1. Write a python program to create and access the elements of array with an example?
2. Explain Numpy arrays with an example program? Explain Slicing on Numpy Arrays?
3. What are the advantages of arrays? Write a program to search a numeric value in a NumPy array.

UNIT 4

4. What are the categories of arguments in functions and explain with an example?
5. Define python Recursion function? Write a python program to factorial using recursion?
6. Write a Python function that takes two lists and returns True if they have at least one common member?
7. What is lambda function? What are the characteristics of a lambda function? Give an example.
8. What are Python OOPs Concepts?
9. Define inheritances and explain with simple example programs?
10. Define Class and Object? How to create Class and Object with simple example Program?

UNIT 5

11. Define Exception? List any 6 types of exception?
12. What is exception? Write the syntax and program to handle exceptions?
13. What are the keywords in exception handling? Differentiate else block and finally block in exception handling?
14. Define file and Explain the different types of file modes in python?
15. Write a python program to open a file Write a program to copy data from FILE1 to FILE2.?

TEAM VISION

TEAM VISION
BE WITH PEOPLE THAT BRING BEST IN YOU

DONE AND PRESENTED BY-

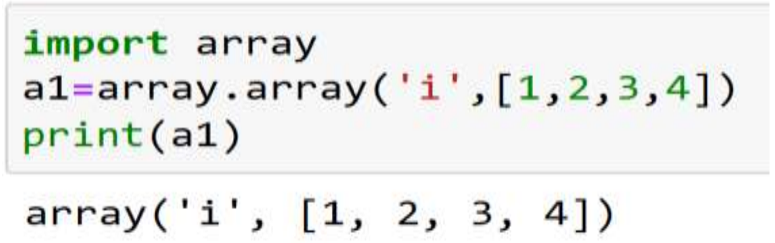
TEAM VISION

Contact:-
teamvision1729@gmail.com

UNIT-3

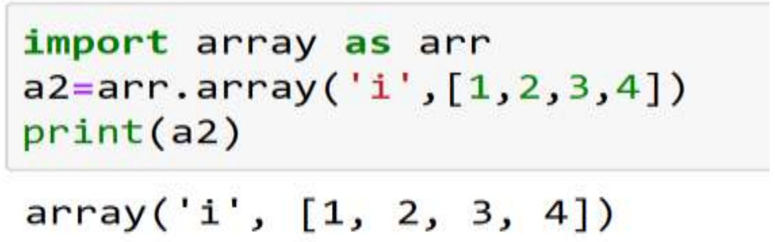
1. **Write a python program to create and access the elements of array with an example?**

Ans: Program to create and access the elements of array.



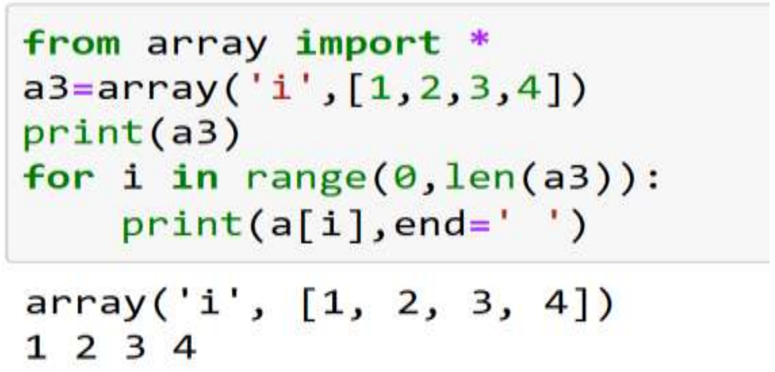
```python
import array
a1=array.array('i',[1,2,3,4])
print(a1)
```

```
array('i', [1, 2, 3, 4])
```



```python
import array as arr
a2=arr.array('i',[1,2,3,4])
print(a2)
```

```
array('i', [1, 2, 3, 4])
```



```python
from array import *
a3=array('i',[1,2,3,4])
print(a3)
for i in range(0,len(a3)):
    print(a[i],end=' ')
```

```
array('i', [1, 2, 3, 4])
1 2 3 4
```

2. Explain NumPy array with an example program? Explain slicing on NumPy Arrays.

Ans.

- **Numpy:-**

-NumPy is a module. The NumPy's array class is known as ndarray. A numpy array is not the same as the standard Python library class array. We can create multi-dimensional arrays using the numpy module.

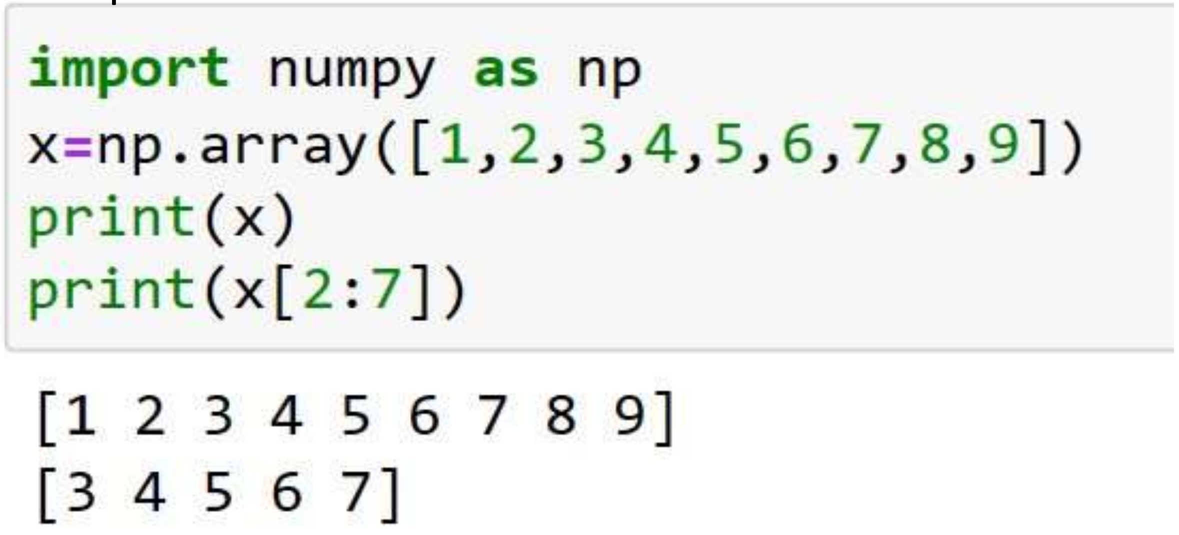
-A NumPy array is a collection of elements, all of the same data type, and is indexed by a tuple of non-negative integers.

- **Slicing arrays:-**

-Slicing in python means taking elements from one given index to another given index from a sequence or string.

-We slice a sequence or string by using range of index: [start:end:step]. The step is optional. The default start is 0 index and the default end is index of the last element and the default step is 1.

