MALLA REDDY UNIVERSITY

Python Programming Question Bank for Minor - 2

UNIT 3

1. Write a python program to create and access the elements of array?
2. Explain Numpy arrays with an example program? Explain Slicing on Numpy Arrays?

UNIT 4

1. What are the categories of arguments in functions and explain with an example?
2. Define python Recursion function? Write a python program to factorial using recursion?
3. Write a Python function that takes two lists and returns True if they have at least one common member?
4. What are Python OOPs Concepts?
5. Define inheritance and brief about different types of inheritances with at least two example Programs?
6. What is Constructor and Method Overriding explain with an example Program?

UNIT 5

1. Define Exception? List any 6 types of exception?
2. Write the syntax and program to handle exceptions?
3. Explain about Raise an Exceptions?
4. Define file and Explain the different types of file modes in python?
5. Write a python program to open a file and check what are the access permissions acquired by that file using os module?

TEAM VISION

TEAM VISION

BE WITH PEOPLE THAT BRING BEST IN YOU

DONE AND PRESENTED BY-

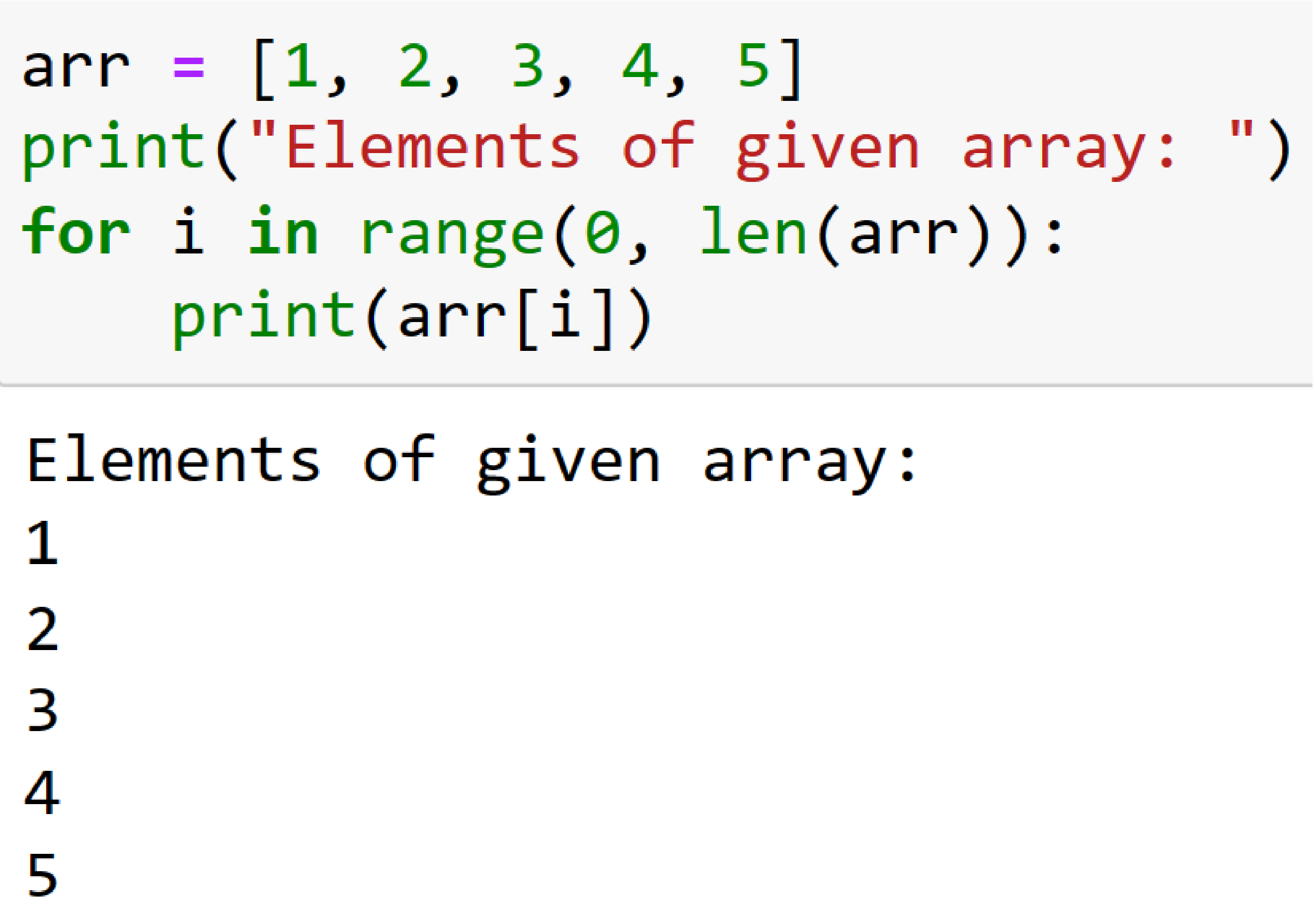
TEAM VISION

Contact:-
teamvision1729@gmail.com

UNIT-3

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

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



```
arr = [1, 2, 3, 4, 5]
print("Elements of given array: ")
for i in range(0, len(arr)):
    print(arr[i])
```

```
Elements of given array:
1
2
3
4
5
```

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.

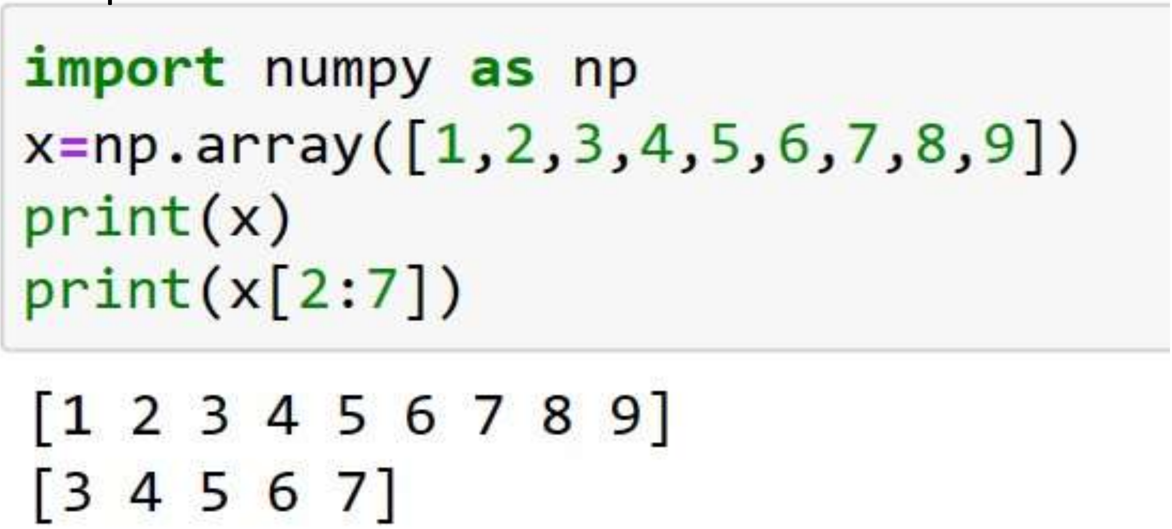
-A numpy array is a grid of values, all of the same types, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array, and the shape of an array is a tuple of integers giving the size of the array along each dimension.

- 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 a range of index: [start:end:step]. 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-



```
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]
```

UNIT-4

1. **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

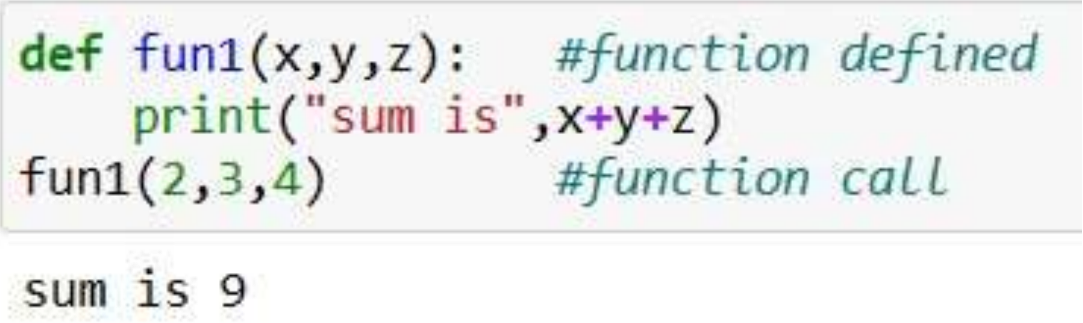
b. Default Argument

c. Keyword Argument

d. Arbitrary Argument

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.



