IT21-PYTHON PROGRAMMING

Q1) Choose the correct option. (½ mark each) **[10]**

i) Which type of programming does python support?

a) all of the mentioned

ii) Which keyword is used for function in python language?

b)def

iii) Which of the following function is used to export graph generated in matplotlib to-png file format?

a) savefig

iv) Which of the following method used for displaying plot?

b)show

v) In python, a class is_____for a concrete object.

d) a blue print

vi) You can delete properties of objects by using the_____keyword.

a) delete b) dedl

c) del d) drop

vii) The class has a documentation string, which can be accessed via?

a) ClassName. – doc —

viii) Which of the following function can be used to check if a file 'Sample,txt'exists?

a) OS.Path. is file ('sample.txt')

ix) Which of following is NOT a correct was of importing math module?

a) a) amport * from math

x) Which of following functions clears the regular expression cache?

c) re.purge ( )

xi) Command to keep thread in sleep mode?

a) Thread.sleep

xii) Which of the following is not an exception hardling keyword is python?

c) accept

xiii) W.r.t MongoDB, which of the following commands will delete a collectionnamed EMP?

a) db.emp.drop ( )

xiv) Chose the correct syntax of reshape ( ) function is Numpy array

.

c) reshape (array, shape)

xv) If zoo = [‘lion’, ‘tiger’], what will be zoo×2?

a) [‘lion,’ ‘tiger’, ‘lion,’ ‘tiger’]

xvi) What is correct syntax for creating single valued tuple?

c) (a,)

xvii) _____ is a string literal denotated by triple quote for providing the specifications of certain program elements.

b) docstring

xviii) The command that is used to install a third-party module in python is

a) pip

xix) The character_____ and ____matches the start and end of strng,respectively.

d)^ and $

xx) The most important object defined in Numpy is an N-dimensional arraytype called?

c) ndrray

Q2) a) Write a program to check the number entered by user is even or odd. Program should accept integer digits only. **[3]**

Solution: num = int (input (“Enter any number to test whether it is odd or even: “)

```
if (num % 2) == 0:

    print (“The number is even”)
else:

    print (“The provided number is odd”)
print(isnumeric(num))
```

b) Illustrate types of arguments used in python function with example. **[3]**

Solution: Python function is a sequence of statements that execute in a certain order, we associate a name with it. This lets us reuse code.

Type of function:

a)Default argument
b) Keyword Argument
c) Arbitrary Arguments

Default Argument: Python Program arguments can have default values. We assign a default value to an argument using the assignment operator in python(=).

When we call a function without a value for an argument, its default value (as mentioned) is used.

```
def add_numbers( a = 7,  b = 8):
    sum = a + b
    print('Sum:', sum)
# function call with two arguments
add_numbers(2, 3)

#  function call with one argument
add_numbers(a = 2)

# function call with no arguments
add_numbers()
```

Keyword Arguments
The idea is to allow the caller to specify the argument name with values so that the caller does not need to remember the order of parameters.

```
def student(firstname, lastname):

    print(firstname, lastname)

# Keyword arguments

student(firstname=’Data’, lastname='Practice')

student(lastname='Practice', firstname=’Data’)
```

Required Arguments

Required arguments are those supplied to a function during its call in a predetermined positional sequence. The number of arguments required in the method call must be the same as those provided in the function's definition.

```
def function( n1, n2 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)
# Calling function and passing two arguments out of order, we need num1 to be 20 and num2 to be 30
print( "Passing out of order arguments" )
function( 30, 20 )
# Calling function and passing only one argument
print( "Passing only one argument" )
```

```
try:
   function( 30 )
except:
   print( "Function needs two positional arguments" )
```

c)Discuss the polymorphism concept in python create suitable pythonclasses**.**

Ans: Define: Polymorphism defines the ability to take different forms. Polymorphism in Python allows us to define methods in the child class with the same name as defined in their parent class.

