunal', 'Orange', 'Mango', 'Atharva'}
{'Vamshi', 'Aniket', 'shivam', 'Atharva'}
{'Vamshi', 'Aniket', 'shivam', 'Kunal'}
{'Aniket', 'shivam', 'Kunal'} set()
{True, 'Vamshi', 'banana', 'Aniket', 'shivam', 'Apple', 10, 'Kunal',
'Orange', 'Mango', 'Atharva'}
{'Mango'}
{'Mango'}
{'banana', 'shivam', 'Orange', 'Atharva'}
{'banana', 'shivam', 'Orange', 'Atharva'}
{'banana', 'Orange', 'Mango'}
{'banana', 'Orange'}
{'banana', 'Orange'}
True
True
True

=== Code Execution Successful ===
```

```
Functions: student = {"name": "Atharva",
        "age": 19,
        "gender": "male", "branch": "IT"}
print(student) print(len(student))
print(type(student)) print(student["name"])
x = student.get("name") print(x) student1 =
dict(name="Aniket", gender="male")
print(student1) x = student.keys() print(x)
student["DOB"] = 2005 print(student)
        x = student.values() print(x)
        student["name"] = "Atharva Pawaskar"
        print(student) x = student.items()
        print(x) student["name"] = "Atharva"
        print(x)   student   =   {"name":
        "Atharva", "age": 19,
        "gender":  "male",  "branch":  "IT"}
        student.update({"name": "Atharva
Pawaskar"}) print(student)
        student.update({"DOB": 2005})
        print(student) student.pop("DOB")
        print(student) student = {"name":
        "Atharva", "age": 19,
        "gender": "male",
```

```
"branch": "IT"}
student.popitem()  print(student)

student  =  {"name":  "Atharva",

"age": 19,

"gender": "male", "branch":

"IT"} print(student) del

student["branch"] print(student)

student.clear() print(student)

del student student = {"name":

"Atharva", "age": 19,

"gender": "male", "branch":

"IT"} for x in student:

print(x) for x in student:

print(student[x]) for x in

student.keys():     print(x)

for x in student.values():

print(x) for x in

student.items():     print(x)

new_dict = student.copy()

print(new_dict) new_dict1 =

dict(student) print(new_dict1)

all_student = {"student1":

{
```

```
        "name": "Atharva",
        "age": 19,
        "gender": "male",
        "branch": "IT"
        },
        "student2":
        {
        "name": "Aniket",
        "age": 18,
        "gender": "male",
        "branch": "IT"
        }
        }
        print(all_student) print(all_student["student1"]["name
"])
        x = ("key1", "key2", "key3") y = 0
        new_dict = dict.frAtharvakeys(x, y)
        print(new_dict) student = {"name":
        "Atharva", "age": 19, "gender":
        "male"} x =
        student.setdefault("branch",
"IT")
      print(x)
```

Output:

```
{'name': 'Atharva', 'age': 19, 'gender': 'male', 'branch': 'IT'}
4
<class 'dict'>
Atharva
Atharva
{'name': 'Aniket', 'gender': 'male'}
dict_keys(['name', 'age', 'gender', 'branch'])
{'name': 'Atharva', 'age': 19, 'gender': 'male', 'branch': 'IT',
'DOB': 2005} dict_values(['Atharva', 19, 'male', 'IT', 2005])
{'name': 'Atharva Pawaskar', 'age': 19, 'gender': 'male', 'branch':
'IT', 'DOB':
2005} dict_items([('name', 'Atharva Pawaskar'), ('age', 19),
('gender', 'male'), ('branch',
'IT'), ('DOB', 2005)]) dict_items([('name', 'Atharva'), ('age',
19), ('gender', 'male'), ('branch', 'IT'), ('DOB', 2005)])
{'name': 'Atharva Pawaskar', 'age': 19, 'gender': 'male', 'branch':
'IT'} {'name': 'Atharva Pawaskar', 'age': 19, 'gender': 'male',
'branch': 'IT', 'DOB': 2005}
{'name': 'Atharva Pawaskar', 'age': 19, 'gender': 'male', 'branch':
'IT'}
{'name': 'Atharva', 'age': 19, 'gender': 'male'}
{'name': 'Atharva', 'age': 19, 'gender': 'male', 'branch': 'IT'}
{'name': 'Atharva', 'age': 19, 'gender': 'male'}
{}
name age gender
branch
Atharva
19 male
IT
name age gender
branch
Atharva
19 male
IT
('name', 'Atharva')
('age', 19)
('gender', 'male')
('branch', 'IT')
{'name': 'Atharva', 'age': 19, 'gender': 'male', 'branch':
'IT'} {'name': 'Atharva', 'age': 19, 'gender': 'male',
'branch': 'IT'}
{'student1': {'name': 'Atharva', 'age': 19, 'gender': 'male',
'branch': 'IT'},
'student2': {'name': 'Aniket', 'age': 18, 'gender': 'male', 'branch':
'IT'}}
Atharva
{'key1': 0, 'key2': 0, 'key3': 0}
IT
```

```
=== Code Execution Successful ===
```

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

Polymorphism

Executable Code:

```
class Specialstring:
def __len__(self):
return 10


# Driver's code if
__name__ == "__main__":
string = Specialstring()
print(len(string))
print("fly with wings")
print("fly with fuel")


class Bird:
    def fly(self):
        print("fly with wings")


class Airplane:
def fly(self):
```

```
        print("fly with fuel")

class Fish:
def swim(self):
print("Dolphins
swim in sea")



for obj in Bird(),
Airplane(), Fish():     if
hasattr(obj, 'fly'):
obj.fly()     else:
print("Cannot fly")

print(10 + 15)
s1 = "Red" s2 =
"Fort" print(s1
+ s2) a = [10,
20, 30] b = [5,
15, -10]
print(a + b)

class BookX:     def
__init__(self, pages):
self.pages = pages
```

```
class BookY:     def
__init__(self, pages):
self.pages = pages


b1 = BookX(30)
b2 = BookY(20) print('Total
Pages=', b1.pages + b2.pages)


class BookX:     def
__init__(self, pages):
self.pages = pages


    def __add__(self, other):
        return self.pages + other.pages


b1 = BookX(10) b2 =
BookX(15) print('Total
Pages=', b1 + b2)


class A:     def
__init__(self, a):
        self.a = a


    def __add__(self, o):
return self.a + o.a
```

```
ob1 = A(1) ob2 =
A(2) ob3 =
A("Hello") ob4 =
A("World")
print(ob1 + ob2)
print(ob3 + ob4)


class complex:     def
__init__(self, a, b):
        self.a = a
self.b = b

    def __add__(self, other):
        return self.a + other.a, self.b + other.b

Ob1 = complex(1, 2)
Ob2 = complex(2, 3)
Ob3 = Ob1 + Ob2
print(Ob3)

class Point:     def
__init__(self, x=0, y=0):
        self.x = x
self.y = y

    def __str__(self):
```

```
        return "({0},{1})".format(self.x, self.y)


    def __lt__(self, other):
        self_mag = (self.x ** 2) + (self.y **
2)         other_mag = (other.x ** 2) +
(other.y ** 2)         return self_mag <
other_mag


p1 = Point(1, 1)
p2 = Point(-2, -
3) p3 = Point(1,
-1) print(p1 <
p2)   print(p2 <
p3) print(p1 <
p3)


class Student():     def
__init__(self, r_no, name, age,
marks):
        self.r_no = r_no
self.name = name
self.age = age
self.marks = marks


    def displayStudent(self):
        print("Roll no:", self.r_no, "Name:", self.name, ", Age:",
self.age, ", Marks:", self.marks)
```

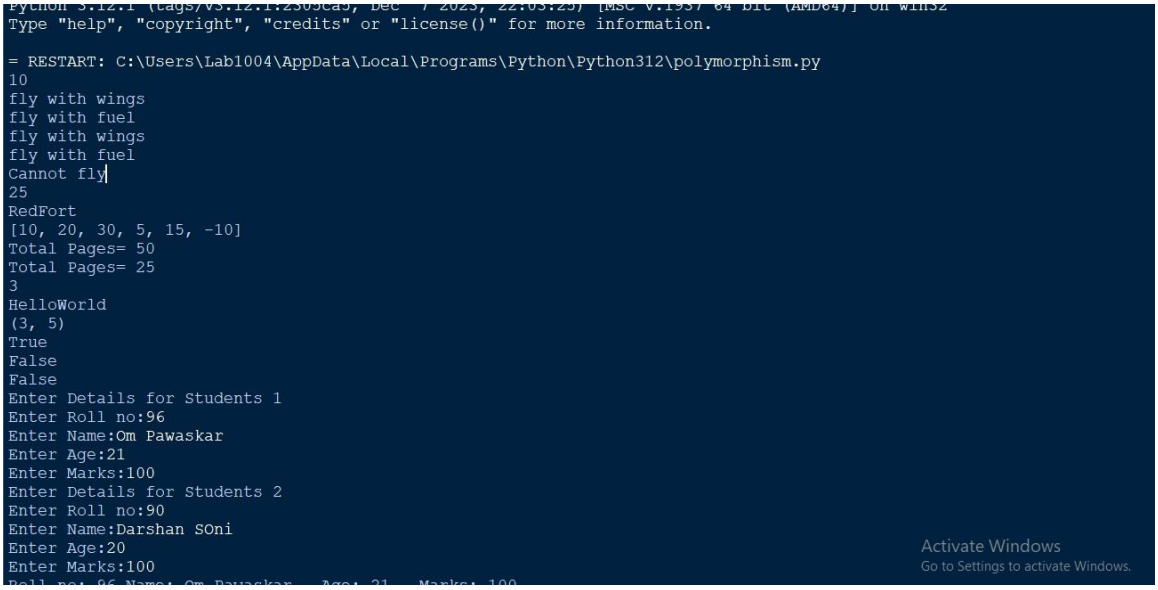
```
    def __str__(self):
        return "({0},{1},{2},{3})".format(self.r_no, self.name,
self.age, self.marks)


    def __eq__(self, other):
        if self.marks == other.marks:
            return self.marks == other.marks
stu = [] for i in
range(1, 3):
    print("Enter Details for Students
%d" % (i))     r_no = int(input("Enter
Roll no:"))     name = input("Enter
Name:")     age = int(input("Enter
Age:"))     marks = input("Enter
Marks:")     stu.append(Student(r_no,
name, age, marks))


for s in stu:
    s.displayStudent()


class Nikhil:     def sum(self, a=None, b=None,
c=None):         if a is not None and b is not
None and c is not None:
            print("Sum of Three=", a +
b + c)         elif a is not None and
```

```
b is not None:             print("Sum
Of two=", a + b)         else:
            print('Please enter two or three Argument')


m = Nikhil()
m.sum(10, 15, 20)
m.sum(10.5, 22.5)
m.sum(10)
class Employee:
def message(self):
        print('This message is from Employee Class')


class Department(Employee):
def message(self):
        print('This Department class is inherited from Employee')


emp = Employee()
emp.message()
print('------------')
dept = Department()
dept.message()


class Employee:
def message(self):
        print('This message is from Employee Class')
```

```
class Department(Employee):

def message(self):

        print('This Department class is inherited from Employee')


class Sales(Department):

def message(self):

        print('This Sales class is inherited from Employee')
emp = Employee()

emp.message()

print('------------')

dept = Department()

dept.message()

print('----------

--') sl = Sales()

sl.message()


class Employee:

def add(self, a, b):

        print('The Sum of Two = ', a + b)


class Department(Employee):

def add(self, a, b, c):

        print('The Sum of Three = ', a + b + c)


emp = Employee()

emp.add(10, 20)

print('------------')
```

```
dept = Department()
dept.add(50, 130, 90)


class Employee:
def message(self):
        print('This message is from Employee Class') class
Department(Employee):    def message(self):        Employee.message(self)
print('This Department class is inherited from Employee')


emp = Employee()
emp.message()
```

Output



```
True
False
False
Enter Details for Students 1
Enter Roll no:96
Enter Name:Om Pawaskar
Enter Age:21
Enter Marks:100
Enter Details for Students 2
Enter Roll no:90
Enter Name:Darshan SOni
Enter Age:20
Enter Marks:100
Roll no: 96 Name: Om Pawaskar , Age: 21 , Marks: 100
Roll no: 90 Name: Darshan SOni , Age: 20 , Marks: 100
Sum of Three= 45
Sum Of two= 33.0
Please enter two or three Argument
This message is from Employee Class
------------
This Department class is inherited from Employee
This message is from Employee Class
------------
This Department class is inherited from Employee
------------
This Sales class is inherited from Employee
The Sum of Two =  30
------------
The Sum of Three =  270
This message is from Employee Class
```

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

INHERITANCE

```python

class SuperClass:
    tabnine: test | explain | document | ask
    def super_method(self):
        print("Super Class method called")

class DerivedClass1(SuperClass):
    tabnine: test | explain | document | ask
    def derived1_method(self):
        print("Derived class 1 method called")

class DerivedClass2(DerivedClass1):
    tabnine: test | explain | document | ask
    def derived2_method(self):
        print("Derived class 2 method called")

d2 = DerivedClass2()
d2.super_method()
d2.derived1_method()
d2.derived2_method()

class shapes:
    tabnine: test | explain | document | ask
    def _init_(self, no_sides):
        self.n = no_sides
        self.sides = [0 for i in range(no_sides)]

    tabnine: test | explain | document | ask
    def takeSides(self):
        self.sides = [float(input("Enter side "+str(i+1)+" : ")) for i in range(self.n)]

    tabnine: test | explain | document | ask
    def disSides(self):
        for i in range(self.n):
            print("Side",i+1,"is",self.sides[i])

class rec(shapes):
    tabnine: test | explain | document | ask
    def _init_(self):
        shapes._init_(self,2)  # Changed number of sides

    tabnine: test | explain | document | ask
    def findArea(self):
        a, b = self.sides
        area = a * b  # Changed area calculation
        print('The area of the rectangle is', area)

t = rec()
t.takeSides()

mytuple = ("apple", "orange", "grapes")  # Changed fruits
myit = iter(mytuple)
print(next(myit))
print(next(myit))
print(next(myit))

mystr = "orange"  # Changed fruit
myit = iter(mystr)
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
```