Example code-



```python
import numpy as np
x=np.array([1,2,3,4,5,6,7,8,9])
print(x)
print(x[2:7])
```

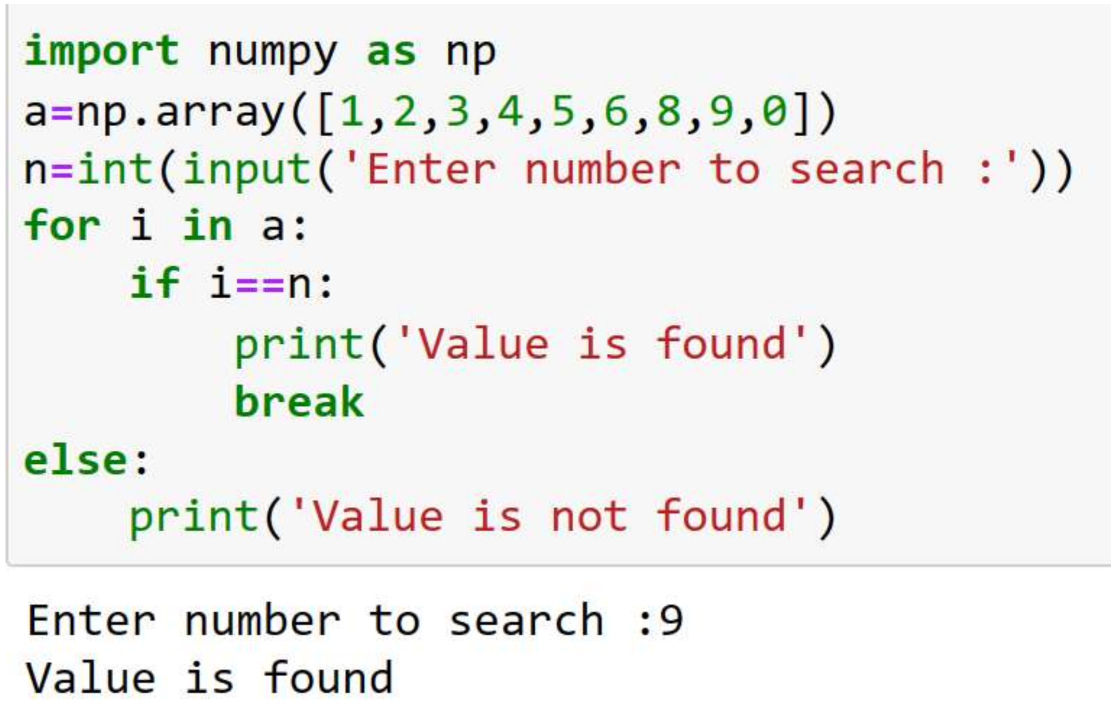
```
[1 2 3 4 5 6 7 8 9]
[3 4 5 6 7]
```

3. What are the advantages of arrays? Write a program to search a numeric value in a NumPy array.

Ans.

- Efficient Storage: Arrays are used to store multiple items in a single variable. This makes it easier to manage and organize data in an efficient manner.
- Easy Access: Elements in arrays can be easily accessed and manipulated by their indices, making it simpler to perform operations like search and sort.
- Dynamic Sizing: Arrays can be dynamically resized as per the requirement, making it easier to handle unexpected data growth.

Program to search a numeric value in a NumPy array:-



```python
import numpy as np
a=np.array([1,2,3,4,5,6,8,9,0])
n=int(input('Enter number to search :'))
for i in a:
    if i==n:
        print('Value is found')
        break
else:
    print('Value is not found')
```

```
Enter number to search :9
Value is found
```

UNIT-4

4. What are the categories of arguments in functions and explain with an example?

Ans: There are 4 categories of arguments in functions they are:

a. Required Argument

b. Default Argument

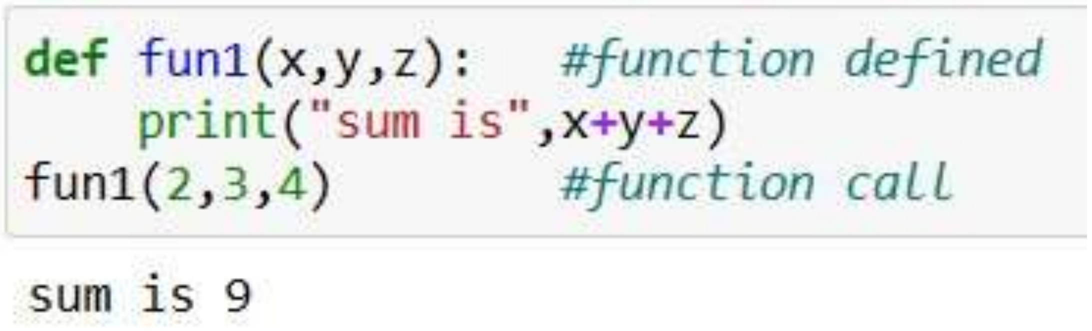
c. Keyword Argument

d. Arbitrary Argument

e. Arbitrary keyword argument

A. REQUIRED ARGUMENT –

- Required arguments are the arguments passed to a function in the **correct positional order.**
- Here, the number of arguments in the function call should match exactly with the function definition.
- Here the example function defined fun1 takes three required parameters. x,y,z values from the function call.
- We should provide the function cell's arguments x, y, and z values.



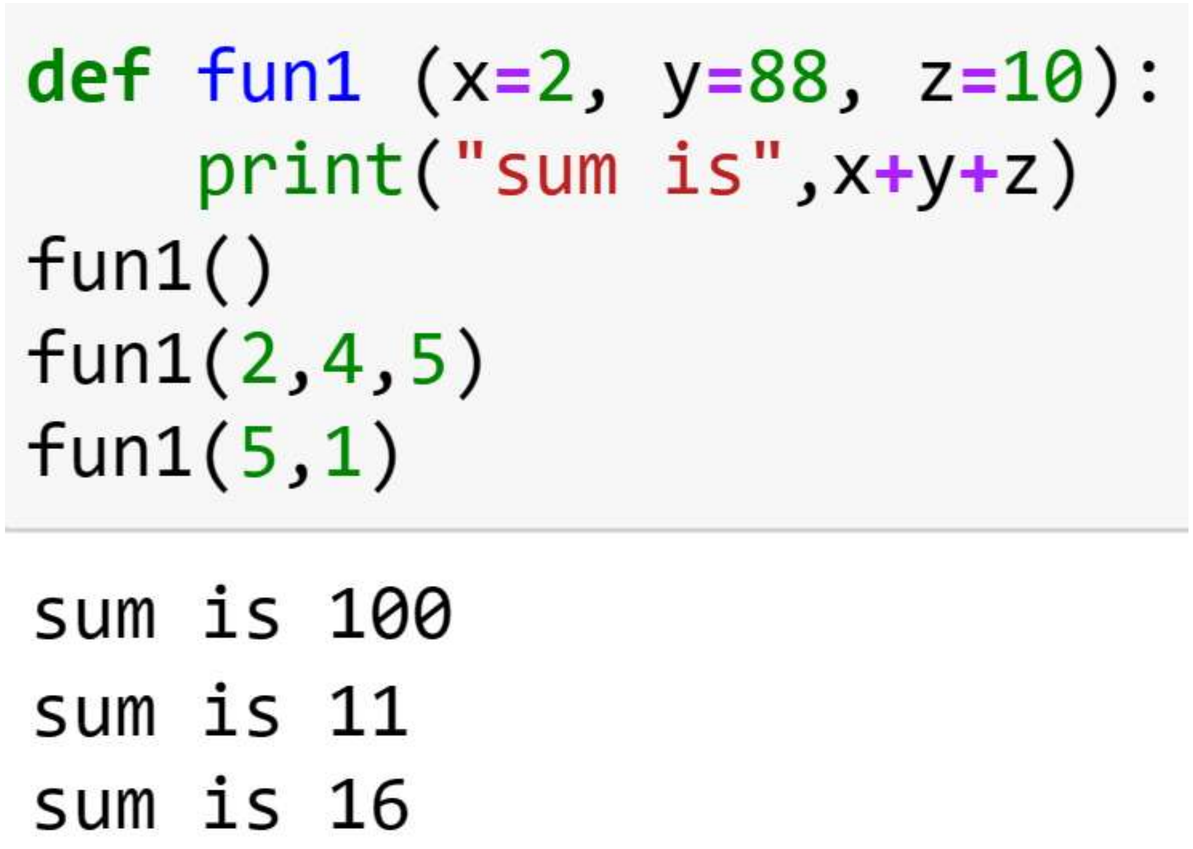
```
def fun1(x,y,z):    #function defined
    print("sum is",x+y+z)
fun1(2,3,4)         #function call
```

```
sum is 9
```