A child class inherits all the methods from the parent class. However, in some situations, the method inherited from the parent class doesn't quite fit into the child class. In such cases, you will have to re-implement method in the child class.

The following problem with this approach is that the developer has to remember the name of each function separately. In a much larger program, it is very difficult to memorize the name of the functions for every small operation. Here comes the role of method overloading.

```
class Square:
   side = 5
   def calculate_area_sq(self):
      return self.side * self.side

class Triangle:
   base = 5
   height = 4
   def calculate_area_tri(self):
      return 0.5 * self.base * self.height

sq = Square()
tri = Triangle()
print("Area of square: ", sq.calculate_area_sq())
print("Area of triangle: ", tri.calculate_area_tri())
```

Lets take an example of polymorphism with inheritance . Polymorphism in python defines methods in the child class that have the same name as the methods in the parent class. In inheritance, the child class inherits the methods from the parent class. Also, it is possible to modify a method in a child class that it has inherited from the parent class.

This is mostly used in cases where the method inherited from the parent class doesn't fit the child class. This process of re-implementing a method in the child class is known as Method Overriding. Here is an example that shows polymorphism with inheritance:

```
class Bird:
    def intro(self):
      print("There are different types of birds")
```

```
    def flight(self):
      print("Most of the birds can fly but some cannot")

class parrot(Bird):
    def flight(self):
      print("Parrots can fly")

class penguin(Bird):
    def flight(self):
      print("Penguins do not fly")

obj_bird = Bird()
obj_parr = parrot()
obj_peng = penguin()

obj_bird.intro()
obj_bird.flight()

obj_parr.intro()
obj_parr.flight()

obj_peng.intro()
obj_peng.flight()
```

d) Write a python program for the following. **[3]**

i) Create list of fruits

ii) Add new fruit in list.

iii) sort the list.

iv) delete last fruit name from list.

Ans: Create list :

```
fruits =  ["orange", "apple", "grapes"]
print(fruits)

Add new item to list:
fruits.append("Water melon")
print(fruits)

Sort list:
 fruits.sort()
 print(fruits)

delete last element of  list

fruits.delete(3)
print(fruits)
```

e) Write a python function to check the given number is even or odd.**[3]**

Handle suitable exceptions.
Ans: class InvalidNumException(Exception):

```
    pass

try:
    input_num = int(input("Enter a number: "))
    if input_num%2!=0:
        raise InvalidNumException
    else:
        print("Number is even")

except InvalidNumException:
    print("Exception occurred: number is odd")
```

f) Explain constructor concept in python with example. [4]

ans: In Python, the method the **__init__()** simulates the constructor of the class. This method is called when the class is instantiated. It accepts the **self**-keyword as a first argument which allows accessing the attributes or method of the class.

We can pass any number of arguments at the time of creating the class object, depending upon the **__init__()** definition. It is mostly used to initialize the class attributes. Every class must have a constructor, even if it simply relies on the default constructor.

Syntax for constructor declaration:

```
def __init__(self):
    # body of the constructor
```

Types of constructors :

- **default constructor:** The default constructor is a simple constructor which doesn't accept any arguments. Its definition has only one argument which is a reference to the instance being constructed.

```
class Student:
    def __init__(self):
        print("The First Constructor")
st = Student()
```

- **parameterized constructor:** constructor with parameters is known as parameterized constructor. The parameterized constructor takes its first argument as a reference to the

instance being constructed known as self and the rest of the arguments are provided by the programmer.

```
class Student:
    # Constructor - parameterized
    def __init__(self, name):
        print("This is parametrized constructor")
        self.name = name
    def show(self):
        print("Hello",self.name)
student = Student("John")
student.show()
```

Multiple constructor in python:

```
class Student:
    def __init__(self):
        print("The First Constructor")
    def __init__(self):
        print("The second contructor")

st = Student()
```

Q3) a) Write a python program to create an employee. txt file and store employeename and address.