```python
mytuple = ("apple", "orange", "grapes")  # Changed fruits
for x in mytuple:
    print(x)

mystr = "orange"  # Changed fruit
for x in mystr:
    print(x)

class MyNumbers:
    tabnine: test | explain | document | ask
    def __iter__(self):
        self.a = 2  # Changed starting number
        return self

    tabnine: test | explain | document | ask
    def __next__(self):
        if self.a <= 22:  # Changed upper limit
            x = self.a
            self.a += 2  # Changed increment value
            return x
        else:
            raise StopIteration

myclass = MyNumbers()
myiter = iter(myclass)
for x in myiter:
    print(x)

def myfunc():
    x = 500  # Changed value
    print(x)

myfunc()


def myfunc():
    x = 500  # Changed value
    def myinnerfunc():
        print(x)
    myinnerfunc()

myfunc()

x = 500  # Changed value
def myfunc():
    print(x)

myfunc()
print(x)

x = 500  # Changed value
def myfunc():
    x = 400  # Changed local value
    print(x)

myfunc()
print(x)

tabnine: test | explain | document | ask
def myfunc():
    global x
    x = 500  # Changed value
myfunc()
print(x)
```

```
Aditya Sharma
Welcome Aditya Sharma to the class of 2005
Enter side 1 : [user input]
Enter side 2 : [user input]
Enter side 3 : [user input]
Side 1 is [side 1 value]
Side 2 is [side 2 value]
Side 3 is [side 3 value]
The area of the triangle is [calculated area value]
Super Class method called
Derived class 1 method called
Derived class 2 method called
Enter side 1 : [user input]
Enter side 2 : [user input]
Side 1 is [side 1 value]
Side 2 is [side 2 value]
```

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

Polymorphism

Executable Code:

```
class Specialstring:    def __len__(self):
return 10


# Driver's code if __name__ == "__main__":
string = Specialstring()    print(len(string))
print("fly with wings")    print("fly with fuel")


class Bird:
   def fly(self):
      print("fly with wings")


class Airplane:    def fly(self):
      print("fly with fuel")


class Fish:    def swim(self):
print("Dolphins swim in sea")
```

```
for obj in Bird(), Airplane(), Fish():     if hasattr(obj, 'fly'):
obj.fly()     else:         print("Cannot fly")

print(10 + 15) s1 = "Red" s2 = "Fort"
print(s1 + s2) a = [10, 20, 30] b = [5,
15, -10] print(a + b)

class BookX:     def __init__(self, pages):
self.pages = pages

class BookY:     def __init__(self, pages):
self.pages = pages

b1 = BookX(30)
b2 = BookY(20) print('Total Pages=', b1.pages + b2.pages)

class BookX:     def __init__(self, pages):
self.pages = pages

    def __add__(self, other):
        return self.pages + other.pages

b1 = BookX(10) b2 = BookX(15) print('Total Pages=',
b1 + b2)

class A:     def __init__(self, a):
        self.a = a

    def __add__(self, o):         return self.a + o.a
```

```
ob1 = A(1) ob2 = A(2) ob3 = A("Hello")
ob4 = A("World") print(ob1 + ob2)
print(ob3 + ob4)


class complex:     def __init__(self, a, b):
        self.a = a         self.b = b


    def __add__(self, other):
        return self.a + other.a, self.b + other.b


Ob1 = complex(1, 2)
Ob2 = complex(2, 3) Ob3 = Ob1 + Ob2
print(Ob3)


class Point:     def __init__(self, x=0, y=0):
        self.x = x         self.y = y


    def __str__(self):
        return "({0},{1})".format(self.x, self.y)


    def __lt__(self, other):
        self_mag = (self.x ** 2) + (self.y ** 2)         other_mag = (other.x ** 2) +
(other.y ** 2)         return self_mag < other_mag
```

```
p1 = Point(1, 1) p2 = Point(-2, -3) p3 =

Point(1, -1) print(p1 < p2)   print(p2 < p3)

print(p1 < p3)


class Student():     def __init__(self, r_no, name, age, marks):

       self.r_no = r_no        self.name = name        self.age =

age        self.marks = marks


    def displayStudent(self):

       print("Roll no:", self.r_no, "Name:", self.name, ", Age:", self.age, ", Marks:", self.marks)


    def __str__(self):

       return "({0},{1},{2},{3})".format(self.r_no, self.name, self.age, self.marks)


    def __eq__(self, other):

       if self.marks == other.marks:

          return self.marks == other.marks
stu = [] for i in range(1, 3):

    print("Enter Details for Students %d" % (i))     r_no = int(input("Enter

Roll no:"))     name = input("Enter Name:")     age = int(input("Enter Age:"))

marks = input("Enter Marks:")     stu.append(Student(r_no, name, age,

marks))


for s in stu:

    s.displayStudent()


class Nikhil:     def sum(self, a=None, b=None, c=None):        if a is not None and b is

not None and c is not None:
```

```
        print("Sum of Three=", a + b + c)        elif a is not None and b is

not None:           print("Sum Of two=", a + b)        else:

        print('Please enter two or three Argument')


m = Nikhil()

m.sum(10, 15, 20)

m.sum(10.5, 22.5)

m.sum(10)
class Employee:     def message(self):

      print('This message is from Employee Class')


class Department(Employee):     def message(self):

      print('This Department class is inherited from Employee')


emp = Employee() emp.message()

print('------------')

dept = Department() dept.message()


class Employee:     def message(self):

      print('This message is from Employee Class')


class Department(Employee):     def message(self):

      print('This Department class is inherited from Employee')


class Sales(Department):     def message(self):

      print('This Sales class is inherited from Employee')
emp = Employee() emp.message()

print('------------')

dept = Department() dept.message()
```

```
print('------------') sl = Sales() sl.message()


class Employee:     def add(self, a, b):

        print('The Sum of Two = ', a + b)


class Department(Employee):     def add(self, a, b, c):

        print('The Sum of Three = ', a + b + c)


emp = Employee() emp.add(10, 20)

print('------------')

dept = Department() dept.add(50, 130, 90)


class Employee:     def message(self):

        print('This message is from Employee Class') class Department(Employee):                def message(self):

Employee.message(self)         print('This Department class is inherited from Employee')


emp = Employee() emp.message()
```

Output

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: C:\Users\Lab1004\AppData\Local\Programs\Python\Python312\polymorphism.py
10
fly with wings
fly with fuel
fly with wings
fly with fuel
Cannot fly
25
RedFort
[10, 20, 30, 5, 15, -10]
Total Pages= 50
Total Pages= 25
3
HelloWorld
(3, 5)
True
False
False
Enter Details for Students 1
Enter Roll no:96
Enter Name:Om Pawaskar
Enter Age:21
Enter Marks:100
Enter Details for Students 2
Enter Roll no:90
Enter Name:Darshan SOni
Enter Age:20
Enter Marks:100
Roll no: 96 Name: Om Pawaskar  Age: 21  Marks: 100
```

```
True
False
False
Enter Details for Students 1
Enter Roll no:96
Enter Name:Om Pawaskar
Enter Age:21
Enter Marks:100
Enter Details for Students 2
Enter Roll no:90
Enter Name:Darshan SOni
Enter Age:20
Enter Marks:100
Roll no: 96 Name: Om Pawaskar , Age: 21 , Marks: 100
Roll no: 90 Name: Darshan SOni , Age: 20 , Marks: 100
Sum of Three= 45
Sum Of two= 33.0
Please enter two or three Argument
This message is from Employee Class
------------
This Department class is inherited from Employee
This message is from Employee Class
------------
This Department class is inherited from Employee
------------
This Sales class is inherited from Employee
The Sum of Two =  30
------------
The Sum of Three =  270
This message is from Employee Class
```

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ATHARVA YADAV

ROLL NO : 127

BATCH : S23

Array operations in python

NUMPY

```
#NumPy is used to work with arrays. The array object in NumPy is called ndarray.
#We can create a NumPy ndarray object by using the array() function.Create an Numpy
array import numpy as np a = np.array([1, 2, 3, 4, 5,6,7,8]) print(a)
print(type(a)) print(len(a))
```

```
    [1 2 3 4 5 6 7 8]
    <class 'numpy.ndarray'>
    8
```

```
#o create an ndarray, we can pass a list, tuple or any array-like
object into the array() method, #and it will be converted into an
ndarray:
#Use a tuple to create a NumPy
array: import numpy as np a =
np.array((1, 2, 3, 4, 5)) print(a)
print(type(a))
```

```
    [1 2 3 4 5]
    <class 'numpy.ndarray'>
```

```
#Create a 0-D array with value
42 import numpy as np a =
np.array(42) print(a)
print(type(a))
```

```
    42
    <class 'numpy.ndarray'>
```

```
#Create a 1-D array containing the values
1,2,3,4,5: import numpy as np a = np.array([1,
2, 3, 4, 5,6,7,8,9]) print(a) print(len(a))
print(type(a))
    [1 2 3 4 5 6 7 8 9]
    9
    <class 'numpy.ndarray'>
```

ND Arrays

```
#Create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6:
import numpy as np a = np.array([[1,
2, 3,7], [4, 5, 6,9]]) print(a)
print(len(a))
```

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

```
#Create a 3-D array with two 2-D arrays, both containing two
arrays with the values 1,2,3 and 4,5,6: import numpy as np a =
np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(a) print(len(a))
```

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

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

```
#Check how many dimensions the arrays have:
import numpy as np a =
np.array(42) b = np.array([1, 2,
3, 4, 5]) c = np.array([[1, 2,
3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

```
   0
   1
   2
   3
```

```
#Create an array with 5 dimensions and verify that it has 5 dimensions:
import numpy as np arr = np.array([1,
2, 3, 4], ndmin=5) print(arr)
print('number of dimensions :',
arr.ndim)
```

```
   [[[[[1 2 3 4]]]]]
   number of dimensions :
   5 Accessing Array
   Elements
```

```
#Get the first element from the following
array: import numpy as np a = np.array([1,
2, 3, 4]) print("Length of array")
print(len(a)) print(a[0]) print(a[1])
print(a[2])
```

```
Length of array
4
1
2
3
```

```
#Get third and fourth elements from the following array and add
them. import numpy as np a = np.array([1, 2, 3, 4]) print("Sum
Of Arrays") print(a[0] + a[1]) print(a[0] + a[2]) print(a[0] +
a[3]) print(a[1] + a[2]) print(a[1] + a[3]) print(a[2] + a[3])
```

```
Sum Of Arrays
3
4
5
5
6
7
```

```
import numpy as np
a = np.array([[1,2,3,4,5],
[6,7,8,9,10]]) print(a[0,0]) print('2nd
element on 1st row: ', a[0, 1])
print(a[0,2]) print(a[0,3])
print(a[0,4]) print(a[1,0])
print(a[1,1]) print(a[1,2])
print(a[1,3]) print(a[1,4]) print(a)
```

```
1
2nd element on 1st row: 2
3
4
5
6
7
8
9
10
[[ 1 2 3 4 5]
 [ 6 7 8 9 10]]
```

```
# 3D array import
numpy as np
a = np.array([[[1, 2, 3], [4, 5, 6]], [[7,
8, 9], [10, 11, 12]]]) print(a[0,0,0])
print(a[0,1,0]) print(a[0,1,2])
print(a[1,0,0]) print(a[1,1,1])
```

```
1
4
6
7
11
```

a[0, 1, 2] prints the value 6.

And this is why:

The first number represents the first

dimension, which contains two arrays:

[[1, 2, 3], [4, 5, 6]] and:

[[7, 8, 9], [10, 11, 12]]

Since we selected 0, we are left with the first
array:

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

The second number represents the second dimension,

which also contains two arrays: [1, 2, 3] and:

[4, 5, 6]

Since we selected 1, we are left with the second
array:

[4, 5, 6]

The third number represents the third dimension,

which contains three values: 4

5

6

Since we selected 2, we end up with the third
value:

6

```
#Use negative indexing to access an array from the end.
#Print the last element from the 2nd dim:
import numpy as np
a = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('Last element from 2nd dim: ', a[1, -1])
```

```
   Last element from 2nd dim: 10
```

Slicing arrays

```
#Slice elements from index 1 to index 5 from the following array:
import numpy as np a =
np.array([1, 2, 3, 4, 5, 6, 7])
print(a[1:5])
print(a[0:6])
print(a[1:6])
#slice elements from particular
index print(a[3:]) print(a[4:])
print(a[2:]) print(a[:4])
print(a[:5]) #Negative Indexing
print(a[-3:-1])
#Return every other element from index 1 to index 5:
print(a[1:5:2])
```

```
#Return every other element from the entire array:
print(a[::2])

#2D array a = np.array([[1, 2, 3, 4, 5], [6,
7, 8, 9, 10]])
print(a)
print(a[1, 1:4])
print(a[0:2, 2])

#From both elements, slice index 1 to index 4 (not
included), this will return a 2-D array: print(a[0:2, 1:4])
```

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

Data Types

```
#Create an array with data type
string: import numpy as np arr =
np.array([1, 2, 3, 4], dtype='S')
print(arr) print(arr.dtype)
```

