Q1 Any FIVE

a. Name different modes of Python.

Python has two basic modes:

1. Script mode (or Normal mode)
    - The mode where the scripted and finished .py files are in the Python Interpreter.
2. Interactive mode
    - A command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory.

b. List identity Operators

Operator	Description	Example
is	Returns `True` if both variables are sames object	x is y
is not	Returns `False` if both variables are sames object	x is not y

c. Describe Dictionary

- A dictionary is collection which is unordered, changeable and indexed.
- Dictionaries are written with curl brackets, and they have keys and values.
- Example:
    ```
    company = {
           "name": "Apple",
           "product": "IPhone"
           "model": "11"
    }
    ```

d. State use of namespace in Python

- A namespace is a simple system to control the names in a program.
- Python implements namespaces in the form of dictionaries.
- It maintains a name-to-object mapping where names act as keys and the objects as values.

e. List different object oriented features supported by Python.

- Python OPP Concepts
    i. Object

ii. Class
iii. Method
iv. Inheritance
v. Polymorphism
vi. Data Abstraction
vii. Encapsulation

f. Write steps involved in creation of a user defined exception?

- Exception can be define by creating a new class.
- This exception class has to be derived, either directly or indirectly, from the built-in `Exception`.
- When the programmer suspects the possibility of exception, he should raise his own exception using `raise`.
- The programmer can insert the code inside a `try` block.
- Catch the exception using `except` block.
- Example:

```
class Error(Exception):
       print("Value can't be 0.")

number = 0

try:
       if number == 0:
               raise Error
       else:
               print("Value is more then 0.")
except Error:
       pass
```

- Output:

```
Value can't be 0.
```

g. Describe Python Interpreter

- Python interpreter converts the code written in Python language by users to language which computer hardware or system can understand.
- Python interpreter is a bytecode interpreter, its input is instruction set sets called bytecode.

h. List features of Python

- Easy to code
- High Level programming language

- Object-Oriented Language
- Portable language
- Use interpreter
- GUI Support

Q2 Any THREE

a. Explain two Membership and two logical operators in python with appropriate examples.

Membership Operators

- Membership operators are used to test whether a value is found within a sequence.
- Example of `in`:

  ```
  x = 4
  y = 8
  list = [1, 2, 3, 4, 5]

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

  Output:

  ```
  X is in list array
  ```

- Example of `not in`:

  ```
  if (y not in list):
        print("Y is not in list array")
  else:
        print("Y is in list array")
  ```

  Output:

  ```
  Y is not in list array
  ```

Logical Operators

- Logical operators are usedto perform locical operations on the values of variables. The value is either `true` or `false`
- Example of `and`, `or` and `not`.

```
a = True
b = False
print('a and b is', a and b)
print('a or b is', a or b)
print('not a is', not a)
```

```
a and b is False
a or b is True
a not b is False
```

b. Describe any four methods of lists in Python

- `append()` - Adds an element at the end of the list.
- `pop()` - Removes the element at the specified position.
- `sort()` - Sorts the list
- `clear()` - Removes all the elements from the list. Example:

```
fruits = ['apple', 'banana', 'cherry']

fruits.append("orange")
print(fruits)

fruits.pop(1)
print(fruits)

fruits.sort()
print(fruits)

fruits.clear()
print(fruits)
```

Output:

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

c. Comparing between local and global variable

Local	Global
It is declare inside a function	It is declared outside the function

Local	Global
It is created when the function starts execution and lost when the function terminate.	It is created before the program's global execution starts and lost when the program terminates.
Local variables can be accessed with the help of statements, inside a function in which they are declared.	You can access global variables by any statement in the program.
Parameters passing is required for local variables to access the value in other function.	Parameters passing is not necessary for a global variable as it is visible throughout the program

d. Write a Python program to print Fibonacci series up to n terms

Example:

```
term = int(input("Enter the term: "))

n1, n2 = 0, 1

if term < 0:
        print("Invalid term")
else:
        for i in range(term):
                print(n1)
                nth = n1 + n2
                n1 = n2
                n2 = nth
```

Output:

```
Enter the term: 7
0
1
1
2
3
5
8
```

Q3 Any THREE

a. Write a program to input any two and interchange the tuple variable.

Example:

```
a = (1, 2, 3, 4, 5)
b = (13, 23, 36, 47, 75)

a,b = b,a

print(a)
print(b)
```

Output:

```
(13, 23, 36, 47, 75)
(1, 2, 3, 4, 5)
```

b. Explain different loops available in python with suitable examples.

`while`

A `while` loop executes a target statement as long as given condition is true.

Syntax:

```
while expression: statement(s)
```

Example:

```
count = 0

while(count < 5):
        print(count)
        count += 1

print("over")
```

Output:

```
0
1
2
3
4
over
```

`for loop`

It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax:

```
for iterating in sequence: statements(s)
```

Example:

```
fruits = ['banana', 'apple', 'mango']

for fruit in fruits:
        print(fruit)

print("over")
```

Output:

```
banana
apple
mango
over
```

Nested loops

Python programming language allows to use one loop inside another loop.

Syntax:

```
for iterating in sequence:
        for iterating in sequence:
                statements(s)
        statements(s)
```

Example:

```
nums = [1, 2, 3]
words = ["hello", "hi", "bye"]

for num in nums:
        print(num)

        for word in words:
                print(word)
```

Output:

```
1
hello
hi
bye
2
hello
hi
bye
3
hello
hi
bye
```

c. Describe various modes of file object? Explain any two in detail.

There are four different methods modes for opening a file:

- **r** - *Read* - Opens a file for reading. Error if the file does not exist.
- **w** - *Write* - Opens a file for writing. Creates the file if it does not exist.
- **x** - *Create* - Creates the specified file. Error if file exist.
- **a** - *Append* - Opens a file for appending. Creates the if it does not exist.

In addition, the file should be handled as binary or text mode:

- t - **Text** - Default value - *Text mode*.
- b - Binary - Binary mode (e.g. images).

Read a file

The `read()` method and **r** mode is used to read files. Before read a file, the file must open using `open()` function.

`text.txt` file content:

```
Hello World
```

Program:

```
f = open("text.txt", "r")
print(f.read())
```

Output:

```
Hello world
```

Write a file

The `write()` method and `a` or `w` modes is used to write files.

Program:

```
f = open("text.txt", "w")
f.write("Hello World")
f.close()
```

`text.txt` file content:

```
Hello World
```

d. Illustrate the use of method overriding? Explain with example

If a class inherits a method from its superclass, then there is a chance to override the method provided. Example:

```
class Parent:
        def echo(self):
                print('I am from Parent class.')

class Child(Parent):
        def echo(self):
                print('I am from Child class.')

p = Parent()
c = Child()

p.echo()
c.echo()
```

Output:

```
I am from Parent class.
I am from Child class.
```

Q4 Any THREE

a. Use of any four methods of tuple in Python?

- `len()` - Returns the **length** of the tuple.
- `max()` - Highest value will returned.
- `min()` - Lowest value be returned.
- `count()` - Returns the number of times a specified value occurs in tuple.

Example:

```
t = (12, 45, 43, 8, 35, 12)
print(len(t))
print(max(t))
print(min(t))
print(t1.count(12))
```

Output:

```
6
45
8
2
```

b. Write a python Program to read contents of first.txt and write same content in second.txt file

`first.txt` file content:

```
Hello World
```

Program:

```
with open('first.txt', 'r') as firstfile, open('second.txt', 'a') as secondfile:

        for line in firstfile:
                secondfile.write(line)
```

`second.txt` file content:

```
Hello world
```

c. Show how try...except block is used for exception handling in Python with example.

- When an exception occurs, Python will normally stop and generate an error message.
- These exceptions can be handled using the `try` statement.
- The `except` block lets you handle the error.
- Syntax:

  ```
  try:
          # Code
  except:
          # Code
  ```

- Example:

  ```
  try:
          print(x)
  except NameError:
          print("Variable x is not defined")
  ```

- Output:

  ```
  Variable x is not defined.
  ```

d. Write the output for the following if the variable fruit = "banana"

```
>>> fruit[:3]
>>> fruit[3:]
>>> fruit[3:3]
>>> fruit[:]
```

Output:

```
>>> fruit = "banana"
>>> fruit[:3]
'ban'
>>> fruit[3:]
'ana'
>>> fruit[3:3]
''
>>> fruit[:]
'banana'
```

Q5 Any TWO

a. Determine various data types available in Python with example.

Numbers

- `int`, `float` and complex numbers fall under numbers category.
- Example:

```
a = 5
a = 2.0
a = 1+2j
```

String

- String is sequence of Unicode characters.
- We can use single quotes or double quotes to represent strings.
- Multi-line string can be denoted using triple quotes `'''` or `"""`.
- Example:

```
s = "This is string"
s = '''
       A multi line string
'''
```

List

- List is an ordered sequence of items.
- It is one of the most used datatype in Python.
- List is very flexible.
- All the items in a list don not need to be the same type.
- Example:

```
a = [1, 2.2, 'python']
```

Tuple

- Tuple is an ordered sequence of items same as a list.
- The only difference is that tuples are immutable.
- Tuples once created cannot be modified.
- Example:

```
t = (5, 'program', 1+3j)
```

Set

- Set is an unordered collection of unique items.
- Set is defined by values separated by comma inside braces { }.
- Items in a set are not ordered
- Example:

  ```
  a = {5,2,3,1,4}
  ```

Dictionary

- Dictionary is an unordered collection of key-value pairs.
- It is generally used when we have a huge amount of data.
- Dictionaries are defined within braces `{}`.
- Example:

  ```
  d = {1:'value','key':2}
  ```

b. Write a python program to calculate factorial of given number using function.

Example:

```
n = int(input("Enter the number: "))
f = 1

for i in range(1, n + 1):
        f = f * 1

print(f)
```

Output:

```
Enter the number: 6
720
```

c. Show the output for the following:

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
```

```
>>> [1, 2, 3] * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> t[1:3] = ['x', 'y']
>>> print(t)
['a', 'x', 'y', 'd', 'e', 'f']
```

Q6 Any TWO

a. Describe Set in python with suitable examples.