Answer:

```
with open('F:\Pyhton_2022-23/readme.txt', 'w+') as f:
 f.writelines("Employee Details: \n  Name:   Address \n Jone  Landmark Street \n Smith Bake road")
```

b) Write a python program to create "employee" collection with fields" (ID, name, address, phone email and dept) in mongoDB. Prform the following operations. **[5]**

i) Display all employees in "Accounts" department

ii) Delete employee with ID - 210345

iii) Update phone with new phone for employee ID -123

Ans: Create database Emp_details: use Emp_details

Create collection employee:db.createCollection("employee")

Insert data in collection employee:

```
db.employee.insertMany([
  {
    _id: ObjectId("64b3e8fc23f6a82038c7308a"),
    emp_id: 'ID-123',
    emp_name: 'Joe',
    emp_phonenum: 99887768,
```

```
  emp_depart: 'Finance'
 },
 {
  _id: ObjectId("64b3e9f923f6a82038c7308c"),
  emp_id: 'ID - 210345',
  emp_name: 'John',
  emp_phonenum: 99887769,
  emp_depart: 'IT'
 },
 {
  _id: ObjectId("64b3ea7623f6a82038c7308f"),
  emp_id: 'ID - 210347',
  emp_name: 'Smith',
  emp_phonenum: 99887764,
  emp_depart: 'Account'
 }])
```

Display all employees in “Accounts” department:

```
db.employee.find({"emp_depart":{$in:["Account"]}})
```

Delete employee with ID - 210345

```
db.employee.deleteOne({"emp_id ":”ID - 210347"})
```

Update phone with new phone for employee ID -123

```
db.employee.updateOne({"emp_id":"ID123"},{$set:{"emp_phonenum":99887712}})
```

Or

```
from pymongo import MongoClient # import mongo client to connect
# Creating instance of mongoclient
client = MongoClient()
# Creating database
db = client.Emp_details
employee = employee = ([
  {
    "emp_id": 'ID-123',
    "emp_name": 'Joe',
    "emp_phonenum": 99887768,
    "emp_depart": 'Finance'
  },
  {
    "emp_id": 'ID - 210345',
    "emp_name": 'John',
    "emp_phonenum": 99887769,
    "emp_depart": 'IT'
  },
```

```
  {
   "emp_id": 'ID - 210347',
    "emp_name": 'Smith',
   "emp_phonenum": 99887764,
    "emp_depart": 'Account'
  }])

# Creating document
employees = db.employee

# Inserting data
employees.insert_many(employee)

# Fetching data
for x in employees.find({}, {"emp_id":
1,"emp_name":1,"emp_phonenum":1,"emp_depart":1}):
    print(x)

filter = {"emp_id": 'ID-123' }

newvalues = { "$set": { ' emp_phonenum ': 99887710} }

employees.update_ne(filter, newvalues)

for x in employees.find({}, {"emp_id":
1,"emp_name":1,"emp_phonenum":1,"emp_depart":1}):

    print(x)

filter1={‘ emp_id ":”ID - 210347’}

employee.delete_one(filter1)
```

c)What is tuple?

What is the difference between list and tuple? [**3]**

Ans: A comma-separated group of items is called a Python triple. The ordering, settled items, and reiterations of a tuple are to some degree like those of a rundown, but in contrast to a rundown, a tuple is unchanging.

Features of Python Tuple

- Tuples are an immutable data type, meaning their elements cannot be changed after they are generated.
- Each element in a tuple has a specific order that will never change because tuples are ordered sequences.

```
int_tuple = (4, 6, 8, 10, 12, 14)
print("Tuple with integers: ", int_tuple)
```

Accessing tuple:

```
thistuple=("apple","banana","cherry")
print(thistuple[1])
```

Updating tuple:

```
x=("apple","banana","cherry")
y=list(x)
y[1]="kiwi"
x=tuple(y)
print(x)
```

Sno	LIST	TUPLE
1	Lists are mutable	Tuples are immutable
2	The implication of iterations is Time-consuming	The implication of iterations is comparatively Faster
3	The list is better for performing operations, such as insertion and deletion.	A Tuple data type is appropriate for accessing the elements
4	Lists consume more memory	Tuple consumes less memory as compared to the list
5	Lists have several built-in methods	Tuple does not have many built-in methods.
6	Unexpected changes and errors are more likely to occur	In a tuple, it is hard to take place.

e)What is set? Explain with example. 3marks

Ans: A Python set is the collection of the unordered items.