B. DEFAULT ARGUMENT-

- Default arguments are values that are provided while defining the function.
- Arguments become optional during the function calls.
- If we provide a value to the default arguments during function calls, it overrides the default value.

#code-

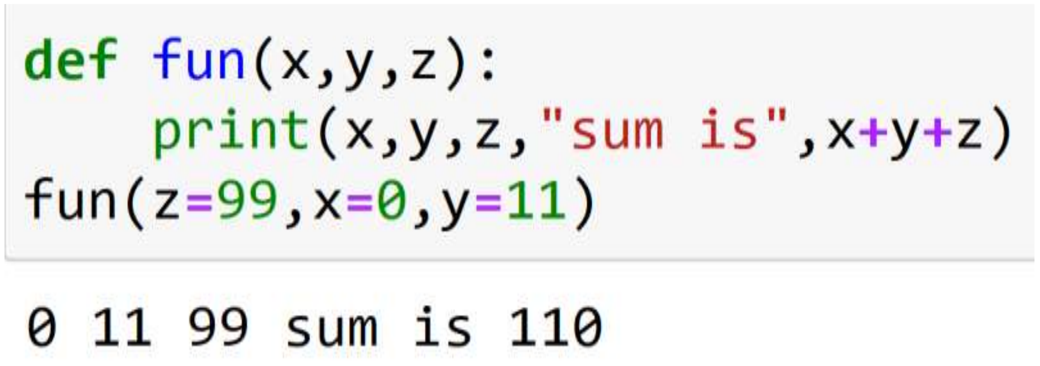


```python
def fun1 (x=2, y=88, z=10):
    print("sum is",x+y+z)
fun1()
fun1(2,4,5)
fun1(5,1)
```

```
sum is 100
sum is 11
sum is 16
```

C. KEYWORD ARGUMENTS-

- Functions can also be called using keyword arguments of the form “parameter=value”.
- During a function call, values passed through arguments need not be in the order of parameters in the function definition. This can be achieved by keyword arguments. But all the keyword arguments should match the parameters in the function definition.



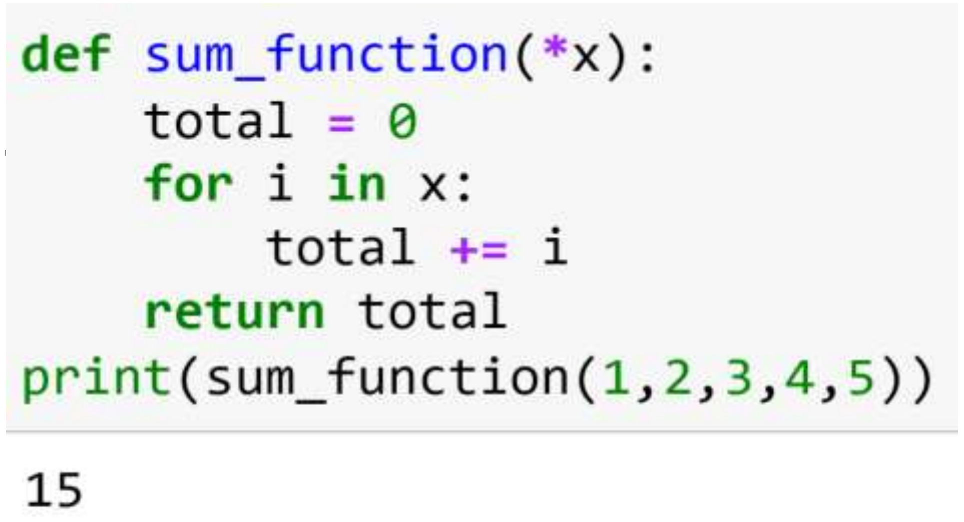
```python
def fun(x,y,z):
    print(x,y,z,"sum is",x+y+z)
fun(z=99,x=0,y=11)
```

```
0 11 99 sum is 110
```

D. ARBITRARY ARGUMENTS-

- Variable-length arguments are also known as **arbitrary arguments**.
- If we don’t know the number of arguments needed for the function.
- in advance, we can use the arbitrary argument.
- In the arbitrary arguments the function definition takes only one
- Parameter but in function call may have more
  a number of parameters.

#code for arbitrary arguments:-



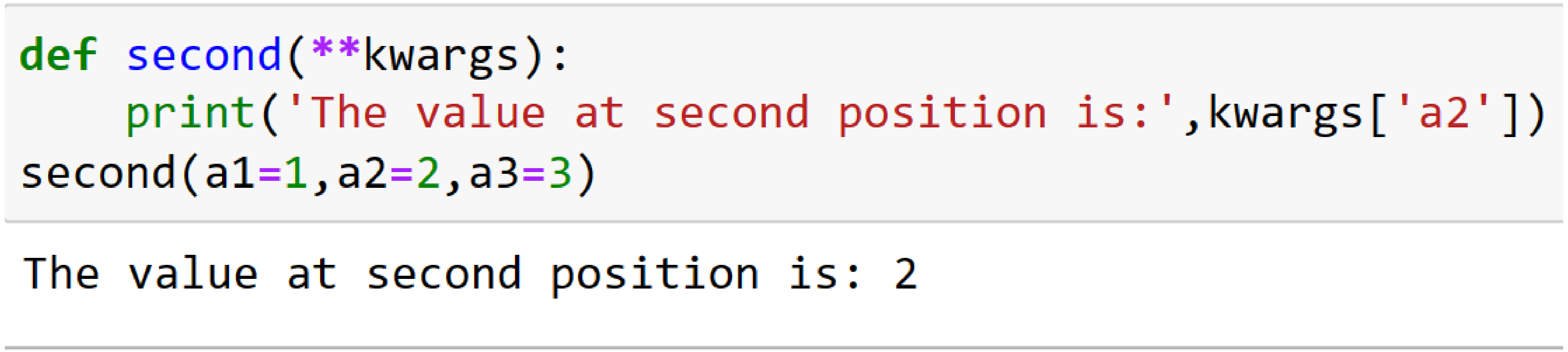
```python
def sum_function(*x):
    total = 0
    for i in x:
        total += i
    return total
print(sum_function(1,2,3,4,5))
```

```
15
```

E. ARBITRARY KEYWORD ARGUMENTS-

- It is the same as arbitrary arguments.
- But we can give arguments in the form of ‘parameter=value’ at the time of function calling.