```
[b'1' b'2' b'3' b'4']
|S1
```

```
#Create an array with data type 4 bytes integer:
import numpy as np arr = np.array([1,
2, 3, 4], dtype='i4') print(arr)
print(arr.dtype)
```

```
[1 2 3 4]
int32
```

```
#Change data type from float to integer by using 'i' as parameter
value: import numpy as np arr = np.array([1.1, 2.1, 3.1])
print(arr) newarr = arr.astype('i') print(newarr)
print(newarr.dtype)
```

```
[1.1 2.1 3.1]
[1 2 3]
int32
```

```
#Change data type from float to integer by using int as parameter
value: import numpy as np arr = np.array([1.1, 2.1, 3.1]) newarr =
arr.astype(int) print(newarr) print(newarr.dtype)
```

```
    [1 2 3]
    int64
```

```
#Change data type from integer to
boolean: import numpy as np arr =
np.array([1, 0, 3]) newarr =
arr.astype(bool) print(newarr)
print(newarr.dtype)
```

```
    [ True False True]
    bool
```

```
#Make a copy, change the original array, and display both
arrays: import numpy as np arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42
print(arr)
print(x)
```

```
    [42 2 3 4 5]
    [1 2 3 4 5]
```

```
#Make a view, change the original array, and display both arrays:
import numpy as np arr =
np.array([1, 2, 3, 4, 5])
x = arr.view()
arr[0] = 42
print(arr)
print(x)
```

```
    [42 2 3 4 5]
    [42 2 3 4 5]
```

```
#Make a view, change the view, and display both
arrays: import numpy as np arr = np.array([1, 2, 3,
4, 5])
x = arr.view()
x[0] = 31
print(arr)
print(x)
```

```
    [31 2 3 4 5]
    [31 2 3 4 5]
```

```
#Print the value of the base attribute to check if an array owns it's data or not:
import numpy as np arr =
np.array([1, 2, 3, 4, 5])
print(arr)
x     = arr.copy()
print(x)
```

```
y     = arr.view()
print(y)
print(x.base)
print(y.base)
```

```
   [1 2 3 4 5]
   [1 2 3 4 5]
   [1 2 3 4 5]
   None
   [1 2 3 4 5]
```

```
#NumPy arrays have an attribute called shape that returns a tuple with each index
having the number of corresponding elements. #Print the shape of a 2-D array:
import numpy as np arr = np.array([[1, 2,
3, 4], [5, 6, 7, 8]]) print(arr.shape)
```

```
   (2, 4)
```

```
#Create an array with 5 dimensions using ndmin using a vector with values 1,2,3,4
#and verify that last dimension has value 4:
import numpy as np arr =
np.array([1, 2, 3, 4], ndmin=5)
print(arr)
print('shape of array :', arr.shape)
```

```
   [[[[[1 2 3 4]]]]] shape of
   array : (1, 1, 1, 1, 4)
```

```
Reshaping arrays
```

```
#Convert the following 1-D array with 12 elements into a 2-D array.
#The outermost dimension will have 4 arrays, each with 3 elements:
import numpy as np arr = np.array([1, 2, 3, 4, 5,
6, 7, 8, 9, 10, 11, 12]) newarr = arr.reshape(4, 3)
print(newarr) newarr = arr.reshape(3, 4)
print(newarr) newarr = arr.reshape(6, 2)
print(newarr) newarr = arr.reshape(2, 6)
print(newarr)
```

```
   [[ 1 2 3]
    [ 4 5 6]
    [ 7 8 9]
    [10 11 12]]
   [[ 1 2 3 4]
    [ 5 6 7 8]
    [ 9 10 11 12]]
   [[ 1 2]
    [ 3 4]
    [ 5 6]
    [ 7 8]
    [ 9 10]
    [11 12]]
   [[ 1 2 3 4 5 6]
    [ 7 8 9 10 11 12]]
```

```
#Convert the following 1-D array with 12 elements into a 3-
D array. #The outermost dimension will have 2 arrays that
contains 3 arrays, each with 2 elements: import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12]) newarr = arr.reshape(2, 3, 2) print(newarr)
newarr = arr.reshape(2, 2, 3) print(newarr)
```

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

    [[ 7 8]
    [ 9 10]
    [11 12]]]
    [[[ 1 2
    3]
    [ 4 5 6]]

    [[ 7 8 9]
    [10 11 12]]]
```

```
#Iterate on the elements of the following 1-D
array: import numpy as np arr = np.array([1, 2,
3])
for x in arr:
 print(x)
```

```
   1
   2
   3
```

```
#Iterate on the elements of the following 2-D array:
import numpy as np arr =
np.array([[1, 2, 3], [4, 5, 6]])
for x in arr:
 print(x)
```

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

```
#Iterate on each scalar element of the 2-D array:
import numpy as np arr =
np.array([[1, 2, 3], [4, 5, 6]])
for x in arr:
 for y in x:
  print(y)
```

```
   1
   2
   3
   4
   5
   6
```

```
#Iterate on the elements of the following 3-D
array: import numpy as np arr = np.array([[[1,
```

```
2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
for x in arr: print(x)
```

```
   [[1 2 3]
    [4 5 6]]
   [[ 7 8 9]
    [10 11 12]] #iterate
down to the scalars:
import numpy as np arr = np.array([[[1, 2, 3], [4, 5, 6]],
[[7, 8, 9], [10, 11, 12]]])
for x in arr:
 for y in x:
  for z in y:
   print(z)
   1
   2
   3
   4
   5
   6
   7
   8
   9
   10
   11
   12


#Iterate through the following 3-D array:
import numpy as np arr = np.array([[[1, 2], [3,
4]], [[5, 6], [7, 8]]])
for x in np.nditer(arr):
 print(x)
```

```
   1
   2
   3
   4
   5
   6
   7
   8


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

```
   [1 2 3 4 5 6]


#join two 2-D arrays along rows
(axis=1): import numpy as np arr1 =
np.array([[1, 2], [3, 4]]) arr2 =
np.array([[5, 6], [7, 8]]) arr =
```

```
np.concatenate((arr1, arr2), axis=1)
print(arr)
```

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

```
#We pass a sequence of arrays that we want to join to the
stack() method along with the axis. #If axis is not
explicitly passed it is taken as 0. import numpy as np arr1
= np.array([1, 2, 3]) arr2 = np.array([4, 5, 6]) arr =
np.stack((arr1, arr2), axis=1) print(arr)
```

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

```
#Stacking along rows import
numpy as np arr1 =
np.array([1, 2, 3]) arr2 =
np.array([4, 5, 6]) arr =
np.hstack((arr1, arr2))
print(arr)
```

```
   [1 2 3 4 5 6]
#Stacking along Columns
import numpy as np arr1 =
np.array([1, 2, 3]) arr2 =
np.array([4, 5, 6]) arr =
np.vstack((arr1, arr2))
print(arr)
```

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

```
#Stacking along
Height(Depth) import numpy
as np arr1 = np.array([1,
2, 3]) arr2 = np.array([4,
5, 6]) arr =
np.dstack((arr1, arr2))
print(arr)
```

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

```
#SPLITING ARRAYS #Split the
array in 3 parts: import numpy
as np arr = np.array([1, 2, 3,
4, 5, 6]) newarr =
np.array_split(arr, 3)
print(newarr)
```

```
#Split the array in 4 parts:
import numpy as np arr =
np.array([1, 2, 3, 4, 5, 6])
```

```
newarr = np.array_split(arr, 4)
print(newarr) #Access the
splitted arrays:

import numpy as np arr =
np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 3)
print(newarr[0])
print(newarr[1])
print(newarr[2])

#Split the 2-D array into three 2-D arrays. import numpy as np
arr = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11,
12]]) newarr = np.array_split(arr, 3) print(newarr)

#Split the 2-D array into three 2-D arrays.
import numpy as np arr = np.array([[1, 2, 3], [4, 5, 6],
[7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]])
newarr = np.array_split(arr, 3) print(newarr)

#Split the 2-D array into three 2-D arrays along rows.
import numpy as np arr = np.array([[1, 2, 3], [4, 5, 6],
[7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]])
newarr = np.array_split(arr, 3, axis=1) print(newarr)

#Use the hsplit() method to split the 2-D array into three
2-D arrays along rows. import numpy as np arr =
np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12],
[13, 14, 15], [16, 17, 18]]) newarr = np.hsplit(arr, 3)
print(newarr)


   [array([1, 2]), array([3, 4]), array([5, 6])]
   [array([1, 2]), array([3, 4]), array([5]), array([6])]
   [1 2]
   [3 4]
   [5 6]
   [array([[1, 2],
    [3, 4]]), array([[5, 6],
    [7, 8]]), array([[ 9, 10],
    [11, 12]])]
   [array([[1, 2, 3],
    [4, 5, 6]]), array([[ 7, 8, 9],
    [10, 11, 12]]), array([[13, 14, 15],
    [16, 17, 18]])]
   [array([[ 1],
    [ 4],
    [ 7],
    [10],
    [13],
    [16]]), array([[ 2],
    [ 5],
    [ 8],
    [11],
    [14],
    [17]]), array([[ 3],
```

```
    [ 6],
    [ 9],
    [12],
    [15],
    [18]])]
   [array([[ 1],
    [ 4],
    [ 7],
    [10],
    [13],
    [16]]), array([[ 2],
    [ 5],
    [ 8],
    [11],
    [14],
    [17]]), array([[ 3],
    [ 6],
    [ 9],
    [12],
    [15],
    [18]])]

#Searching Arrays
#Find the indexes where the value is
4: import numpy as np arr =
np.array([1, 2, 3, 4, 5, 4, 4]) x =
np.where(arr == 4) print(x)

#Find the indexes where the values are
even: import numpy as np arr =
np.array([1, 2, 3, 4, 5, 6, 7, 8]) x =
np.where(arr%2 == 0) print(x)

#Find the indexes where the values are
odd: import numpy as np arr =
np.array([1, 2, 3, 4, 5, 6, 7, 8]) x =
np.where(arr%2 == 1) print(x)

#Find the indexes where the value 7 should be
inserted: import numpy as np arr = np.array([6, 7,
8, 9])
x = np.searchsorted(arr, 7)
print(x)

#Find the indexes where the value 7 should be
inserted, starting from the right: import numpy as
np arr = np.array([6, 7, 8, 9])
x= np.searchsorted(arr, 7, side='right')
print(x)

#Find the indexes where the values 2, 4,
and 6 should be inserted: import numpy as
np arr = np.array([1, 3, 5, 7])
x = np.searchsorted(arr, [2, 4, 6])
print(x)
```

```
(array([3, 5, 6]),)
```

```
  (array([1, 3, 5, 7]),)
  (array([0, 2, 4, 6]),)
  1
  2 [1 2
  3]
```

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

File and Exception Handling

CODE:

```
try:
    # Opening a file in default mode (read
text)     with open("demofile.txt") as f:
        print(f.read())

    # Opening a file in read text mode
explicitly     with open("demofile.txt", "rt")
as f:
        print(f.read())

    # Reading only the first 5 characters of
the file     with open("demofile.txt", "r") as
f:
        print(f.read(5))

    # Reading one line of the file
with open("demofile.txt", "r") as f:
        print(f.readline())

    # Reading the first two lines of the
file     with open("demofile.txt", "r") as
f:
```

```
        print(f.readline())
print(f.readline())

    # Looping through the file line by
line     with open("demofile.txt", "r")
as f:         for x in f:
print(x)

    # Writing to an existing file by appending
    with open("demofile2.txt", "a") as f:
        f.write("Now the file has more
content!")

    # Reading the file after appending
with open("demofile2.txt", "r") as f:
        print(f.read())

    # Overwriting the content of a file
with open("demofile3.txt", "w") as f:
        f.write("Woops! I have deleted the
content!")

    # Reading the file after overwriting
with open("demofile3.txt", "r") as f:
        print(f.read())

    # Creating a new file using "x"
mode     with open("myfile.txt", "x")
as f:         pass

    # Creating a new file using "x" mode, again
to
demonstrate error handling
try:
```

```
        with open("myfile.txt", "x") as f:
            pass               except
FileExistsError:           print("The
file already exists.")

    # Deleting a file     import os
file_to_delete = "demofile.txt"     if
os.path.exists(file_to_delete):
os.remove(file_to_delete)
print(f"The file {file_to_delete} has been
deleted.")     else:
        print("The file does not exist.")

    #  Deleting  a  file  with  error
handling             file_to_delete  =
"demofile.txt"     try:
        os.remove(file_to_delete)
print(f"The  file  {file_to_delete}  has  been
deleted.")     except FileNotFoundError:
        print("The file does not exist.")