Each element in the set must be unique, immutable, and the sets remove the duplicate elements. Sets are mutable which means we can modify it after its creation.

Unlike other collections in Python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index. However, we can print them all together, or we can get the list of elements by looping through the set.

Creating a set

The set can be created by enclosing the comma-separated immutable items with the curly braces {}. Python also provides the set() method, which can be used to create the set by the passed sequence.

```
Days = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"}
print(Days)
print(type(Days))
print("looping through the set elements ... ")
for i in Days:
    print(i)
```

Using set() method

```
Days = set(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"])
print(Days)
print(type(Days))
print("looping through the set elements ... ")
for i in Days:
    print(i)
```

Adding items to the set

Python provides the **add()** method and **update()** method which can be used to add some particular item to the set. The add() method is used to add a single element whereas the update() method is used to add multiple elements to the set. Consider the following example.

Use

```
Months = set(["January","February", "March", "April", "May", "June"])
print("\nprinting the original set ... ")
print(months)
print("\nAdding other months to the set...");
Months.add("July");
Months.add ("August");
print("\nPrinting the modified set...");
print(Months)
print("\nlooping through the set elements ... ")
for i in Months:
    print(i)
```

f)Write a program to retrieve and display employee details from "Employee" collection stored in mangoDB database. 3marks

```
# importing module
from pymongo import MongoClient

# creation of MongoClient
client=MongoClient()
```

```
# Connect with the portnumber and host
client = MongoClient(“mongodb://localhost:27017/”)

# Access database
mydatabase = client.Emp_details

# Access collection of the database
mycollection=mydatabase.employee

cursor = mycollection.find()
for record in cursor:
    print(record)
```

g) Write a program to update the employee details stored in “Employee”collection stored in Mangodb database. 4marks

```
# importing module
from pymongo import MongoClient

# creation of MongoClient
client=MongoClient()

# Connect with the portnumber and host
client = MongoClient(“mongodb://localhost:27017/”)

# Access database
mydatabase = client.Emp_details

# Access collection of the database
mycollection=mydatabase.employee

cursor = mycollection.find()
for record in cursor:
    print(record)

filter = {“emp_depart”: 'Account' }
newvalues = { "$set": { ' emp_depart ': “Finance_depart”} }

employee.update_one(filter, newvalues)

cursor = employee.find()
for record in cursor:
    print(record)
```

h) Write python program to read “employee” . txt” file and display alternateemployee record. 2marks

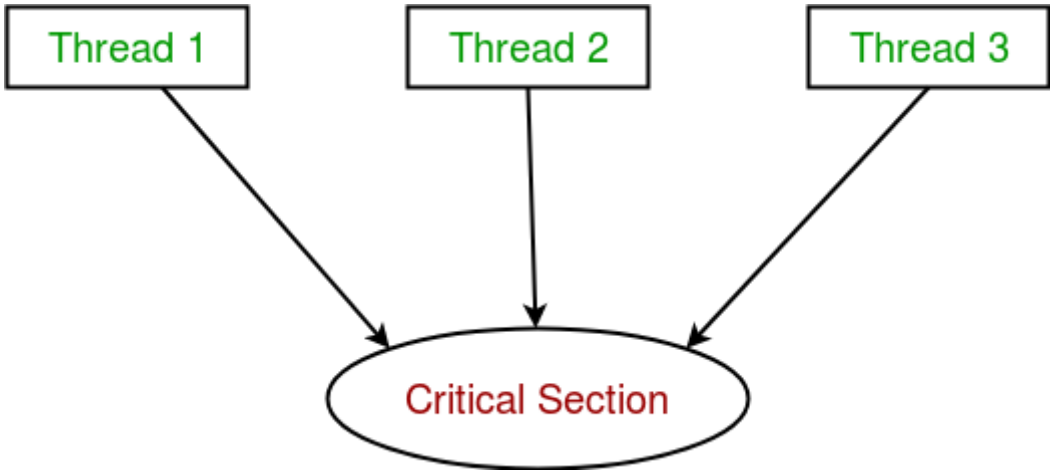
```
Print(“Following operation are to open file in read mode and display the content of the file”)
f = open("employee.txt", "r")
print(f.read())
```