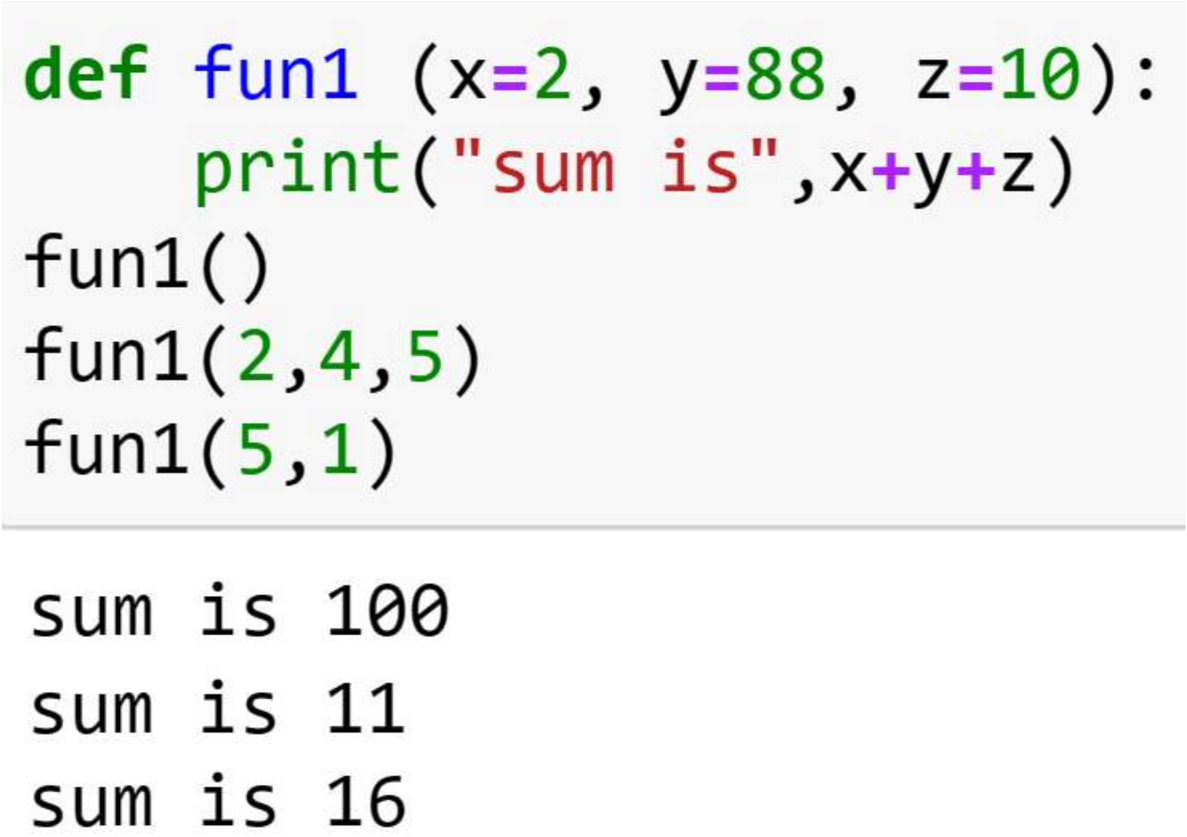
```
def fun1(x,y,z):   #function defined
    print("sum is",x+y+z)
fun1(2,3,4)        #function call
```

```
sum is 9
```

B. DEFAULT ARGUMENT-

- Default arguments are values that are provided while defining the function.
- Default 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-