    # Deleting an entire folder
folder_to_delete = "myfolder"     if
os.path.exists(folder_to_delete):
        os.rmdir(folder_to_delete)
        print(f"The  folder  {folder_to_delete}
has been deleted.")     else:
        print("The folder does not exist.")

except Exception as e:
    print("An error occurred:", e)
```

```
from scipy import constants
print(dir(constants))
```

```
['Avogadro', 'Boltzmann', 'Btu', 'Btu_IT', 'Btu_th', 'ConstantWarning', 'G', 'Julian_year', 'N_A', 'Planck', 'R', 'Rydberg', 'Stefan_Bol
```

Generate a slider using jupyter widgets Close

```
print(constants.yotta)
print(constants.zetta)
print(constants.exa)
print(constants.peta)
print(constants.tera)
print(constants.giga)
print(constants.mega)
print(constants.kilo)
print(constants.hecto)
print(constants.deka)
print(constants.deci)
print(constants.centi)
print(constants.milli)
print(constants.micro)
print(constants.nano)
print(constants.pico)
print(constants.femto)
print(constants.atto)
print(constants.zepto)
```

```
1e+24
1e+21
1e+18
1000000000000000.0
1000000000000.0
1000000000.0
1000000.0
1000.0
100.0
10.0
0.1
0.01
0.001
1e-06
1e-09
1e-12
1e-15
1e-18
1e-21
```

```
print(constants.kibi)
print(constants.mebi)
print(constants.gibi)
print(constants.tebi)
print(constants.pebi)
print(constants.exbi)
print(constants.zebi)
print(constants.yobi)
```

```
1024
1048576
1073741824
1099511627776
1125899906842624
1152921504606846976
1180591620717411303424
1208925819614629174706176
```

```
print(constants.degree)
print(constants.arcmin)
print(constants.arcminute)
print(constants.arcsec)
print(constants.arcsecond)
```

```
0.017453292519943295
0.0002908882086657216
0.0002908882086657216
4.84813681109536e-06
```

```
4.84813681109536e-06
```

```
print(constants.minute)
print(constants.hour)
print(constants.day)
print(constants.week)
print(constants.year)
print(constants.Julian_year)
```

```
60.0
3600.0
86400.0
604800.0
31536000.0
31557600.0
```

```
print(constants.inch)
print(constants.foot)
print(constants.yard)
print(constants.mile)
print(constants.mil)
print(constants.pt)
print(constants.point)
print(constants.survey_foot)
print(constants.survey_mile)
print(constants.nautical_mile)
print(constants.fermi)
print(constants.angstrom)
print(constants.micron)
print(constants.au)
print(constants.astronomical_unit)
print(constants.light_year)
print(constants.parsec)
```

```
0.0254
0.30479999999999996
0.9143999999999999
1609.3439999999998
2.5399999999999997e-05
0.00035277777777777776
0.00035277777777777776
0.3048006096012192
1609.3472186944373
1852.0
1e-15
1e-10
1e-06
149597870700.0
149597870700.0
9460730472580800.0
3.085677581491367e+16
```

```
from scipy import linalg
import numpy as np
a = np.array([[3, 2, 0], [1, -1, 0], [0, 5, 1]])
b = np.array([2, 4, -1])
x = linalg.solve(a, b)
print(x)
```

```
[ 2. -2.  9.]
```

```
from scipy import linalg
import numpy as np
A = np.array([[1,2],[3,4]])
x = linalg.det(A)
print (x)
```

```
-2.0
```

```
from scipy import linalg
import numpy as np
A = np.array([[1,2],[3,4]])
l, v = linalg.eig(A)
print (l)
print (v)
```

```
[-0.37228132+0.j  5.37228132+0.j]
[[-0.82456484 -0.41597356]
 [ 0.56576746 -0.90937671]]
```

```
import numpy as np
array1 = np.array([1, 3, 5])
array2 = np.array([2, 4, 6])
result = np.dot(array1, array2)
print(result)
```

```
44
```

```
array1 = np.array([[1, 3],[5, 7]])
array2 = np.array([[2, 4],[6, 8]])

result = np.inner(array1, array2)
print(result)
```

```
[[14 30]
 [38 86]]
```

```
array1 = np.array([1, 3, 5])
array2 = np.array([2, 4, 6])

result = np.outer(array1, array2)
print(result)
```

```
[[ 2  4  6]
 [ 6 12 18]
 [10 20 30]]
```

```
A = np.array([[2, 4],
[6, 8]])

b = np.array([5, 6])

x = np.linalg.solve(A, b)
print(x)
```

```
[-2.    2.25]
```

```
import numpy as np

array1 = np.array([[6, 3, 5],
[9, 2, 1],
[7, 8, 4]])
result = np.trace(array1)
print(result)
```

```
12
```

Start coding or generate with AI.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

ASSIGNMENT 6. DATA VISUALIZATION (MATPLOTLIB)

AIM : Write Python code to demonstrate the use of Data Visualization (Matplotlib).

CODE :

```
#Draw a line in a diagram from position (0,0) to position
(9,300):
import
matplotlib.pyplot as
pit import numpy as np

xpoints = np.array([0,
9]) ypoints =
np.array([0, 300])

pit.plot(xpoints,
ypoints) pit.show()


#Draw two points in the
diagram. import
matplotlib.pyplot as pit
import numpy as np
```

```
xpoints =
np.array([1,5])
ypoints =
np.array([13,1])

pit.plot(xpoints,
ypoints, 'o')
pit.show()

#Multiple point
xpoints =
np.array([1,2,5,7])
ypoints =
np.array([3,7,1,15])

pit.plot(xpoints,
ypoints) pit.show()

#Default x points ypoints
=
np.array([3,7,5,12,15,21])

pit.plot(ypoints)
pit.show() #Mark
each point with a
circle: import
matplotlib.pyplot
as plt import numpy
as np ypoints =
np.array([13, 7,
11, 1])
plt.plot(ypoints,
marker = '*')
plt.show()

#Dashed Line
import matplotlib.pyplot
as plt import numpy as
np ypoints =
np.array([13, 4, 11, 3])
```

```
plt.plot(ypoints, 'o-
.r') plt.show()


##Encircled Points import
matplotlib.pyplot as plt import
numpy as np ypoints = np.array([13,
2, 11, 7]) plt.plot(ypoints, marker
= 'o', ms = 20, mec = 'r')
plt.show()

import matplotlib.pyplot
as plt import numpy as
np ypoints =
np.array([2, 5, 11, 10])
plt.plot(ypoints, color
= 'b') plt.show()

#Intersecting Lines
import
matplotlib.pyplot as
plt import numpy as np
y1 = np.array([13, 18,
1, 10]) y2 =
np.array([6, 12, 17,
11])
plt.plot(y1)
plt.plot(y2)
plt.show()

#Names Axis 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("Runner Data")
plt.xlabel("Heart
Rate")
plt.ylabel("Meters(m)"
) plt.show()
```

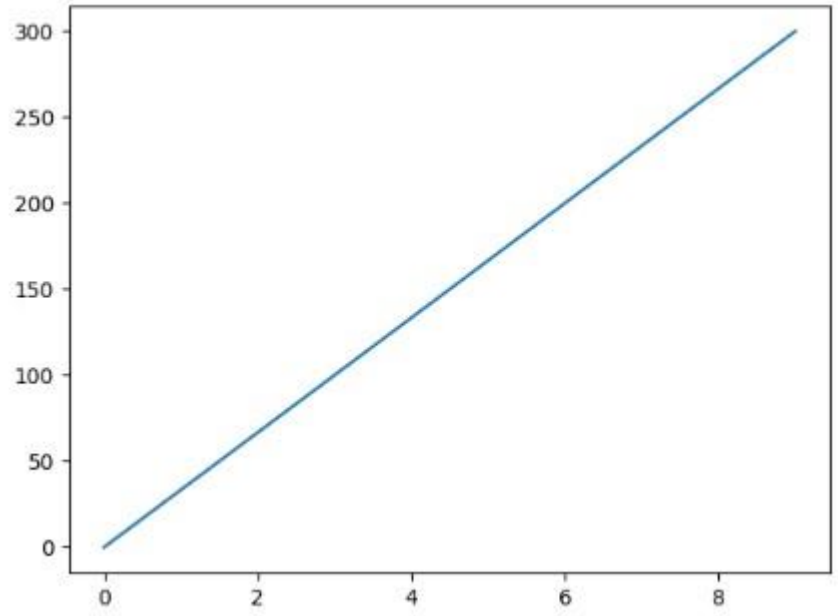
```
#Use of Grids
import numpy as
np
import matplotlib.pyplot as plt x =
np.array([80, 85, 90, 95, 100, 105, 110, 115,
120, 125]) y = np.array([240, 250, 260, 270,
280, 290, 300, 310, 320, 330]) plt.plot(x, y)
plt.title("Runner Data") plt.xlabel("Heart
Rate") plt.ylabel("Meters(m)") plt.grid()
plt.plot(x, y) plt.show()

#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(2, 1,
1) plt.plot(x,y)
#plot 2:
x = np.array([10,
11,3, 3]) y =
np.array([10, 20, 30,
40]) plt.subplot(2,
1, 2) plt.plot(x,y)
plt.show()

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)
```

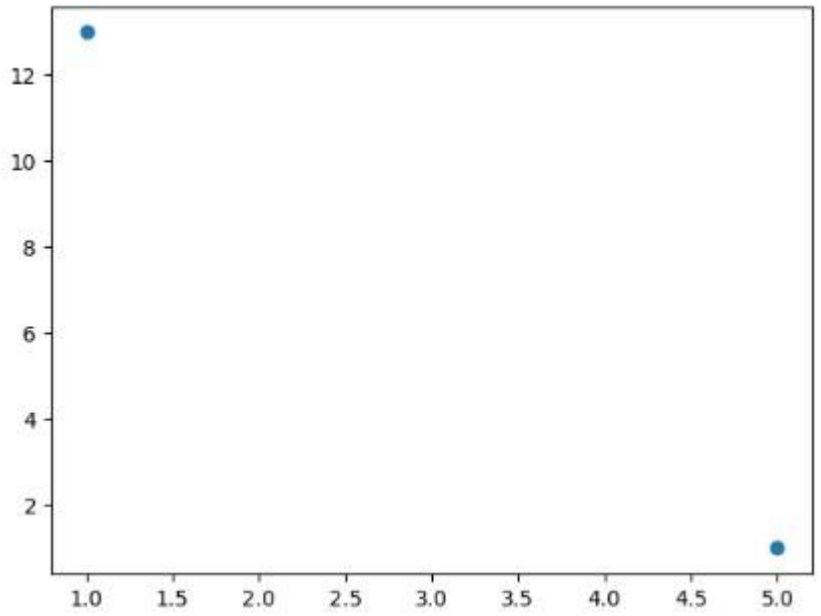
```
plt.title("SALES")
#plot 2:
x = np.array([0, 1,
2, 3]) y =
np.array([10, 20, 30,
40]) plt.subplot(1,
2, 2) plt.plot(x,y)
plt.title("INCOME")
plt.suptitle("MY
SHOP") plt.show()

#Plot of two graphs
import
matplotlib.pyplot as
plt import numpy as np
#day one, the age and speed of 13 cars:
x =
np.array([5,7,8,7,2,17,2,9,4,11,12,9,6]) y
=
np.array([99,86,87,88,111,86,103,87,94,78,7
7,85,86]) plt.scatter(x, y)
#day two, the age and speed of 15 cars:
x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y =
np.array([100,105,84,105,90,99,90,95,94,100,79,112
,91,80,85]) plt.scatter(x, y) plt.show()


#Graphs import sys
import matplotlib
matplotlib.use('Agg')
import
matplotlib.pyplot as
plt import numpy as np
x = ["APPLES",
"BANANAS"] y = [250,
350]
plt.bar(x, y)
plt.show()
```

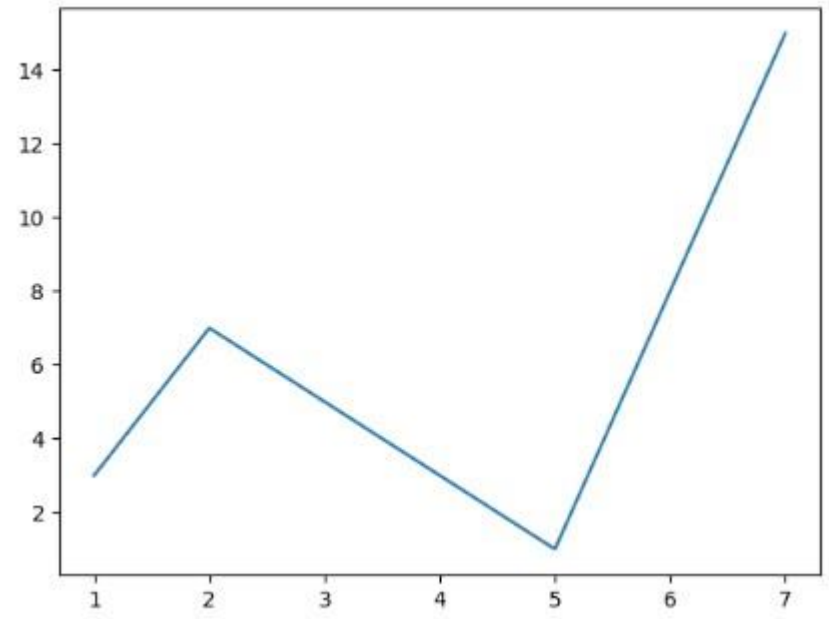
```
import
matplotlib.pyplot as
plt import numpy as np
x = np.array(["A", "B",
"C", "D"]) y =
np.array([8, 18, 9, 19])
plt.barh(x, y, height =
0.2) plt.show()

#Joint Graphs import
matplotlib.pyplot as plt
import numpy as np x =
np.random.normal(190, 90,
250)
plt.hist(x)
plt.show()
#Pie Chart
import
matplotlib.pyp
lot 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()