Q4.a) Write a program for extracting email address from a given webpage [4M]

```
# import library
import re

# url link
s = 'https://www.TutorialForPython.com/'

# finding the protocol
obj1 = re.findall('(\w+)://',
                  s)
print(obj1)

# finding the hostname which may
# contain dash or dots
obj2 = re.findall('://www.([\w\-\.]+)',
                  s)
print(obj2)
```

b)What is synchronization in threading? Explain with example. [4m]

Thread synchronization is defined as a mechanism which ensures that two or more concurrent threads do not simultaneously execute some particular program segment known as **critical section**.



Concurrent accesses to shared resource can lead to **race condition**.

A race condition occurs when two or more threads can access shared data and they try to change it at the same time.

```
import threading

# global variable x
x = 0

def increment():
    """
    function to increment global variable x
    """
    global x
    x += 1
```

```
def thread_task():
    """
    task for thread
    calls increment function 100000 times.
    """
    for _ in range(100000):
        increment()

def main_task():
    global x
    # setting global variable x as 0
    x = 0

    # creating threads
    t1 = threading.Thread(target=thread_task)
    t2 = threading.Thread(target=thread_task)

    # start threads
    t1.start()
    t2.start()

    # wait until threads finish their job
    t1.join()
    t2.join()

if __name__ == "__main__":
    for i in range(10):
        main_task()
        print("Iteration {0}: x = {1}".format(i,x))
```

```
Iteration 0: x = 175005

Iteration 1: x = 200000

Iteration 2: x = 200000

Iteration 3: x = 169432

Iteration 4: x = 153316

Iteration 5: x = 200000

Iteration 6: x = 167322

Iteration 7: x = 200000

Iteration 8: x = 169917

Iteration 9: x = 153589
```

In above program:

- Two threads **t1** and **t2** are created in **main_task** function and global variable **x** is set to 0.
- Each thread has a target function **thread_task** in which **increment** function is called 100000 times.

- **increment** function will increment the global variable **x** by 1 in each call.

The expected final value of **x** is 200000 but what we get in 10 iterations of **main_task** function is some different values.
This happens due to concurrent access of threads to the shared variable **x**. This unpredictability in value of **x** is nothing but **race condition**.
threading module provides a **Lock** class to deal with the race conditions.
Lock class provides following methods:

- **acquire([blocking]) :** To acquire a lock. A lock can be blocking or non-blocking.
    - When invoked with the blocking argument set to **True** (the default), thread execution is blocked until the lock is unlocked, then lock is set to locked and return **True**.
    - When invoked with the blocking argument set to **False**, thread execution is not blocked. If lock is unlocked, then set it to locked and return **True** else return **False** immediately.
- **release() :** To release a lock.
    - When the lock is locked, reset it to unlocked, and return. If any other threads are blocked waiting for the lock to become unlocked, allow exactly one of them to proceed.
    - If lock is already unlocked, a **ThreadError** is raised.

```
import threading

# global variable x
x = 0

def increment():
    """
    function to increment global variable x
    """
    global x
    x += 1

def thread_task(lock):
    """
    task for thread
    calls increment function 100000 times.
    """
    for _ in range(100000):
        lock.acquire()
        increment()
        lock.release()

def main_task():
    global x
    # setting global variable x as 0
    x = 0

    # creating a lock
    lock = threading.Lock()

    # creating threads
```

```
    t1 = threading.Thread(target=thread_task, args=(lock,))
    t2 = threading.Thread(target=thread_task, args=(lock,))

    # start threads
    t1.start()
    t2.start()

    # wait until threads finish their job
    t1.join()
    t2.join()

if __name__ == "__main__":
    for i in range(10):
        main_task()
        print("Iteration {0}: x = {1}".format(i,x))
```

c)What is module? Explain with example. 2Marks

A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use.