#code for arbitrary keyword arguments



```
def second(**kwargs):
    print('The value at second position is:',kwargs['a2'])
second(a1=1,a2=2,a3=3)
```

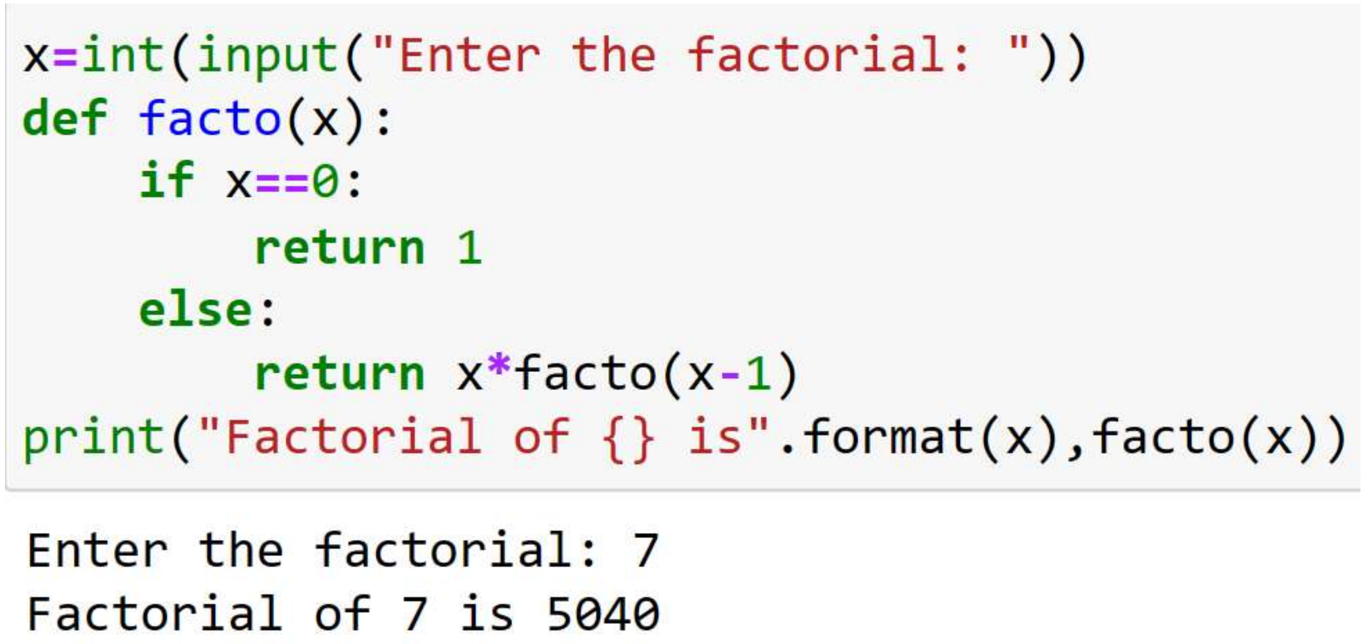
```
The value at second position is: 2
```

5. Define the python Recursion function? Write a python program to factorial using recursion.

Ans:

- Recursion means a defined function can call itself is called a Recursion function.
- The Recursive functions are used to make the iterative loop-like structure to evaluate a certain task or to reach a result.
- This can be useful for solving problems that can be broken down into smaller subproblems that are similar to the original problem.

#code-



```python
x=int(input("Enter the factorial: "))
def facto(x):
    if x==0:
        return 1
    else:
        return x*facto(x-1)
print("Factorial of {} is".format(x),facto(x))
```

```
Enter the factorial: 7
Factorial of 7 is 5040
```

6. Write a python function that takes two lists and returns True if they have at least one common member?

Ans:

```python
def common(a,b):
    c=len(set(a).intersection(b))
    if c>=1:
        return True
    else:
        return False
list1=[1,2,3,4]
list2=[5,2,1,8]
common(list1,list2)
```

```
True
```

7. What is lambda function? What are the characteristics of a lambda function? Give an example.

Ans.

Lambda function:-

- A lambda function in python is a small anonymous function that can take any number of arguments but only have one expression.
- It is used to reduce the usage of functions.

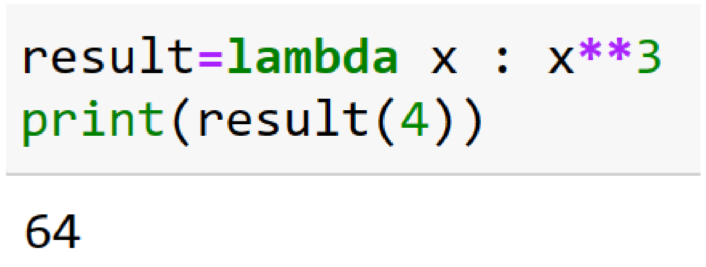
Syntax:-

lambda arguments: expression

Characteristics of lambda functions:-

- Anonymous: Lambda functions are anonymous and do not have a name like normal functions.
- Single Expression: A lambda function can only contain a single expression, which is executed and returned when the function is called.
- Arguments: A lambda function can take any number of arguments, but the expression within the function can only access these arguments.

-Example program to print cube of a number using lambda function:-



```python
result=lambda x : x**3
print(result(4))
```

```
64
```

8. What are python OOPs concepts?

Ans: (OOPs)-Object Oriented Programming:-

- The main aim of object-oriented programming is to develop real-world entities.
- Object-oriented programming satisfies the mandatory concepts of the following:-

➢ Class

➢ Object

➢ Inheritance

➢ Abstraction

➢ Polymorphism

- **CLASS:-**

• A class is a structure or plan or blueprint or model to define an object.

• We can also create user-defined classes by using the keyword classes.

- **OBJECT:-**

• An object is a real-time entity, it is also called an instance of a class. {Objectname=Classname}

- **INHERITANCE:-**

- Inheritance is the process of creating a new class from an already existing class.
- Inheritance is also defined as an object of one class that has the properties of an object of another class.

The inheritances are:-

-Single inheritance

-Multi-level inheritance

-Multiply inheritance

-Hierarchical inheritance

- **ABSTRACTION**:-

- Abstraction is defined as a process of handling complexity by hiding unnecessary information from the user.
- The user is kept unaware of the basic implementation of a function property. The user is only able to view basic functionalities whereas the internal details are hidden.

- **POLYMORPHISM**:-

- Invoking functions are based on different parameters
- Invoking functions based on different parameters. Polymorphism is the process of exhibiting different behaviours in different instances.
- Example-Operator overloading and operator function overloading.

9. Define inheritance and briefly about different types of inheritances with simple example programs.

Ans. **INHERITANCE**:-

- Inheritance is the process of creating a new class from an already existing class.
- Inheritance is also defined as an object of one class that has the properties of an object of another class.