Creating a set

Set can be created using curly braces `{}` or using `set()` method.

```
fruits = {'apple', 'banana', 'cherry'}
fruits = set(['apple', 'banana', 'cherry'])
```

Adding items to the set

Item can added using `add()` method.

Example:

```
fruits = {'apple', 'banana', 'cherry'}
fruits.add('orange')
print(fruits)
```

Output:

```
{'apple', 'banana', 'cherry', 'orange'}
```

Removing items from the set

There three methods to remove sets:

- `discard()` - removes given items from set.
- `remove()` - removes given item from set. If item is not avaliable it will give error.
- `pop()` - removes list item from the set.

Example:

```
fruits = {'apple', 'banana', 'cherry', 'orange'}
fruits.discard("apple")
fruits.remove("banana")
fruits.pop()
print(fruits)
```

Output:

```
{'cherry'}
```

Comparison of set

- `|` - shows the union of two set.
- `&` - shows the intersection of two set.
- `-` - shows the difference of two set.
- `<`, `>`, `<=`, `>=`, `==` - comparison operators can also be use in set.

```
fruits = {'apple', 'banana', 'cherry'}
fruits2 = {'orange', 'pineapple', 'apple'}

print(fruits|fruits2)
print(fruits&fruits2)
print(fruits-fruits2)
print(fruits>fruits2)
print(fruits<fruits2)
print(fruits==fruits2)
```

Output:

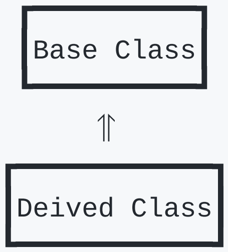
```
{'apple', 'banana', 'cherry', 'orange', 'pineapple', 'apple'}
{'apple'}
{'banana', 'cherry', 'orange', 'pineapple'}
False
False
True
```

b. Illustrate class inheritance in Python with an example

Simple Inheritance

In inheritance, the child class acquires the properties and access all the data members and function defined in the parent class.

Illustration:



Syntax:

```
class Base:
        # Body of base class
class Derived(Base):
        # Body of derived class
```

Example:

```
class Parent:
        parentname = ""
        childname = ""
        def show_parent(self):
                print(self.parentname)

class Child(Parent):
        def show_child(self):
                print(self.childname)

c = Child()
c.parentname = "Arati"
c.childname = "Purva"
c.show_parent()
c.show_child()
```

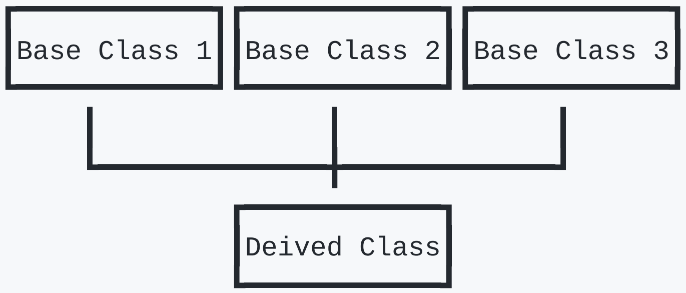
Output:

```
Arati
Purva
```

Multiple inheritance

Multiple inheritace means that you're inheriting the property of multiple classes into one.

Illustration:



Syntax:

```
class A:
        # variable of class A
class B:
        # variable of class B
class C(A, B):
        # variable of class C
```

Example:

```
class Parent1:
        def echo(self):
                print("Parent class 1")

class Parent2:
        def echo2(self):
                print("Parent class 2")

class Child(Parent1, Parent2):
        def show(self):
                print("Child class")

c = Child()
c.echo()
c.echo2()
c.show()
```

Output:

```
Parent class 1
Parent class 2
Child class
```

c. Design a class Employee with data members: name, department and salary. Create suitable methods for reading and printing employee information

Example:

```python
class Employee:
        name = ""
        department = ""
        salary = 0

        def setData(self):
                self.name = input("Enter Name: ")
                self.department = input("Enter Department: ")
                self.salary = int(input("Enter Salary: "))

        def showData(self):
                print("Name:", self.name)
                print("Department:", self.department)
                print("Salary:", self.salary)

e = Employee()
e.setData()
e.showData()
```

Output:

```
Enter Name: Jonney
Enter Department: Testing
Enter Salary: 20000
Name: Jonney
Department: Testing
Salary: 20000
```