Create a simple module:

1)A simple module, calc.py

```
def add(x, y):
   return (x+y)

def subtract(x, y):
   return (x-y)
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches for importing a module. For example, to import the module calc.py, we need to put the following command at the top of the script.

2)Import module

```
import calc

print(calc.add(10, 2))

print(calc. subtract (10, 2))
```

c)Write a program to validate URL using regular expression. Explain everyspecial character of the regular expression used in this program. **[4]**

Ans:

```
import re as r
# Making a regular expression to validate an Email
regexp = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'

# Function to validate Email
def check(the_email):
```

```
        if(r.fullmatch(regexp, the_email)):
                print("The entered email is valid")

        else:
                print("The entered email is invalid")

if __name__ == '__main__':

        # Enter the email
        the_email = "rahulpaul123@gmail.com"

        # calling run function
        check(the_email)

        the_email = "my.suraj@our-earth.org"
        check(the_email)
```

1. The uppercase and lowercase letters of English(**A,....,Z**), (**a,.....,z**)
2. Numbers (**0-9**).
3. Characters: **!, #, &, $, %, ', *, +, -, /, =, ?, ^, _, `, {, |, },** ~.
4. Character . (**period, dot, or full stop**) such that it will not come consecutively one after the other.
5. The Company name can exclusively have upper and lowercase letters and numbers.
6. The domain name can exclusively contain upper and lowercase letters.
7. The highest length of the extension can be 3.

e)Write a multithreading program, where one thread prints square of a number and another thread prints factorial of a number. Also display thetotal time taken for the execution. [4Marks]

```
import threading
import time


start = time.time()
def factorial(x):
    """This is a recursive function
    to find the factorial of an integer"""

    if x == 1:
        return 1
    else:
        # recursive call to the function
        return (x * factorial(x-1))

def squarenumber(num):
    # function to print square of given num
    print("Square: {}" .format(num * num))
```

```
if __name__ =="__main__":
    # creating thread
    t1 = threading.Thread(target= factorial, args=(5,))
    t2 = threading.Thread(target= squarenumber, args=(10,))

    # starting thread 1
    t1.start()
    # starting thread 2
    t2.start()

    # wait until thread 1 is completely executed
    t1.join()
    # wait until thread 2 is completely executed
    t2.join()

    # both threads completely executed
    print("Done!")

end = time.time()

print("Execution time of the program is- ", end-start)
```

f) What is anonymous function in python? Demonstrate with example [2Marks]
Answer: An anonymous function in Python is one that has no name when it is defined. In Python, the **lambda** keyword is used to define anonymous functions rather than the def keyword, which is used for normal functions. As a result, lambda functions are another name for anonymous functions.
Syntax
Following is the syntax of the lambda function -

lambda [args] : expression

Example1 : lambda : print('Hello World')

```
Example 2:  triangle = lambda m,n : 1/2 * m * n
res=triangle(34,24)
print("Area of the triangle: ",res)
```

Q5) a)Create 5×5 2D numpy assay and retrieve top left corner 2×2 array from it.[2marks]
Ans:
```
import numpy as np
x = np.full((5,5),7)
print("Original Array:")
print(x)
print(x[0:2,1:3])
```

b)Write a program to illustrate slicing in numpy array. **[2]**

Slicing in python means taking elements from one given index to another given index.

We pass slice instead of index like this: [*start*:*end*].

We can also define the step, like this: [*start*:*end*:*step*].

If we don't pass start its considered 0

If we don't pass end its considered length of array in that dimension

If we don't pass step its considered 1

Example:

In the following example we have consider 1-D array, where we are slicing 1-D array

import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])

In the following example we have consider 2-D array, where we are slicing 2-D array