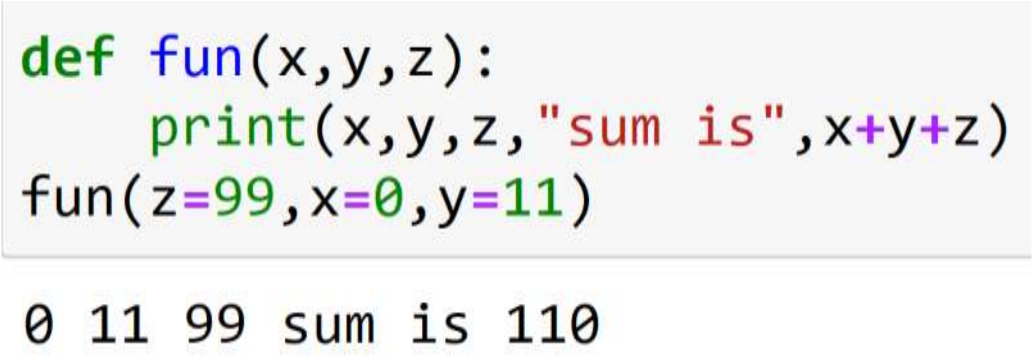


```python
def fun1 (x=2, y=88, z=10):
    print("sum is",x+y+z)
fun1()
fun1(2,4,5)
fun1(5,1)
```

```
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.



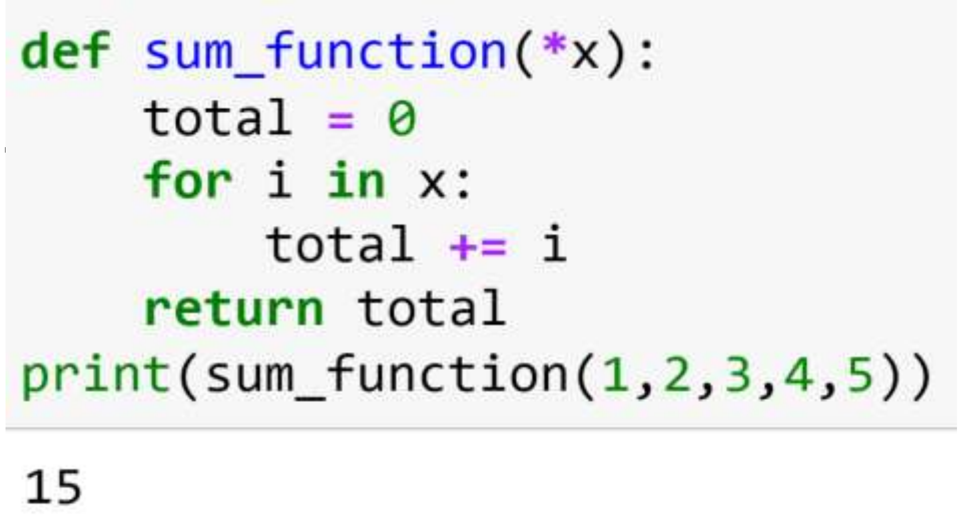
```
def fun(x,y,z):
    print(x,y,z,"sum is",x+y+z)
fun(z=99,x=0,y=11)
```