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

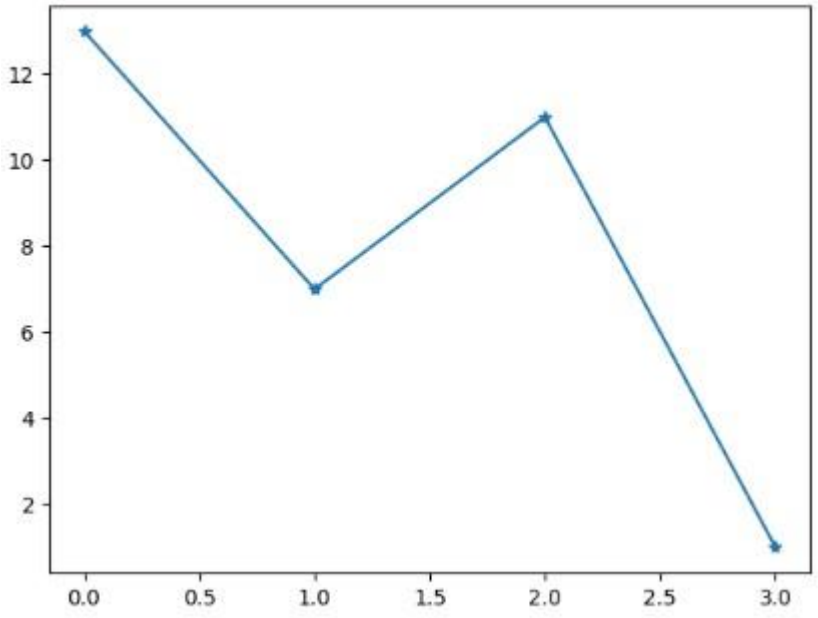
```
#Pie Chart with Tables import
matplotlib.pyplot as plt import numpy as
np y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas",
"Cherries", "Dates"] plt.pie(y, labels =
mylabels) plt.legend(title = "Four
Fruits:") plt.show()
```

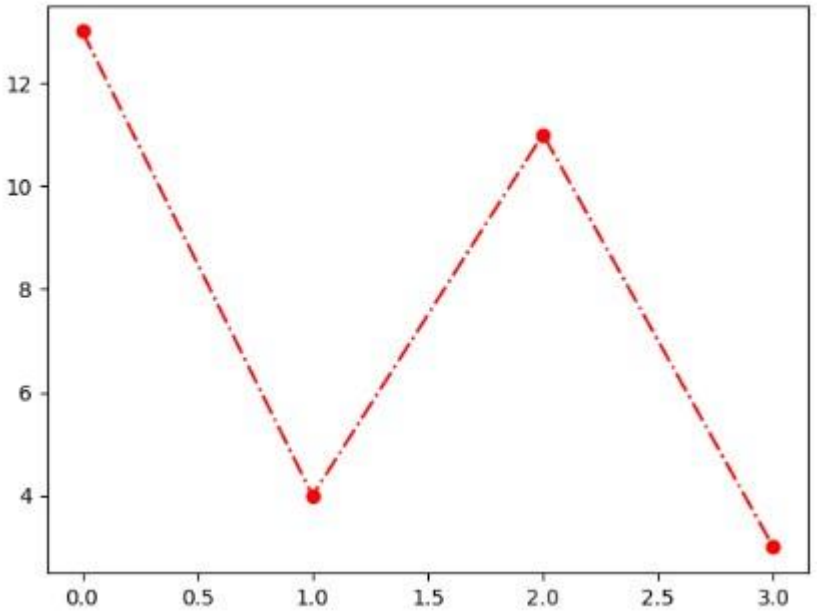
OUTPUT :