```
import numpy as np

arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[1, 1:4])
```

where first parameter is which list you want to slice 0= 1st list,1=2ndlist and so on and second parameter is about from which to which element slicing will be done.

c)Create pandas dataframe using two dimensional list. Perform followingoperations. **[4]**

i) Count number of rows.

ii) Count missing values in first column.

iii) Display number of columns in data frame.

Ans:

```
import pandas as pd
    # List1
lst = [['tom', 'reacher', 25], ['krish', 'pete', 30],
     ['nick', 'wilson', 26], ['juli', 'williams', 22]]
```

```
df = pd.DataFrame(lst, columns =['FName', 'LName', 'Age'])
print(df)
```

Count number of rows:

```
print('Row count is:', len(df))
```

Count missing values in first column:

```
df.isnull().sum()
```

Display number of columns in data frame.

```
# Getting the list of columns
col = df.columns

print('Number of columns :', len(col))
```

d) Explain the terms w.r.t. oops in python. **[2]**

i) --lnit-- ii)Self

Ans: __init__: The __init__ method is the Python equivalent of the C++ constructorin an object-oriented approach. The __init__ function is called every time an object is created from a class. The __init__ method lets the class initialize the object's attributes and serves no other purpose. It is only used within classes.

Example: # init method or constructor

def __init__(self, fruit, color):

self.fruit = fruit

self.color = color

Self: The self is used to represent the instance of the class. With this keyword, you can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

Example: # init method or constructor

def __init__(self, fruit, color):

self.fruit = fruit

self.color = color

d)Create 3×3 numpy array and display column wise mean and median.**[2M]**

Ans:

```
import numpy as np
  arr = [[14, 17, 12],
    [15, 6, 27, ],
    [23, 2, 54, ]]
print("\nmedian of arr, axis = None : ", np.median(arr))
```

```
print("\nmean of arr",np.mean(arr))
```

f) Create a series from numpy array and find max and mean of uniqueitems of series. **[2M]**

```
import pandas as pd

# Creating the Series
sr = pd.Series([10, 25, 3, 25, 24, 6])

# Create the Index
index_ = ['Coca Cola', 'Sprite', 'Coke', 'Fanta', 'Dew', 'ThumbsUp']

# set the index
sr.index = index_
result = sr.max()

# Print the result
print(result)
```

g) Given data frame as below: **[4]**

ID	Name	HRA	TA	DA
1001	Mohan	12000	6000	10000
1002	Sachin	13000	5000	9000
1003	Virat	11000	4000	8000

```
import pandas as pd
# making a dict of list
info = {ID: [1001,1002,1003],
        'Name' :
["Mohan","Sachin","Virat"],'HRA':[12000,13000,11000],'TA':[6000,5000,4000],'DA':[100
000,9000,8000]}
data = pd.DataFrame(info)
# Compute sum of each column.
# sum of all HRA TA and DA
data['HRASum'] = data[",'HRA'].sum()
data['TASum'] = data[",'TA'].sum()
data['DASum'] = data[",'DA'].sum()
print(data)
#print mean of each integer column
print(data['HRA'].mean())
```

```
print(data[‘TA’].mean())
print(data['DA'].mean())
# Compute median of each integer column
print(data['HRA'].median())
print(data['TA'].median())
print(data['DA'].median())
```

h) Explain overloading in python with example.**[2M]**
Ans: **Method Overloading:**
Two or more methods have the same name but different numbers of parameters or different types of parameters, or both. These methods are called overloaded methods and this is called method **overloading**.

Example:

```
def product(a, b):
    p = a * b
    print(p)

def product(a, b, c):
    p = a * b*c
    print(p)

# Uncommenting the below line shows an error
# product(4, 5)


# This line will call the second product method
product(4, 5, 5)
```

**********All the Best**********