The inheritances are:-

-Single inheritance

-Multi-level inheritance

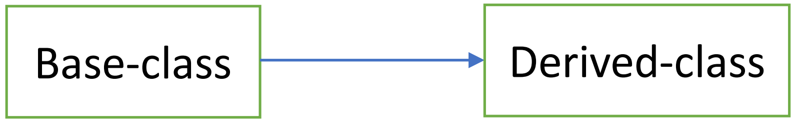
-Multiply inheritance

-Hierarchical inheritance

➢ **SINGLE INHERITANCE:-**

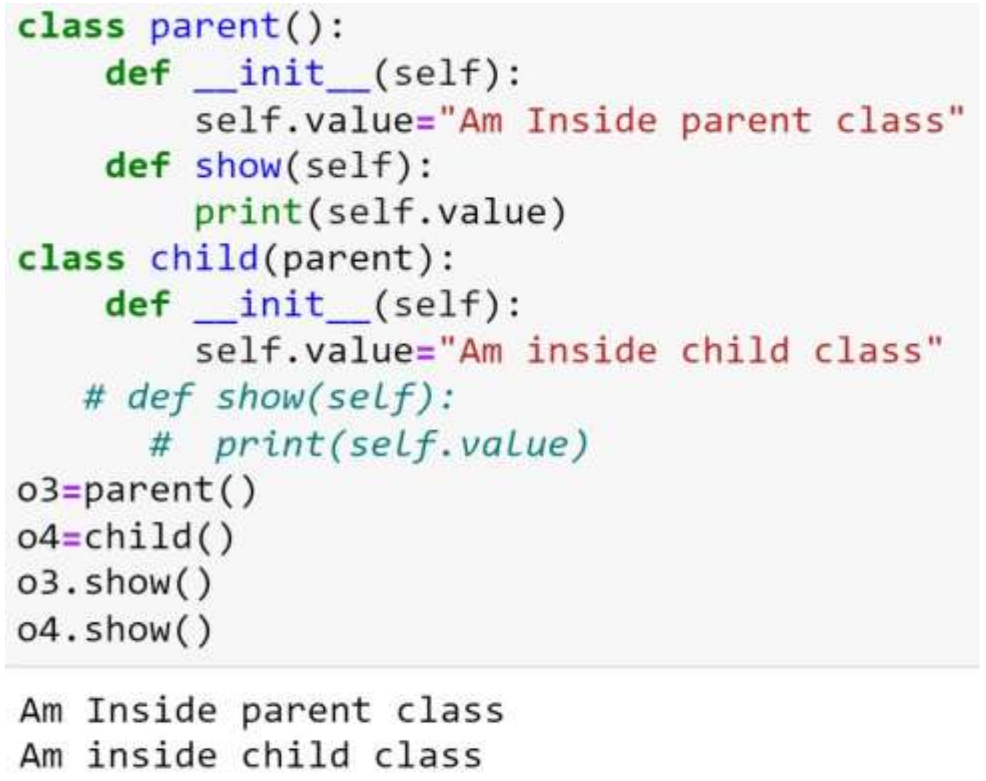
- In single inheritance a derived class

  is created using a single-based class.



#Parent class #Child class

#single inheritance code-



```
class parent():
    def __init__(self):
        self.value="Am Inside parent class"
    def show(self):
        print(self.value)
class child(parent):
    def __init__(self):
        self.value="Am inside child class"
   # def show(self):
      #  print(self.value)
o3=parent()
o4=child()
o3.show()
o4.show()
```

```
Am Inside parent class
Am inside child class
```

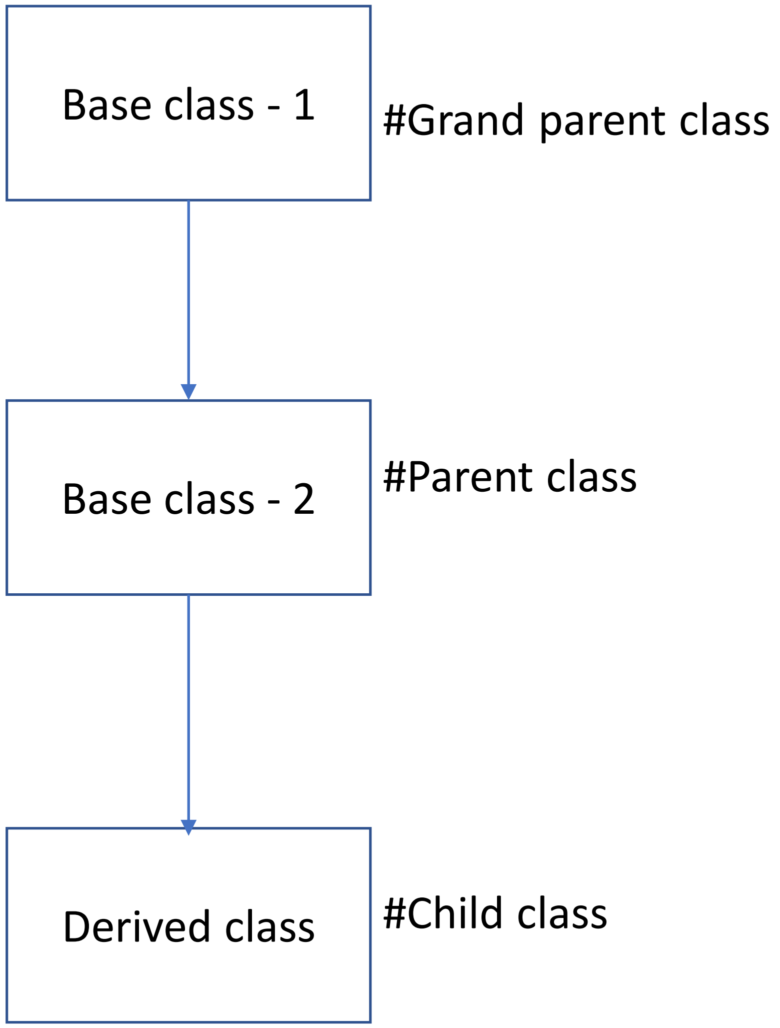
➢ MULTI LEVEL INHERITANCE:-

- In this every new class is derived from a single-parent class.

#code for multi-level inheritance

```
class Grandparent:
    def gpf(self):
        print('This is grand parent class')
class Parent(Grandparent):
    def pf(self):
        print('This is parent class')
class Child(Parent):
    def cf(self):
        print('This is child class')
ch=Child()
ch.cf()
ch.pf()
ch.gpf()
```