```
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:-



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

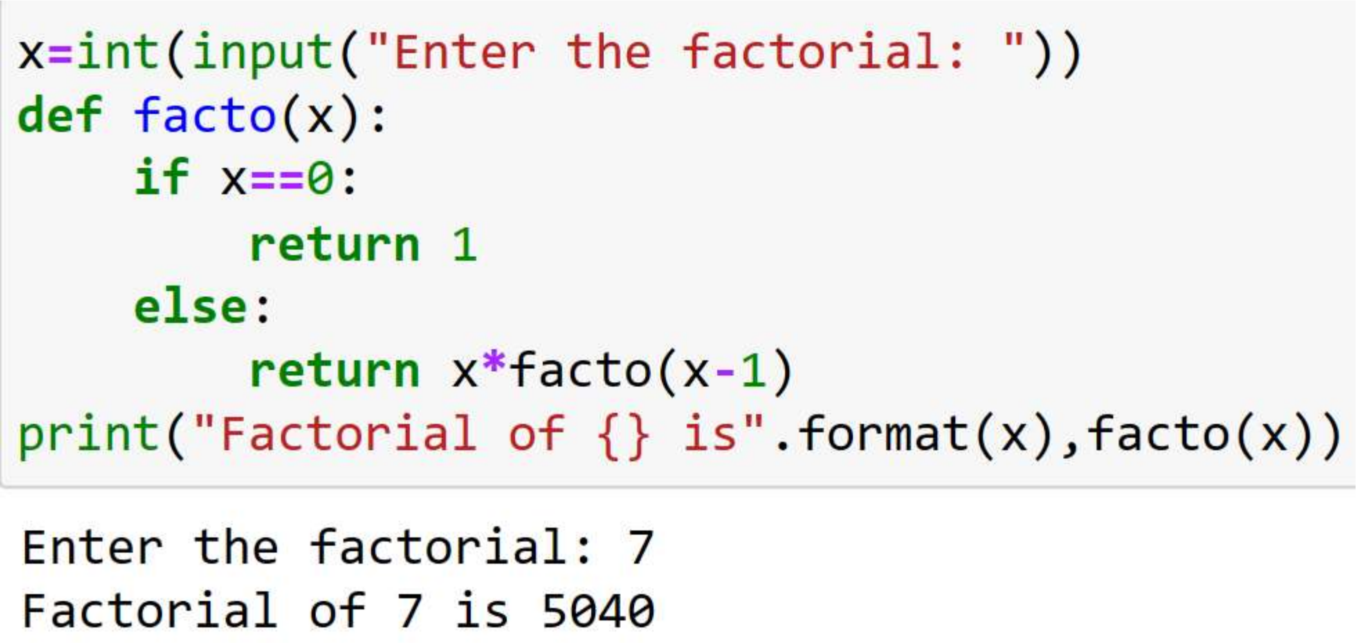
```
15
```

2. 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-

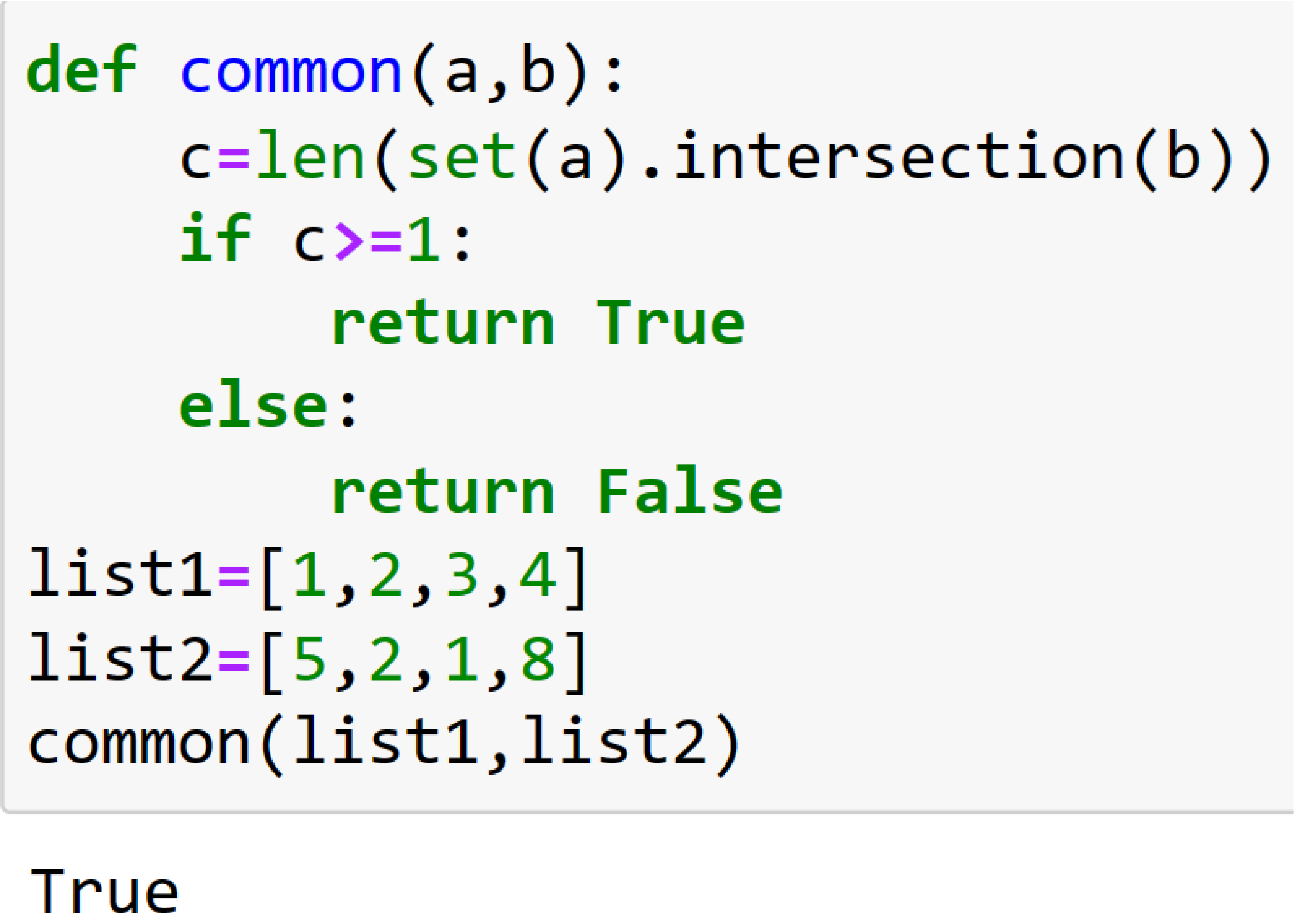


```
x=int(input("Enter the factorial: "))
def facto(x):
    if x==0:
        return 1
    else:
        return x*facto(x-1)
print("Factorial of {} is".format(x),facto(x))
```