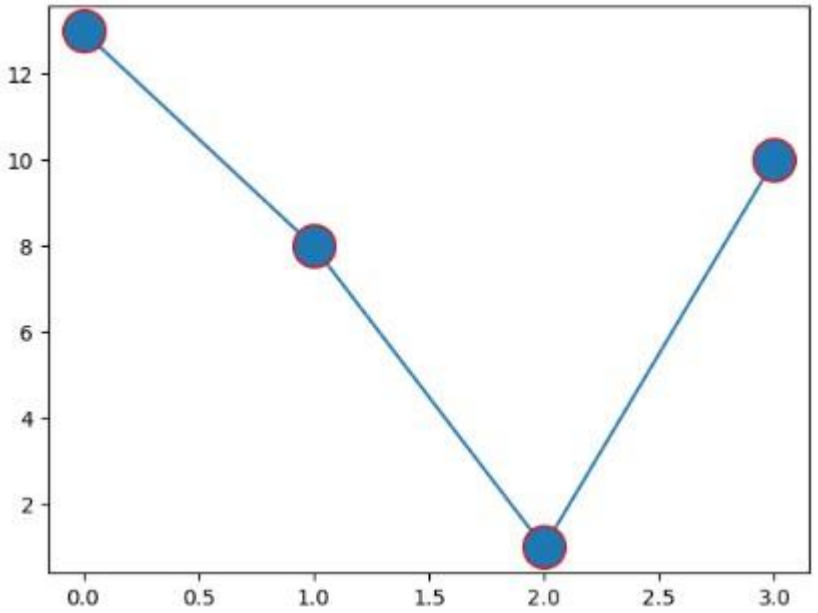


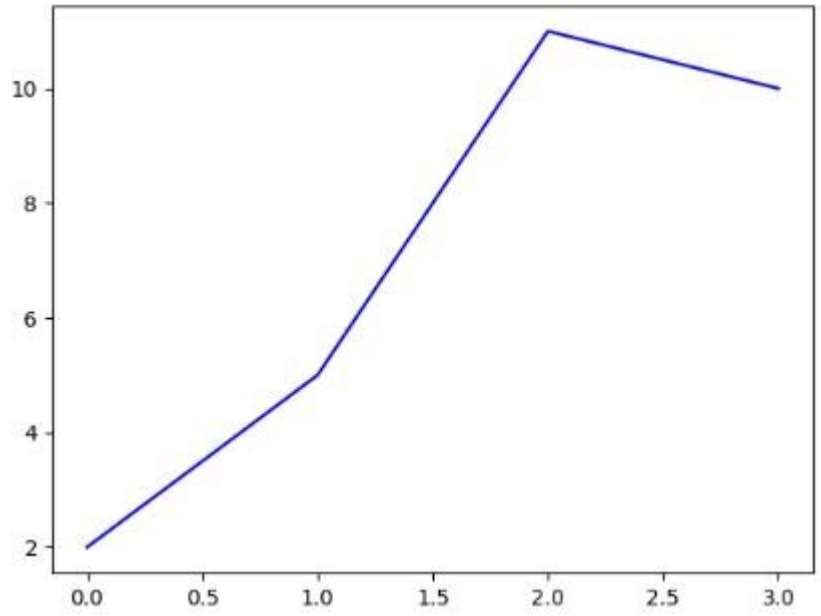

12
10
8
6
4
2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

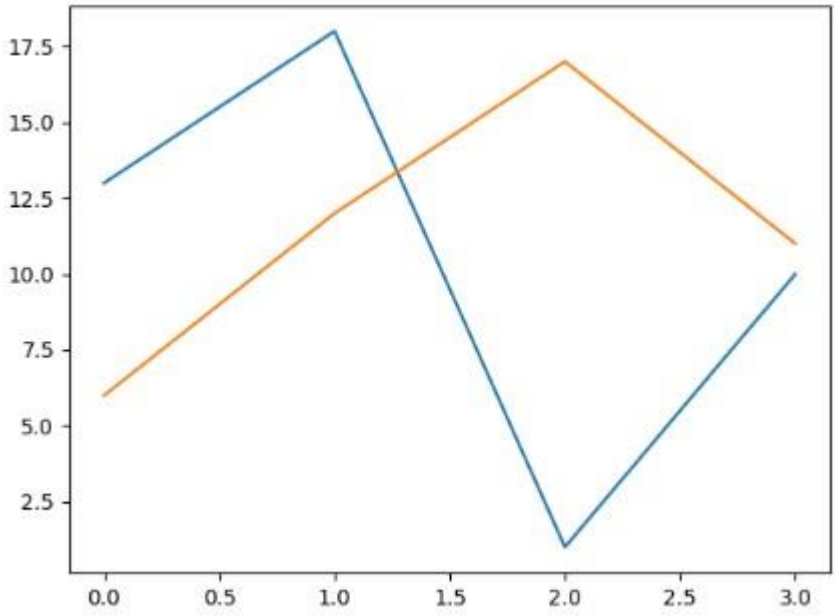

14
12
10
8
6
4
2
1
2
3
4
5
6
7

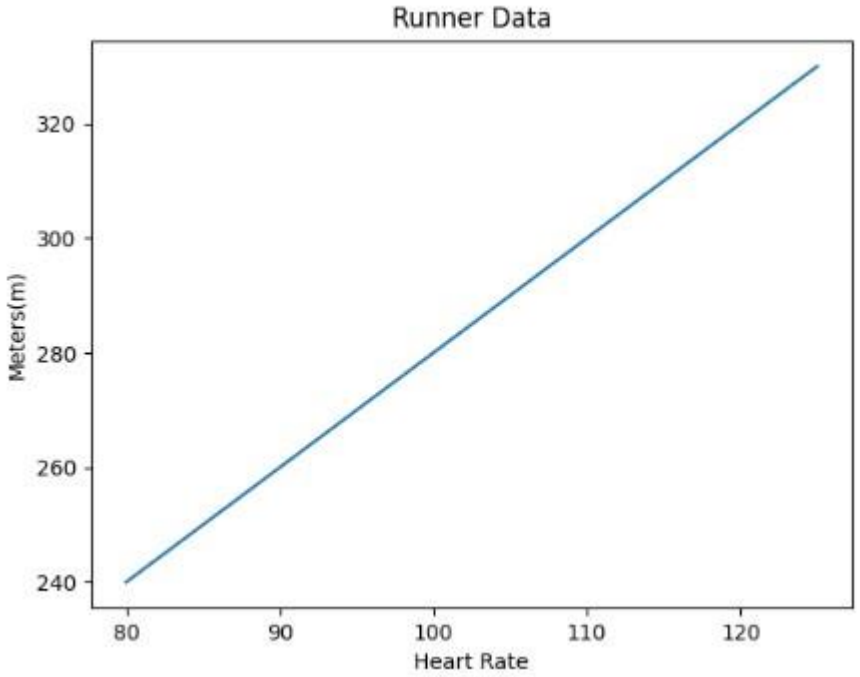

12
10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0

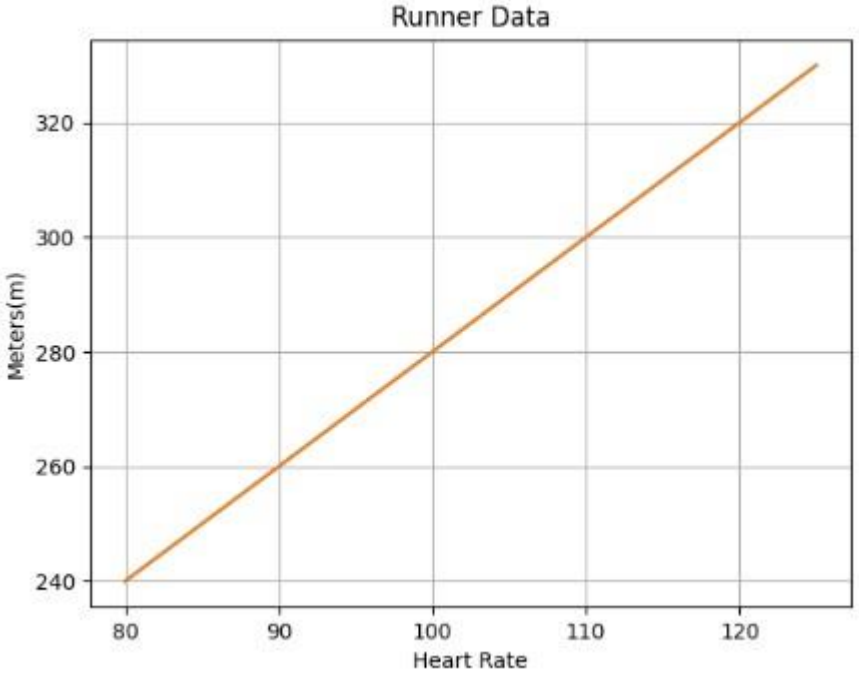

12
10
8
6
4
0.0
0.5
1.0
1.5
2.0
2.5
3.0

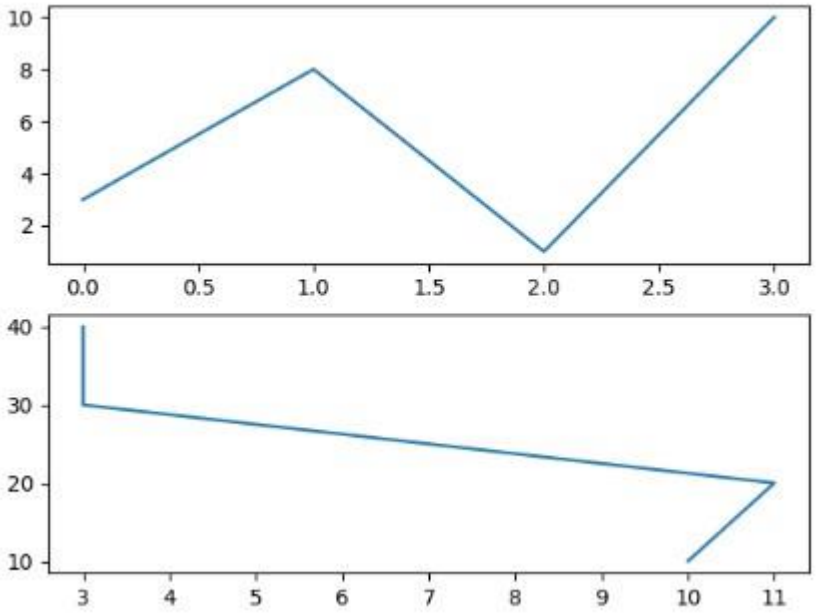

12
10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0


10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0

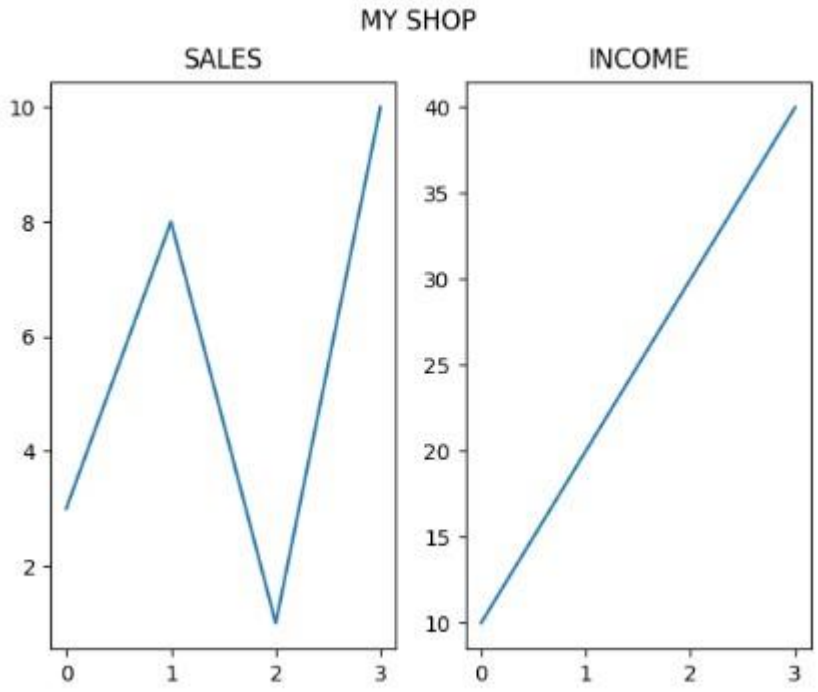

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0

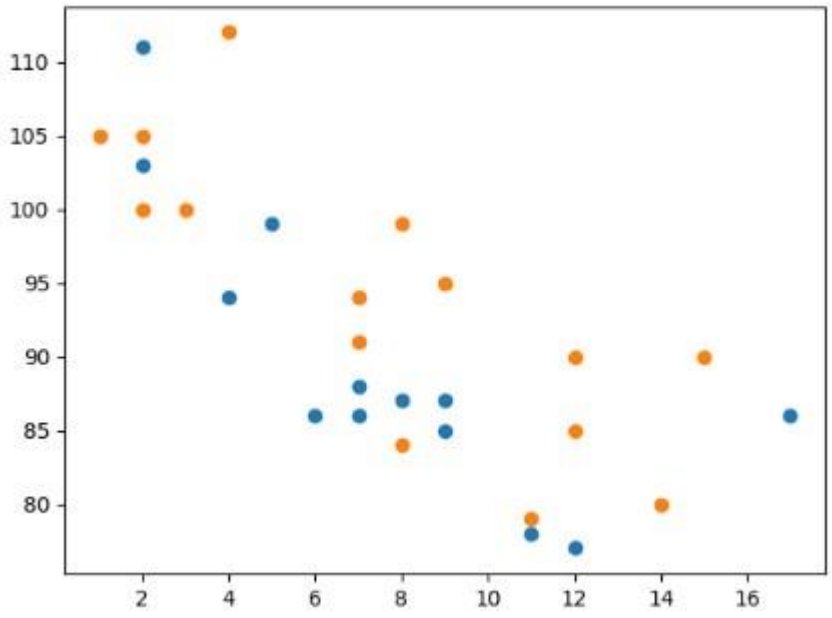

Runner Data
Meters(m)
320
300
280
260
240
80
90
100
110
120
Heart Rate

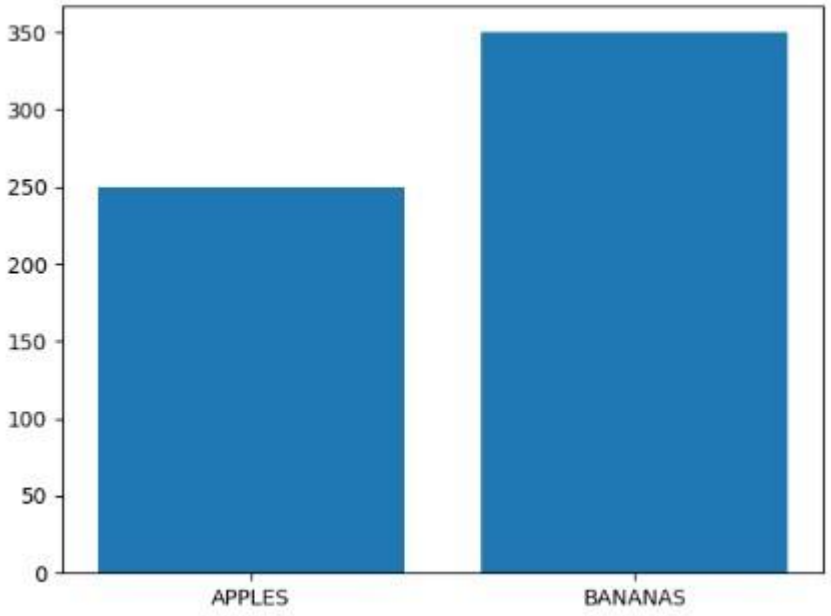

Runner Data
Meters(m)
320
300
280
260
240
80
90
100
110
120
Heart Rate

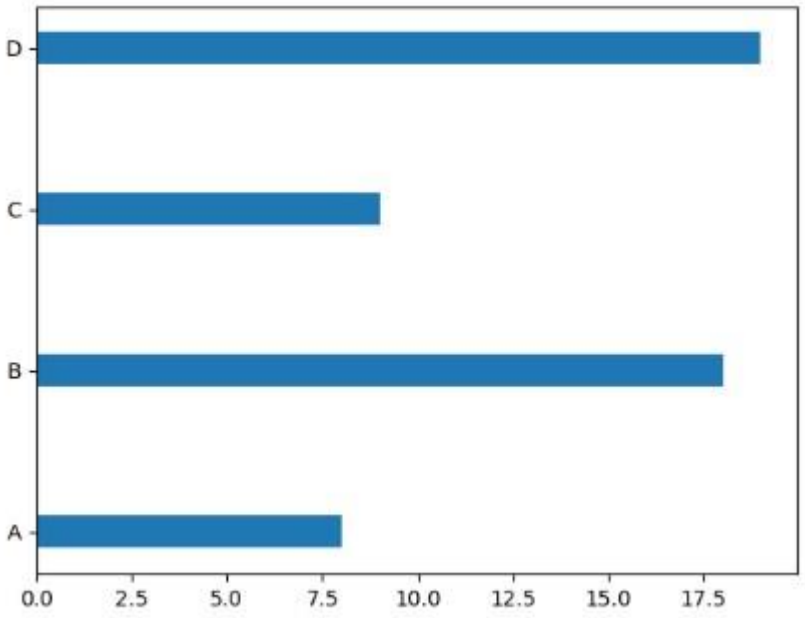

10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0

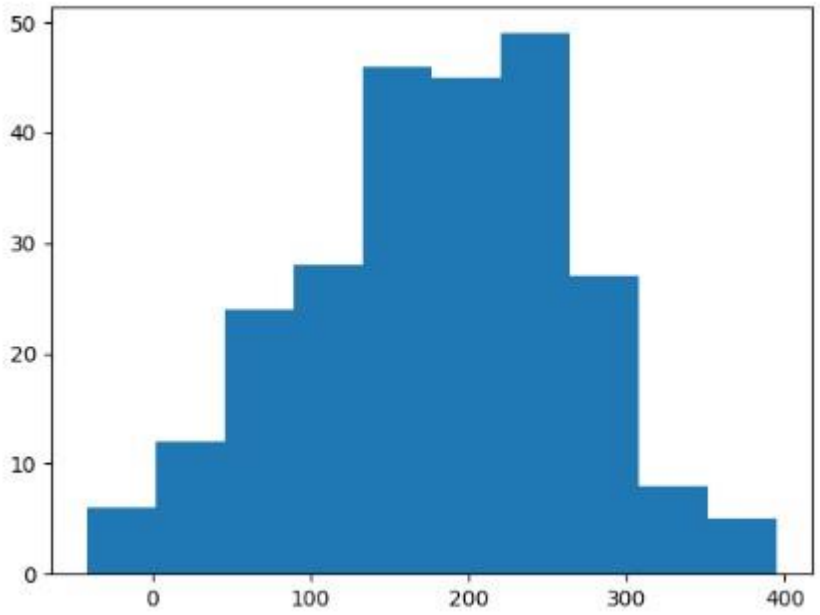
40
30
20
10
3
4
5
6
7
8
9
10
11

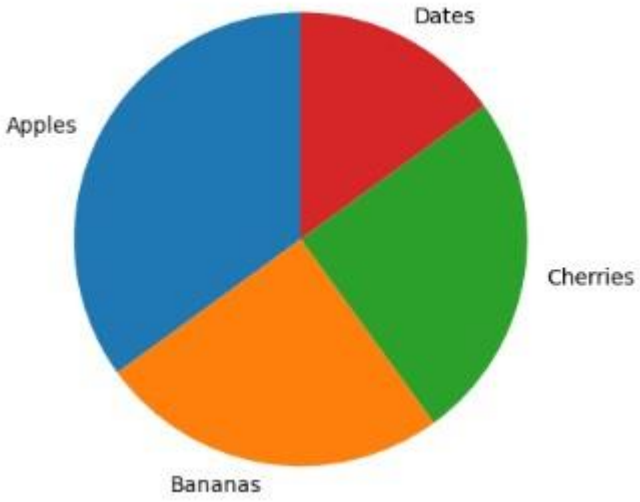

MY SHOP
SALES
INCOME
10
8
6
4
2
40
35
30
25
20
15
10
0
1
2
3

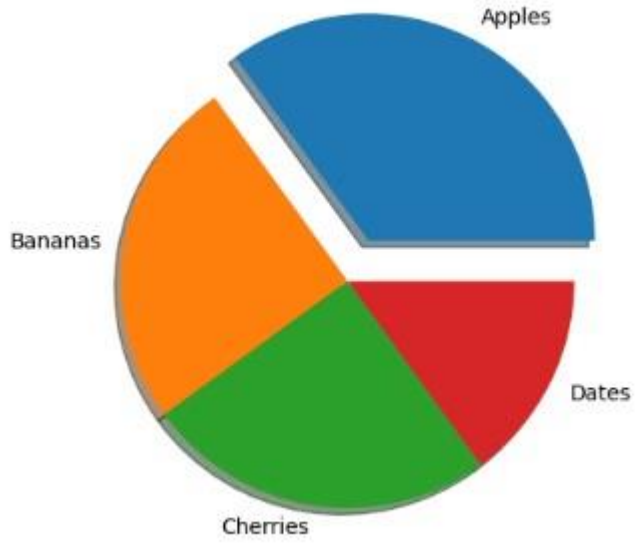

110
105
100
95
90
85
80
2
4
6
8
10
12
14
16

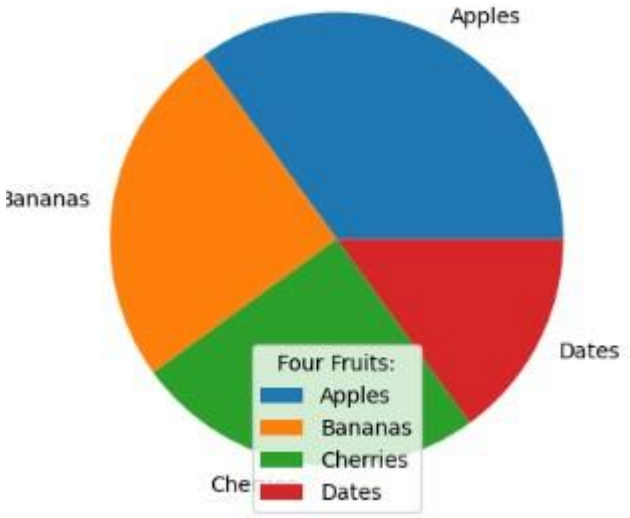

350
300
250
200
150
100
50
0
APPLES
BANANAS


D
C
B
A
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


50
40
30
20
10
0
0
100
200
300
400


Dates
Apples
Cherries
Bananas


Apples
Bananas
Dates
Cherries


Apples
Bananas
Dates
Four Fruits:
Apples
Bananas
Cherries
Dates

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

Python Pandas Assignment

Pandas Basic:

```
import pandas


mydataset ={
    'cars':["Bmw","Ford","Ferrari"],
    "price":[1,2,3]
}


my_var = pandas.DataFrame(mydataset)
print(my_var.loc[[0,1]])


a = [1, 7, 2] myvar = pandas.Series(a, index
= ["x", "y", "z"]) print(myvar)


calories = {"day1": 420, "day2": 380, "day3":
390} myvar = pandas.Series(calories)
print(myvar)


data = {
"calories": [420, 380, 390],
```

```
"duration": [50, 40, 45]
}

myvar = pandas.DataFrame(data) print(myvar)

data = {
"calories": [420, 380, 390],
"duration": [50, 40, 45]
}
df = pandas.DataFrame(data, index = ["day1", "day2", "day3"])
print(df)

#json

data = {
"Duration":{
    "0":60,
    "1":60,
    "2":60,
    "3":45,
    "4":45,
    "5":60
},
"Pulse":{
    "0":110,
    "1":117,
    "2":103,
    "3":109,
    "4":117,
```

```
    "5":102
},
"Maxpulse":{
    "0":130,
    "1":145,
    "2":135,
    "3":175,
    "4":148,
    "5":127
},
"Calories":{
    "0":409,
    "1":479,
    "2":340,
    "3":282,
    "4":406,
    "5":300
}
}


df = pandas.DataFrame(data) print(df)
```

Output:

```
   cars  price
0   Bmw      1 1  Ford
2 x    1 y    7
z    2 dtype: int64 day1
420 day2    380 day3
390
```

```
dtype: int64    calories

duration

0       420         50

1       380         40 2       390

        45        calories  duration day1

        420         50 day2       380

        40 day3        390         45

   Duration  Pulse  Maxpulse  Calories

0        60    110       130       409

1        60    117       145       479 2
         60    103       135       340

3        45    109       175       282

4        45    117       148       406 5

         60    102       127       300
```