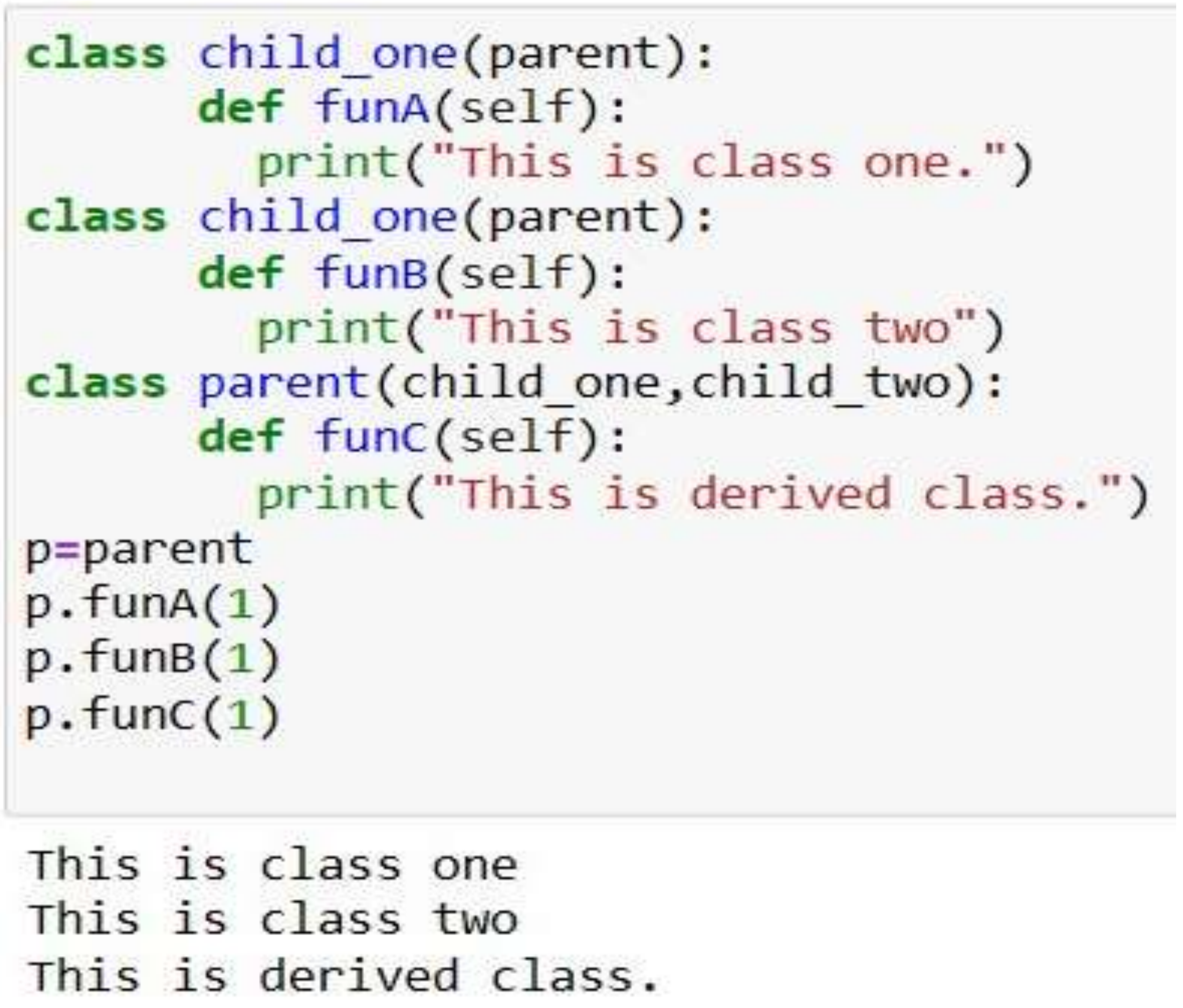
```
This is child class
This is parent funtion
This is grand parent class
```



➢ MULTIPLE-LEVEL INHERITANCE:-

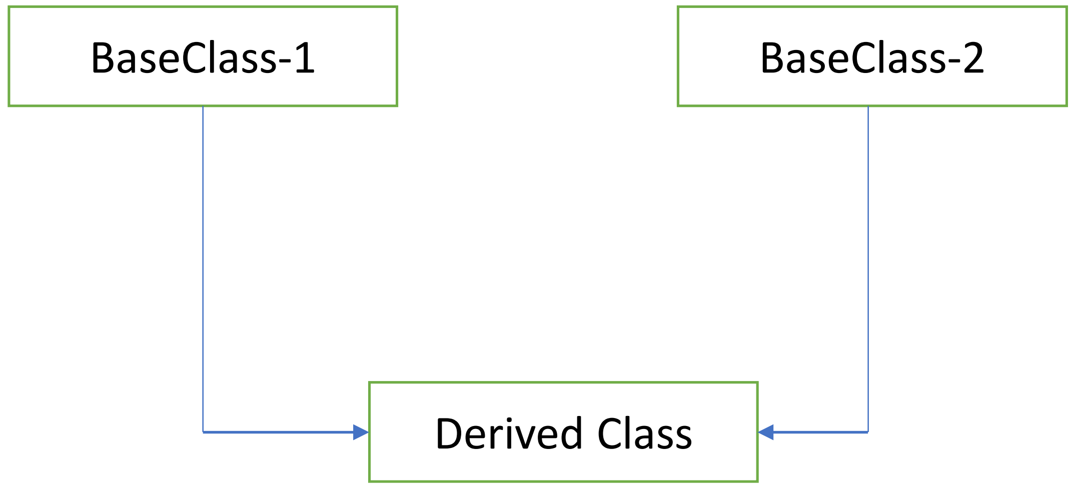
- In Multiple inheritances a derived class is created by using two or more base classes.

#optional code for studying



```
class child_one(parent):
        def funA(self):
          print("This is class one.")
class child_one(parent):
        def funB(self):
          print("This is class two")
class parent(child_one,child_two):
        def funC(self):
          print("This is derived class.")
p=parent
p.funA(1)
p.funB(1)
p.funC(1)
```

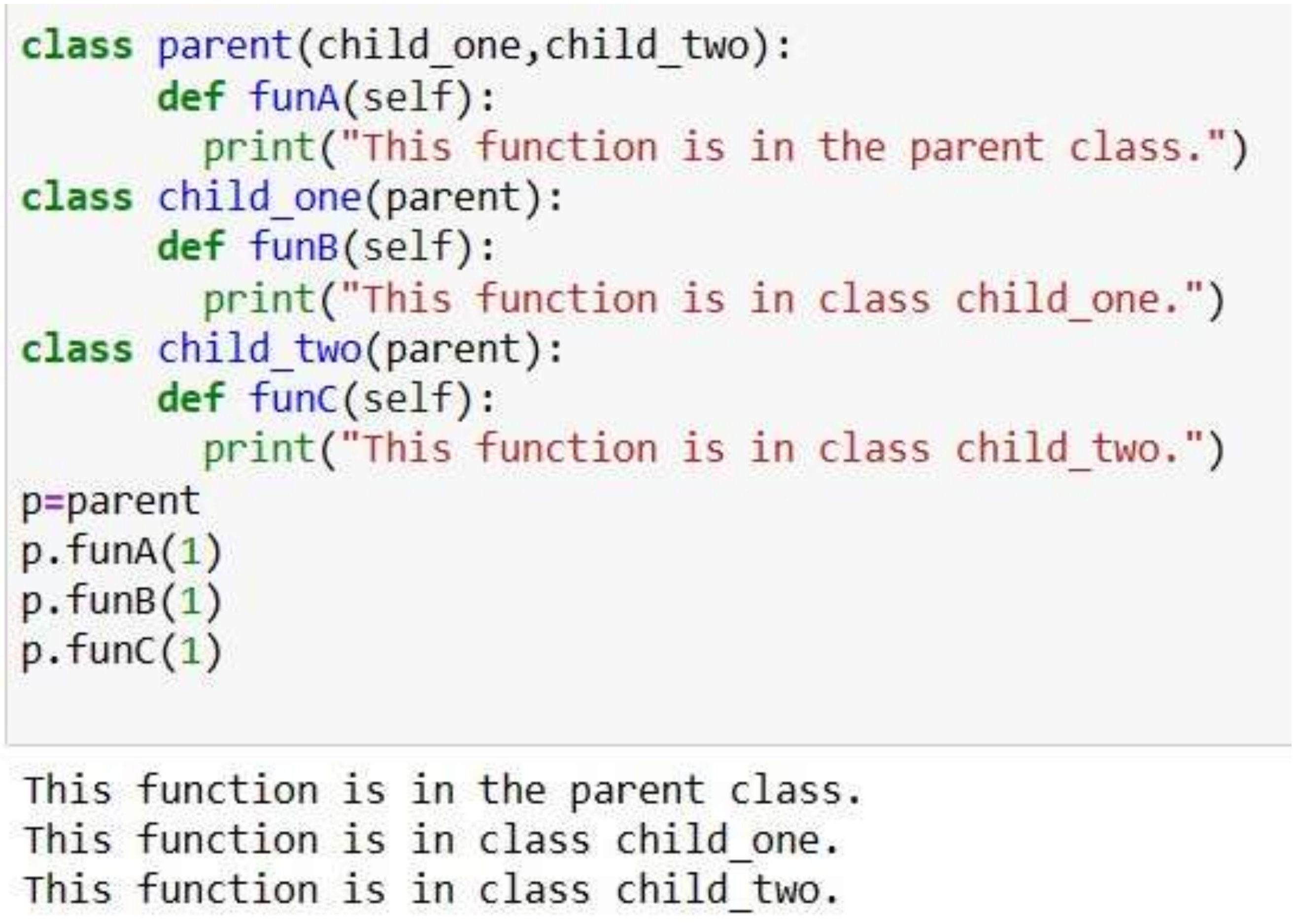
```
This is class one
This is class two
This is derived class.
```