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

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

Ans:



```
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
```

4. 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.

5. Define inheritance and briefly about different types of inheritances with at least two 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

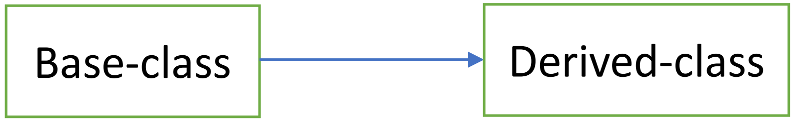
-Multiply inheritance

-Hierarchical inheritance

➢ SINGLE INHERITANCE:-

- In single inheritance a derived class

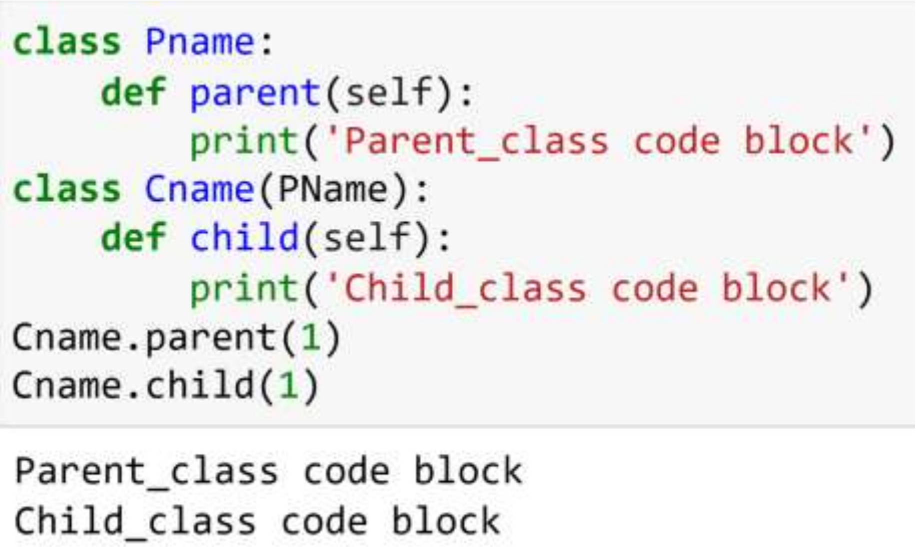
  is created using a single-based class.