Pandas CSV Files

Executable Code: import

```
pandas as pd


pd.options.display.max_rows = 9999

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

print(df.head(20))


hamburger_row = df.loc[df["item"] == "Hamburger"]

print(hamburger_row)
```

```
# new_df = df.dropna()

# print(new_df.to_string())
```

Output:

```
PS C:\Users\Lab1004\Desktop\s22_104> py PandaAssignment.py

   restaurant                                                item
calories  ...  vit_c  calcium  salad

0   Mcdonalds                Artisan Grilled Chicken Sandwich
    380
...   20.0     20.0  Other

1   Mcdonalds                  Single Bacon Smokehouse Burger
840  ...   20.0     20.0  Other

2   Mcdonalds                  Double Bacon Smokehouse Burger
1130  ...   20.0     50.0  Other

3   Mcdonalds       Grilled Bacon Smokehouse Chicken Sandwich
    750
...   25.0     20.0  Other

4   Mcdonalds        Crispy Bacon Smokehouse Chicken Sandwich
920  ...   20.0     20.0  Other

5   Mcdonalds                                          Big Mac
540  ...    2.0     15.0  Other

6   Mcdonalds                                     Cheeseburger
300  ...    2.0     10.0  Other

7   Mcdonalds                         Classic Chicken Sandwich
510  ...    4.0      2.0  Other

8   Mcdonalds                               Double Cheeseburger
430  ...    4.0     15.0  Other

9   Mcdonalds             Double Quarter Pounder® with Cheese
770  ...    6.0     20.0  Other
```

```
10  Mcdonalds                                  Filet-O-Fish®
380  ...    0.0     15.0  Other

11  Mcdonalds                      Garlic White Cheddar Burger
620  ...   10.0     35.0  Other

12  Mcdonalds  Grilled Garlic White Cheddar Chicken Sandwich
    530
...   20.0     35.0  Other

13  Mcdonalds   Crispy Garlic White Cheddar Chicken Sandwich
    700
...   15.0     35.0  Other

14  Mcdonalds                                     Hamburger
250  ...    2.0      4.0  Other

15  Mcdonalds                                  Lobster Roll
290  ...    6.0     15.0  Other

16  Mcdonalds                Maple Bacon Dijon 1/4 lb Burger
    640
...   15.0     15.0  Other

17  Mcdonalds     Grilled Maple Bacon Dijon Chicken Sandwich
    580
...   30.0     30.0  Other

18  Mcdonalds      Crispy Maple Bacon Dijon Chicken Sandwich
    740
...   20.0    290.0  Other

19  Mcdonalds                                     McChicken
    350  ...    2.0      4.0  Other


[20 rows x 17 columns]

      restaurant       item  calories  cal_fat  total_fat  ...
protein  vit_a  vit_c  calcium  salad

14     Mcdonalds  Hamburger       250       70          8  ...
13.0    2.0    2.0      4.0  Other
```

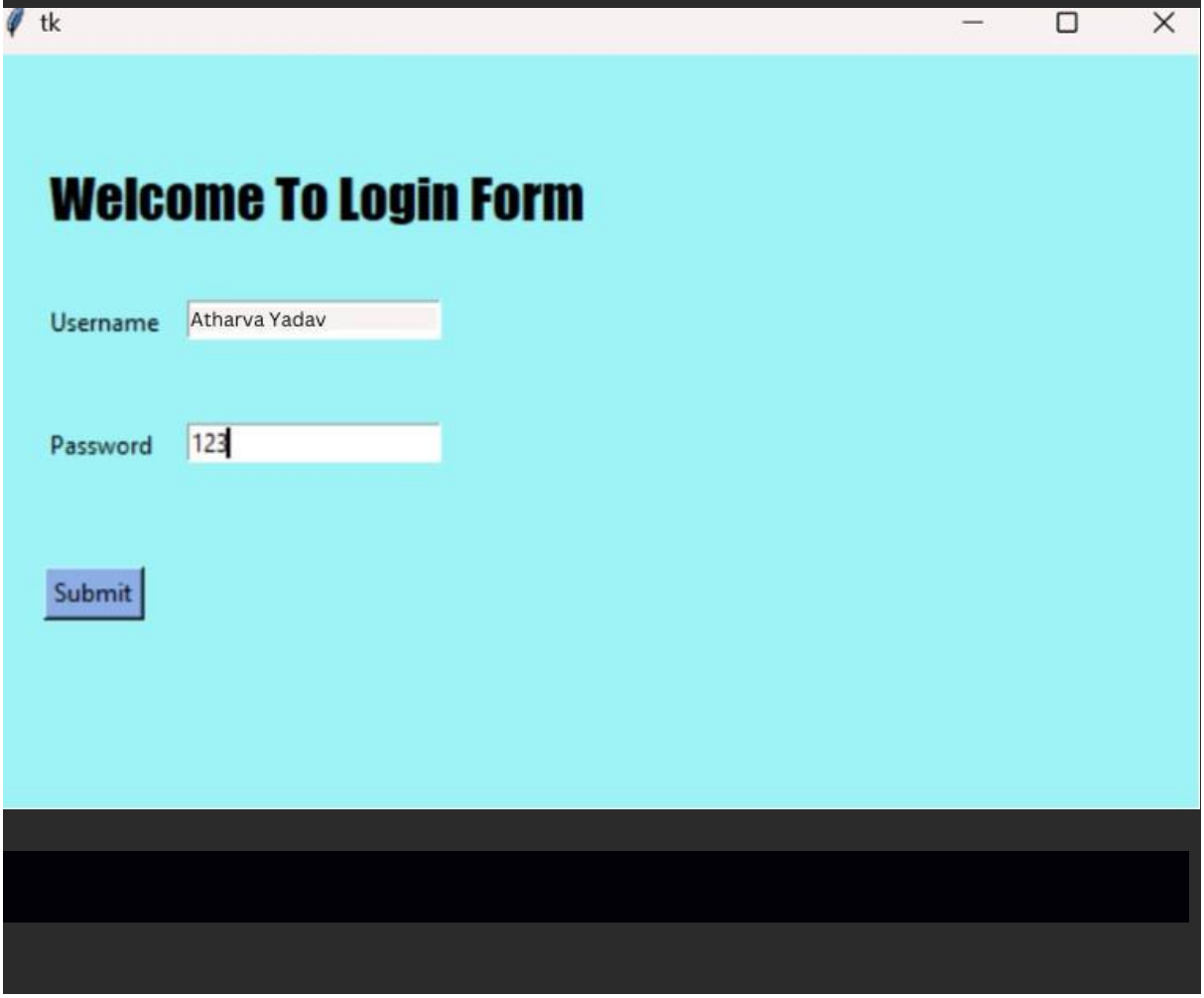
```
206  Burger King  Hamburger       260       90         10  ...
13.0    NaN    NaN      NaN  Other


[2 rows x 17 columns]
```

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

```python
from tkinter import *

window = Tk()
window.geometry("600x400")
window.configure(background="#9ef3f7")

hello_user = Label(window,text="Welcome To Login Form",
font=("Impact",20),background="#9ef3f7")
hello_user.place(x=30, y=50)

exp_username = "Om Pawaskar"
exp_password = "123"
# creating label
username = Label(window, text="Username",background="#9ef3f7")
username.place(x=30, y=120)

password = Label(window, text="Password",background="#9ef3f7")
password.place(x=30, y=180)
entry_box = Entry(window, width=20)
entry_box.place(x=100, y=120)

entry_box2 = Entry(window, width=20)
entry_box2.place(x=100, y=180)

result = Label(window, text="",background="#9ef3f7")
result.place(x=30, y=300)

def submit():
    if entry_box.get() == exp_username and entry_box2.get() == exp_password:
        result.configure(text="Login Success", font=("Impact", 20))
    else:
        result.configure(text="Login Failed, Invalid Credentials",
        font=("Impact", 20))
```

```
submit_button = Button(window, text="Submit", command=submit,
background="#8caee6")

submit_button.place(x=30, y=250)
window.mainloop()
```




```
from tkinter import *
import math
import tkinter.messagebox

root = Tk()
root.title("Scientific Calculator")
root.configure(background='white')
root.resizable(width=False, height=False)
root.geometry("480x568+455+90")
calc = Frame(root)
calc.grid()
```

```
class
Calc:
  def __init__(self):
      self.total = 0
      self.current = ''
      self.input_value = True
      self.check_sum = False
      self.op = ''
      self.result = False

  def numberEnter(self, num):
      self.result = False
      firstnum = txtDisplay.get()
      secondnum = str(num)
      if self.input_value:
          self.current = secondnum
```

```
          self.input_value = False
      else:
          if secondnum == '.':
              if secondnum in firstnum:
                  return
          self.current = firstnum + secondnum
      self.display(self.current)

  def sum_of_total(self):
      self.result = True
      self.current = float(self.current)
      if self.check_sum:
          self.valid_function()
      else:
          self.total = float(txtDisplay.get())

  def display(self, value):
      txtDisplay.delete(0, END)
      txtDisplay.insert(0, value)

                      def
     valid_function(self):
     if self.op == "add":
         self.total += self.current
     if self.op == "sub":
         self.total -= self.current
     if self.op == "multi":
         self.total *= self.current
     if self.op == "divide":
         self.total /= self.current
```

```
        if self.op == "mod":
            self.total %= self.current
        self.input_value = True
        self.check_sum = False
        self.display(self.total)

    def operation(self, op):
        self.current = float(self.current)
        if self.check_sum:
            self.valid_function()
        elif not self.result:
            self.total = self.current
            self.input_value = True
        self.check_sum = True
        self.op = op
        self.result = False

    def Clear_Entry(self):
        self.result = False
        self.current = "0"
        self.display(0)
        self.input_value = True

    def All_Clear_Entry(self):
```

```
        self.Clear_Entry()
        self.total = 0

    def pi(self):
        self.result = False
        self.current = math.pi
        self.display(self.current)

    def tau(self):
        self.result = False
        self.current = math.tau
        self.display(self.current)

    def e(self):
        self.result = False
        self.current = math.e
        self.display(self.current)

    def mathPM(self):
        self.result = False
       self.current = -(float(txtDisplay.get()))
        self.display(self.current)
```

```
            def
    squared(self):
    self.result = False
   self.current = math.sqrt(float(txtDisplay.get()))
    self.display(self.current)

           def
    cos(self):
    self.result = False
   self.current = math.cos(math.radians(float(txtDisplay.get())))
    self.display(self.current)

def cosh(self):
    self.result = False
   self.current = math.cosh(math.radians(float(txtDisplay.get())))
    self.display(self.current)

def tan(self):
    self.result = False
   self.current = math.tan(math.radians(float(txtDisplay.get())))
    self.display(self.current)

def tanh(self):
    self.result = False
   self.current = math.tanh(math.radians(float(txtDisplay.get())))
    self.display(self.current)

def sin(self):
    self.result = False
   self.current = math.sin(math.radians(float(txtDisplay.get())))
    self.display(self.current)
```

```
def sinh(self):
    self.result = False
   self.current = math.sinh(math.radians(float(txtDisplay.get())))
    self.display(self.current)

def log(self):
    self.result = False
    self.current = math.log(float(txtDisplay.get()))
    self.display(self.current)

def exp(self):
    self.result = False
    self.current = math.exp(float(txtDisplay.get()))
```

```
        self.display(self.current)

    def acosh(self):
        self.result = False
        self.current = math.acosh(float(txtDisplay.get()))
        self.display(self.current)

                def
        asinh(self):
        self.result = False
       self.current = math.asinh(float(txtDisplay.get()))
        self.display(self.current)


                def
        expm1(self):
        self.result = False
       self.current = math.expm1(float(txtDisplay.get()))
        self.display(self.current)

                 def
        lgamma(self):
        self.result = False
       self.current = math.lgamma(float(txtDisplay.get()))
        self.display(self.current)

    def degrees(self):
        self.result = False
        self.current = math.degrees(float(txtDisplay.get()))
        self.display(self.current)

    def log2(self):
        self.result = False
        self.current = math.log2(float(txtDisplay.get()))
        self.display(self.current)

    def log10(self):
        self.result = False
        self.current = math.log10(float(txtDisplay.get()))
        self.display(self.current)

    def log1p(self):
        self.result = False
        self.current = math.log1p(float(txtDisplay.get()))
```

```
        self.display(self.current)
```

```
added_value = Calc()

txtDisplay = Entry(calc, font=('Helvetica', 20, 'bold'),
                   bg='black', fg='white',
                   bd=30, width=28, justify=RIGHT)
txtDisplay.grid(row=0, column=0, columnspan=4, pady=1)
txtDisplay.insert(0, "0")

numberpad = "789456123"
i = 0
btn = []
for j in range(2, 5):
   for k in range(3):
       btn.append(Button(calc, width=6, height=2,
                         bg='black', fg='white',
                         font=('Helvetica', 20, 'bold'),
                         bd=4, text=numberpad[i]))
      btn[i].grid(row=j, column=k, pady=1)
     btn[i]["command"] = lambda x=numberpad[i]: added_value.numberEnter(x)
       i += 1


Button(calc,          text=chr(67),
width=6,
     height=2, bg='powder blue',
     font=('Helvetica', 20, 'bold')
     , bd=4, command=added_value.Clear_Entry
     ).grid(row=1, column=0, pady=1)

Button(calc,      text=chr(67)      +
chr(69),
     width=6, height=2,
     bg='powder blue',
      font=('Helvetica', 20, 'bold'),
      bd=4,
      command=added_value.All_Clear_Entry
      ).grid(row=1, column=1, pady=1)

Button(calc, text="\u221A", width=6, height=2,
      bg='powder blue', font=('Helvetica',
                              20, 'bold'),
      bd=4, command=added_value.squared
      ).grid(row=1, column=2, pady=1)

Button(calc, text="+", width=6, height=2,
      bg='powder blue',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=lambda: added_value.operation("add")
```

```
      ).grid(row=1, column=3, pady=1)

Button(calc, text="-", width=6,
      height=2, bg='powder blue',
      font=('Helvetica', 20, 'bold'),
```

```
      bd=4, command=lambda: added_value.operation("sub")
      ).grid(row=2, column=3, pady=1)

Button(calc, text="x", width=6,
      height=2, bg='powder blue',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=lambda: added_value.operation("multi")
      ).grid(row=3, column=3, pady=1)

Button(calc, text="/", width=6,
      height=2, bg='powder blue',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=lambda: added_value.operation("divide")
      ).grid(row=4, column=3, pady=1)

Button(calc, text="0", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=lambda: added_value.numberEnter(0)
      ).grid(row=5, column=0, pady=1)

Button(calc,          text=".",
width=6,
     height=2, bg='powder blue',
     font=('Helvetica', 20, 'bold'),
     bd=4, command=lambda: added_value.numberEnter(".")
     ).grid(row=5, column=1, pady=1)

Button(calc,          text=chr(177),
width=6,
    height=2, bg='powder blue', font=('Helvetica', 20, 'bold'),
     bd=4, command=added_value.mathPM
     ).grid(row=5, column=2, pady=1)

Button(calc,          text="=",
width=6,
      height=2, bg='powder blue',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.sum_of_total
      ).grid(row=5, column=3, pady=1)
```

```
# ROW 1 :
Button(calc, text="pi", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.pi
      ).grid(row=1, column=4, pady=1)

Button(calc, text="Cos", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.cos
      ).grid(row=1, column=5, pady=1)

Button(calc, text="tan", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.tan
      ).grid(row=1, column=6, pady=1)

Button(calc, text="sin", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.sin
      ).grid(row=1, column=7, pady=1)

# ROW 2 :
Button(calc, text="2pi", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.tau
      ).grid(row=2, column=4, pady=1)

Button(calc, text="Cosh", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
     bd=4, command=added_value.cosh
     ).grid(row=2, column=5, pady=1)


Button(calc,          text="tanh",
width=6,
     height=2, bg='black', fg='white',
     font=('Helvetica', 20, 'bold'),
     bd=4, command=added_value.tanh
     ).grid(row=2, column=6, pady=1)
```

```
Button(calc,          text="sinh",
width=6,
     height=2, bg='black', fg='white',
     font=('Helvetica', 20, 'bold'),
     bd=4, command=added_value.sinh
      ).grid(row=2, column=7, pady=1)

# ROW 3 :
Button(calc, text="log", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.log
      ).grid(row=3, column=4, pady=1)

Button(calc, text="exp", width=6, height=2,
      bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.exp
      ).grid(row=3, column=5, pady=1)

Button(calc, text="Mod", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=lambda: added_value.operation("mod")
      ).grid(row=3, column=6, pady=1)
```

```
Button(calc, text="e", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.e
      ).grid(row=3, column=7, pady=1)

# ROW 4 :
Button(calc, text="log10", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.log10
      ).grid(row=4, column=4, pady=1)

Button(calc, text="log1p", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.log1p
      ).grid(row=4, column=5, pady=1)

Button(calc,          text="expm1",
width=6,
```

```
     height=2, bg='black', fg='white',
     font=('Helvetica', 20, 'bold'),
     bd=4, command=added_value.expm1
     ).grid(row=4, column=6, pady=1)


Button(calc,          text="gamma",
width=6,
     height=2, bg='black', fg='white',
     font=('Helvetica', 20, 'bold'),
     bd=4, command=added_value.lgamma
     ).grid(row=4, column=7, pady=1)
# ROW 5 :
Button(calc,          text="log2",
width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.log2
      ).grid(row=5, column=4, pady=1)

Button(calc, text="deg", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.degrees
      ).grid(row=5, column=5, pady=1)

Button(calc, text="acosh", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
      bd=4, command=added_value.acosh
      ).grid(row=5, column=6, pady=1)

Button(calc, text="asinh", width=6,
      height=2, bg='black', fg='white',
      font=('Helvetica', 20, 'bold'),
```

```
      bd=4, command=added_value.asinh
      ).grid(row=5, column=7, pady=1)

lblDisplay = Label(calc, text="Scientific Calculator",
                   font=('Helvetica', 30, 'bold'),
                   bg='black', fg='white', justify=CENTER)

lblDisplay.grid(row=0, column=4, columnspan=4)


def iExit():
   iExit = tkinter.messagebox.askyesno("Scientific Calculator",
```

```
                                        "Do you want to exit ?")
    if iExit > 0:
        root.destroy()
        return


def Scientific():
    root.resizable(width=False, height=False)
   root.geometry("944x568+0+0")



def
Standard():
   root.resizable(width=False, height=False)
    root.geometry("480x568+0+0")


menubar = Menu(calc)

# ManuBar 1 :
filemenu       =       Menu(menubar,
tearoff=0)
menubar.add_cascade(label='File',
menu=filemenu)
filemenu.add_command(label="Standard", command=Standard)
filemenu.add_command(label="Scientific", command=Scientific)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=iExit)

# ManuBar 2 :
editmenu = Menu(menubar, tearoff=0)
menubar.add_cascade(label='Edit', menu=editmenu)
editmenu.add_command(label="Cut")
editmenu.add_command(label="Copy")
editmenu.add_separator()
editmenu.add_command(label="Paste")

root.config(menu=menubar)

root.mainloop()
```