➢ HIERARCHICAL INHERITANCE:-

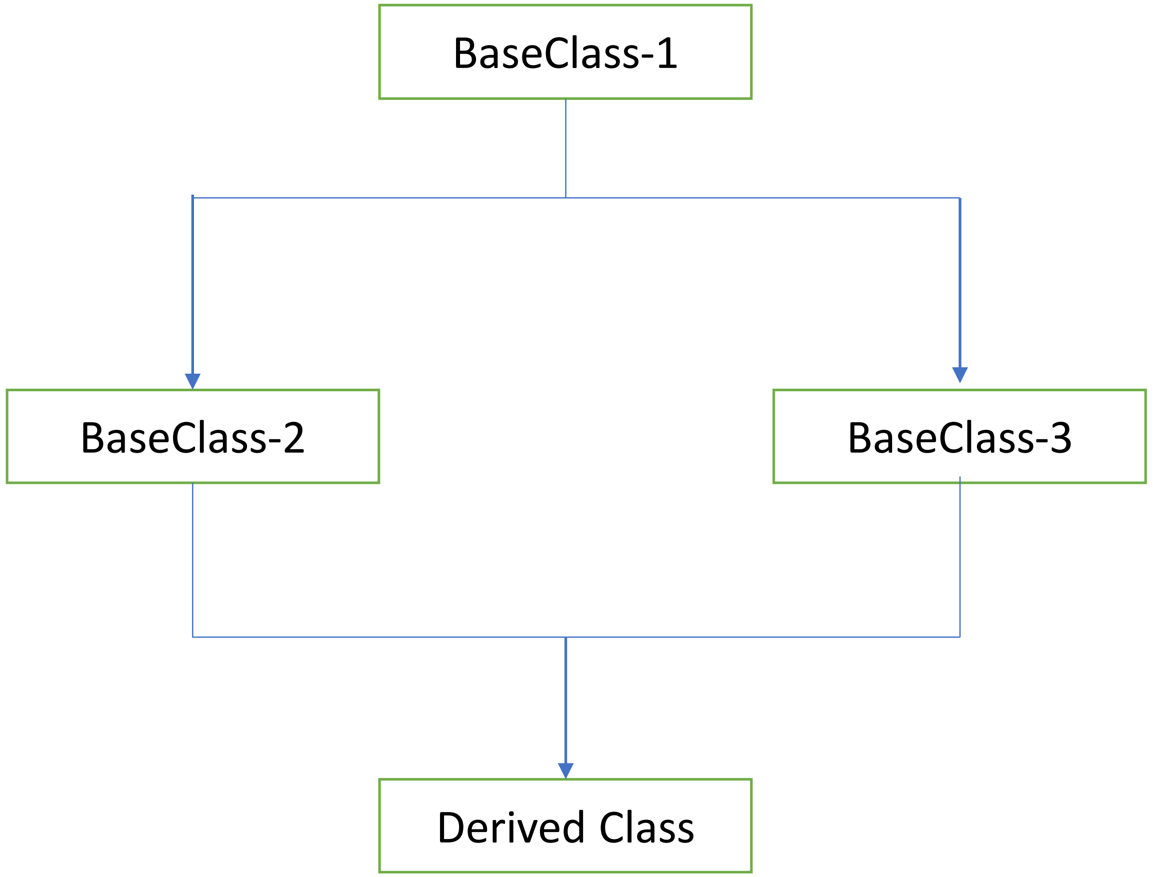
- Hierarchical inheritance is the process of creating a child class from a single-parent class and also multiple-base classes.

#optional code for studying -



```
class parent(child_one,child_two):
      def funA(self):
        print("This function is in the parent class.")
class child_one(parent):
      def funB(self):
        print("This function is in class child_one.")
class child_two(parent):
      def funC(self):
        print("This function is in class child_two.")
p=parent
p.funA(1)
p.funB(1)
p.funC(1)
```

```
This function is in the parent class.
This function is in class child_one.
This function is in class child_two.
```



10. Define a class and object? How to create Class and Object with a simple example program?

Ans.

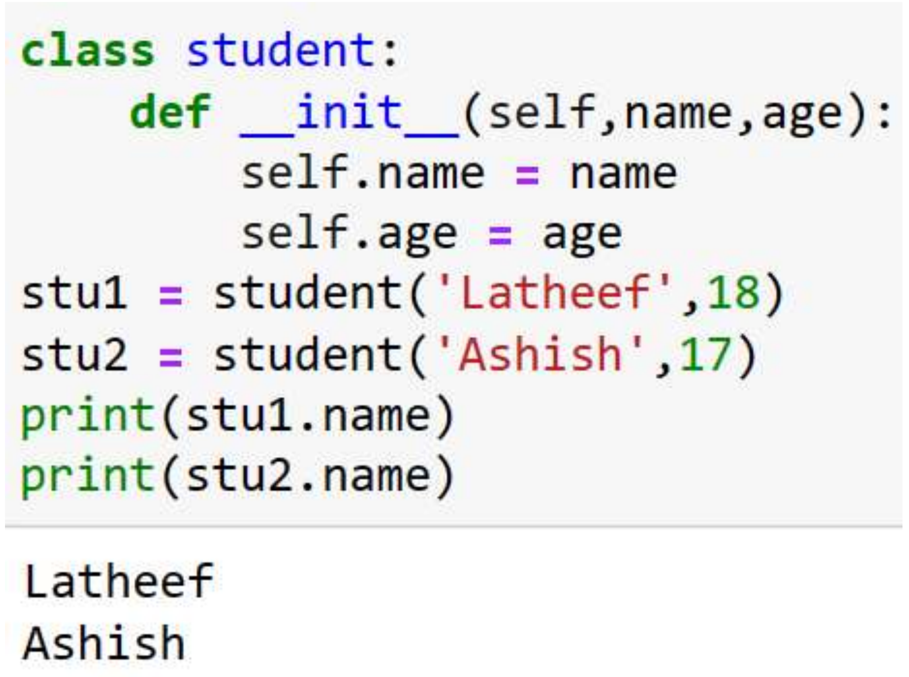
Class:-

A class is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behaviour (member functions or methods).