Scientific Calculator
File
Edit
675.0
C
CE
√
+
7
8
9
-
4
5
6
x
1
2
3
/
0
.
±
=

Atharva Yadav S23 - 127

PAGE No.
DATE / /

Python Written Assignment 1.

(1) Key features of python.

1. Free and open source:-
Python language is freely available at the official website and you can download it from ~~the given~~ the website.

(2) Easy to code:
Python is a high level programming language python is very easy to learn the language as compared to other languages as compared to other Lang like C, C++, javascript.

(3) Easy to Read.
As you will see, learning python is quite simple Python's syntax is really straightforward.

(4) object oriented programing.
one of key features of python is OOP's Python supports classes, object, encapsulation

(5) GUI programing support.
GUI can be made using a module such as PyQt, Tk in python.

(6) High level language:
python is a high level ~~for~~ language. we donot need to remember the system architecture, nor do we need to manage the memory.

(7) Large community support.
Python has gained popularity over the years. our question are constantly answered by the enormous stack overflow community.

(8) Easy to debug. Excellent information you mistake tracing. simply by glancing at the code, you can determine what it is designed to perform.

PAGE No.
DATE / /

what is python namespace? Explain local, global and built in namespace on detail.

In python a namespace is a container that holds a set of names and their corresponding objects, preventing naming conflicts. There are three main types of namespaces: local, global and built in.

① Local Namespace:-

i) Scope limited to a specific function or block of code.

ii) creation: created when a fn. is called and destroyed when the fn exists.

iii) Access: contains local variables, parameters and tempory variables used within the fn.

② Global Namespace:-

i) scope Encompasses the entire module or script.

ii) Creation created when the script or module is executed and lasts until the program terminates.

iii) Access: Holds global variables and functions that can be accessed from any

③ Built in Namespaces:

i) Scope inherit to python itself, containing builtin fns and exceptions

ii) Automatically available when python starts.

iii) Provides access to built in fns like print(), len() and built in exceptions like TypeError.

Q3] Differentiate between python arrays and lists.

Ans Lists:

i) Datatypes: Lists can contain elements of different data types.

ii) Functionality: Lists provide a versatile range of built in methods and fns.

iii) Library: Part of the core Python lang.

iv) Flexibility: Lists are dynamic and can be easily resized.

v) Syntax: Defined using square brackets []

```
mylist = [1, 2, 'Hello', 3.14]
```

Arrays:-

(i) Data types: Arrays typically hold elements of the same data type.

ii) Functionality: Numpy library enhances array functionality and provides extensive mathematical operations.

iii) Library: Requires the Numpy library to work with arrays.

iv) Performance: Arrays can offers better performance for mathematical operations.

vi) Syntax: Defined using the array class from the Numpy library

```
import numpy as np
my_array = np.array([1, 2, 3, 4])
```

PAGE No.
DATE / /

03] How does break, continue and pass work?

Ans (1) Break:

Usage: used to exit a loop permaturely based on a certain condition.

Effect: When encounted, the loop (eg for or while) is immediately termined and the program continues with the next statement after the loop.

```
for i in range (5):
    if i == 3:
        break
    print(i)
```

In this example the loop will print values 0, 1, 2 and break when 'i' equals to 3

(2) continue:

Skips the rest of the code inside a loop for the current iteration and proceeds to next iteration.

The loop continues with next iteration ignoring the remaining code for the current iteration.

```
for i in range (5):
    if i == 2:
        continue
    print(i)
```

(3) pass:

Acts as no operation statement.
Used when syntactically some code is req. but no action is desired.

```
for i in range(5):
    if i == 2:
        pass
    else:
        print(i)
```

Q] Describe the comman built in data types in python.

Ans: (i) integers ('int'):
used to represent whole nos.
Examples x = 5

(ii) Floating point nos. ('float'):
used to represent decimal or floating point nos.

(iii) strings ('str')
used to represent seq of characters (text)
Example text = "Hello Python"

(iv) Boolean ('bool'):
Represents either 'True' or 'false'
often used for logical comparisons.

(v) Lists.
ordered mutable seq of elements.

(vi) Tuples: ordered, immutable seq of elements.

Munty
26/3/24

Atharva Yadav S23-127

Written Assignment 2

PAGE No. / DATE / /

Q] Exception handling in Python

Exception handling in python is managed through the use of try, except, else and finally blocks. These blocks enable developers to anticipate and manage errors that may occur during the execution of a program, allowing for more robust and error resistant code.

* try block: code that might cause an exception is placed inside a try block.

except block: If an exception occurs in the try block, the flow of exception moves to an except block where the exception can be handled. Multiple except blocks can cater different types of exception.

else block: If no exceptions occur within the try block, the else block is executed.

finally block: code within the finally block executes regardless of whether an exception occurred, making it ideal for cleaning up resources or executing code that must run no matter what.

Example:

Output: You can't divide by zero!

```
python
try:
    result = 10/0
except ZeroDivisionError:
    print("You can't divide by zero!")
else:
    print("Division successfull")
finally:
    Print("Execution complete")
```

PAGE No.
DATE / /

6] Polymorphism in Python:

Polymorphism in programming refers to the ability of different objects to respond, in their own way, to the same method call. In Python, polymorphism can be demonstrated in several ways:

Method overriding: When a subclass provides a specific implementation for a method that is already defined in its superclass.

Duck typing: Python's approach to polymorphism where the class of an object is less important than the methods/attributes the objects has. If an object can "quack" like a duck, Python allows treating it as a duck.

Example of Method ~~Ov~~ overriding:

```
python.
class Bird:
    def fly(self):
        print("Some birds can fly.")
class Parrot(Bird):
    bird.fly()

test_bird(Bird())
test_bird(Parrot())
```

Output:

Some birds can fly

parrots can fly

PAGE No.
DATE / /

Lambda function in Python:-

Lambda function in Python are small, anonymous functions defined by the keyword lambda Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions are often used when a simple function is passed as an argument to higher order functions:

Example:

```
Python
multiply = lamda x, y: x * y
print(multiply(5,6)) # output: 30
```

Output : 30

Q] Multiple inheritance in python.

Multiple inheritance is supported in python. allowing a class to inherit from more than one parent class. This feature enables the child class to access attributes and methods of all the parent classes.

Example:

```
class Father:
    def gardening (self):
        print("I Enjoy gardening").
class Mother:
    def cooking (self):
        print ("I enjoy cooking")
```

PAGE No.
DATE / /

```
class child(father, mother):
    def sports(self):
        print("I enjoy sports").

c = child()                    Output: I enjoy gardening
c.gardening()                          I Love cooking.
c.cooking()
```

8] Creating Arrays in Python.

Arrays in python can be created using the array module for 1D arrays or the numpy library for 1D, 2D and 3D arrays, offering more functionality and efficiency for large datasets.

1D Array:

```
python
import numpy as np
array_1d = np.array([1, 2, 3])
```

2D Array:

```
Python
array_2d = np.array([[1,2,3], [4, 5, 6]])
```

3D Array:

```
python.
array_3d = np.array([[[1, 2],[3,3]], [[5, 6], [7,8]]])
```

These steps and concepts illustrate basic to advanced functionalities in python, demonstrating its versatility from handling errors gracefully to supporting complex inheritance

patterns and accommodating various data structures.

operating overloading in Python allows you to define how operators behave for user defined objects. ~~This is~~ This is achieved by defining special methods in your class that are called when certain operators are used on instances of the class. Here is an example of operator overloading in Python:

```
class vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def add(self, other):
        return vector(self.x + other.x, self.y +
                                        other.y)

    def __mul__(self, scaler):
        return vector(self.x * scaler, self.y * scaler)

    def __str__(self):
        return f"vector({self.x}, {self.y})"

v1 = vector(1, 2)
v2 = vector(3, 4)
result_addition = v1 + v2
print("Addition:", result_addition)
result_multiplication = v1 * 2
print("Multiplication:", result_multiplication)
```

PAGE No.
DATE / /

Output

Addition : vector (4, 6)

Multiplication : vector (2, 4)

Anusha
26/3/24