Object:-

An object is an instance of a class, and it contains the data and behaviour defined by the class. When an object is created, it is called an instance of the class. An object can access the attributes and methods defined in its class.

-Example program creating class and object:-



```python
class student:
    def __init__(self,name,age):
        self.name = name
        self.age = age
stu1 = student('Latheef',18)
stu2 = student('Ashish',17)
print(stu1.name)
print(stu2.name)
```

```
Latheef
Ashish
```

Here, student is a class and stu1, stu2 are objects

UNIT-5

11. Define Exception? List any 6 types of Exceptions.

Ans:

- An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instruction. In general, when a python script encounters a situation that it cannot cope with, it raises an exception. An exception is a python object that represents an error.

Exceptions are classified as follows-

- **File not found-** This error is raised when the file is not available.
- **Index error-** This error is raised when the index to sequence is out of bound.
- **Key error-** This error is raised when the non-existent error key is requested for a set or dictionary.
- **Name-error-** Non-existent identifies used.
- **Type-error-** This error is raised when the wrong type of parameter is sent to the function.
- **Zero-division error-** This exception is raised when the division is done by using zero.

12.What is Exception? Write the syntax and program to handle exceptions?

Ans:

Exception:-

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error. When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

Syntax:-

```
try:
   #statement(s) where error may exist
except ExceptionName:
   #statement(s) to be executed when exception raises
else:
   #statement(s) to be executed if no exception is raised
finally:
   #statement(s) that are always executed
```

Example program for handling exception:-

```
try:
    a=int(input('Enter a value: '))
    b=int(input('Enter b value: '))
except(ValueError):
    print('Enter a valid number')
else:
    print('sum =',a+b)
finally:
    print('End of the program')
```

```
Enter a value: 5
Enter b value: a
Enter a valid number
End of the program
```

13. What are keywords in exception handling? Differentiate else block and finally block in exceptional handling?

Ans.

-Following are the keywords of exception handling:-

- ‘try’: The try block encloses the code that might raise an exception.
- ‘except’: The except block is executed if an exception is raised in the try block. It contains the code to handle the exception.
- ‘else’: The else block is executed if no exception is raised in the try block skipping the except block.
- ‘finally’: The finally block is executed after the try and except blocks. It contains code that will be executed regardless of whether an exception was raised or not.

-Following is the difference between else block and finally block:-

else block	finally block
➢ The else block is executed if no exception is raised in the try block. It contains code that will be executed only if the try block completes without raising an exception. The else block is executed after the try block and before the finally block.	➢ The finally block is executed after the try and except blocks. It contains code that will be executed regardless of whether an exception was raised or not. The finally block is used to clean up resources, such as closing files or releasing memory, that should be executed regardless of the outcome of the try block.

14. Define File and explain different types of file modes in python.

Ans:

File: A file is a collection of data stored on a computer's storage device with a unique name and location. In Python, a file can be opened using the open function and can be read, written, or appended to, depending on the mode in which it was opened.

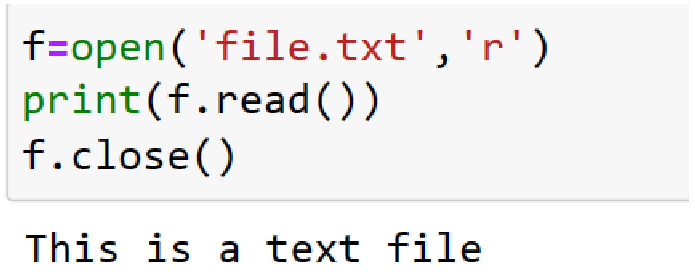
Different types of file modes in Python:

- 'r' (Read-only mode): Open the file for reading. This is the default mode.
- 'w' (Write mode): Open the file for writing. If the file already exists, its contents are overwritten. If the file does not exist, a new file is created.
- 'a' (Append mode): Open the file for writing and append data to the end of the file. If the file does not exist, a new file is created.
- 'x' (Exclusive creation mode): Open the file for writing, but only if it does not already exist. If the file exists, an error is raised. 'b' (Binary mode): Open the file in binary mode for reading or writing binary data.
- 't' (Text mode): Open the file in text mode for reading or writing text data. This is the default mode.

15. **Write a python program to open a file Write a program to copy data from FILE1 to FILE2. ?**

Ans.

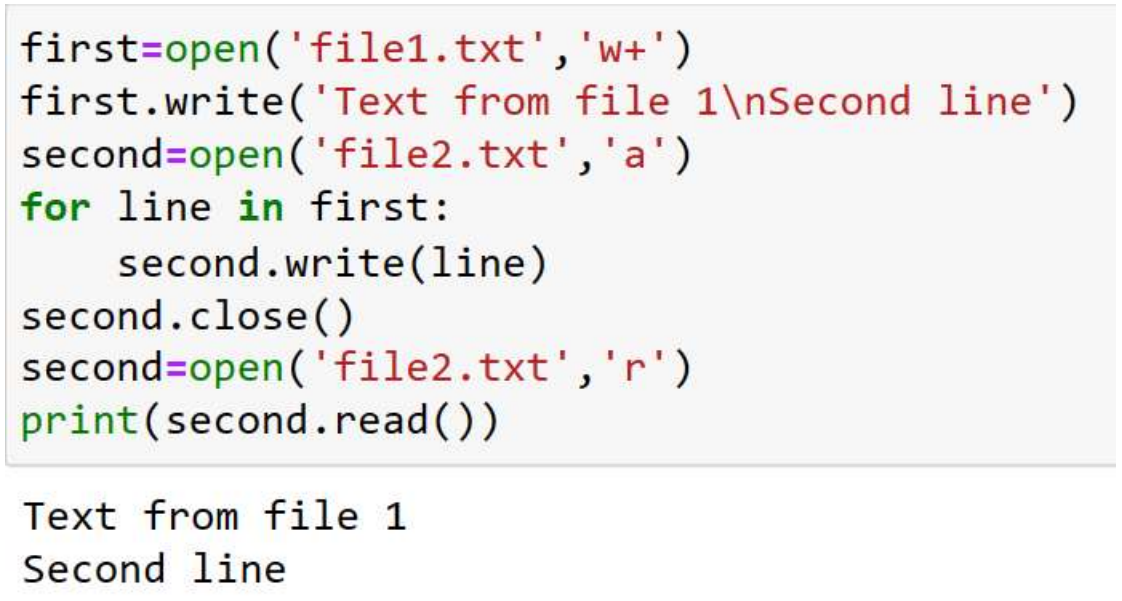
Program to open and read a file:-



```
f=open('file.txt','r')
print(f.read())
f.close()
```

```
This is a text file
```

Program to copy data from ‘file1’ to ‘file2’:-



```
first=open('file1.txt','w+')
first.write('Text from file 1\nSecond line')
second=open('file2.txt','a')
for line in first:
    second.write(line)
second.close()
second=open('file2.txt','r')
print(second.read())
```

```
Text from file 1
Second line
```