#Parent class

#Child class

#single inheritance code



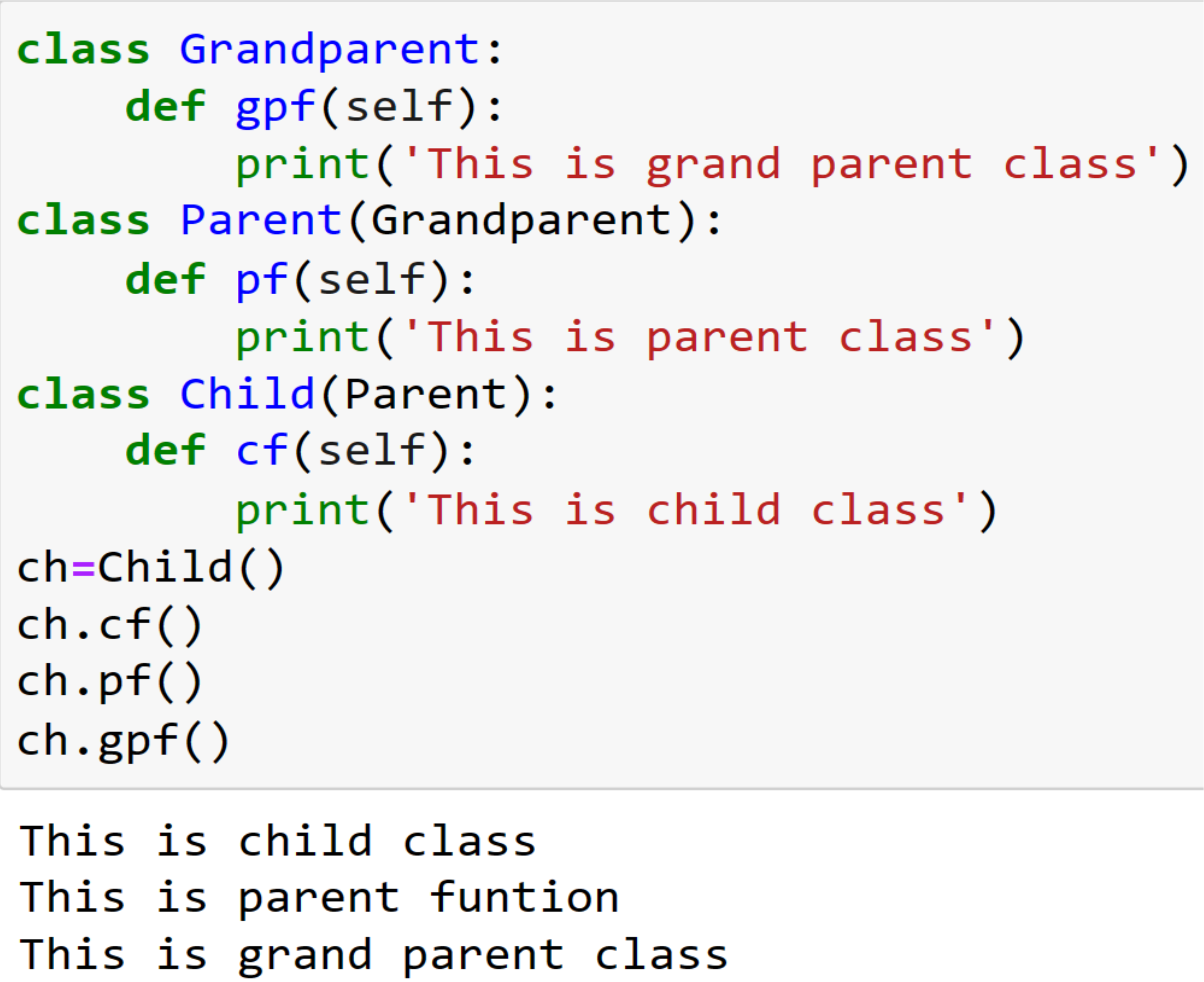
```
class Pname:
    def parent(self):
        print('Parent_class code block')
class Cname(PName):
    def child(self):
        print('Child_class code block')
Cname.parent(1)
Cname.child(1)
```

```
Parent_class code block
Child_class code block
```

➢ MULTI LEVEL INHERITANCE:-

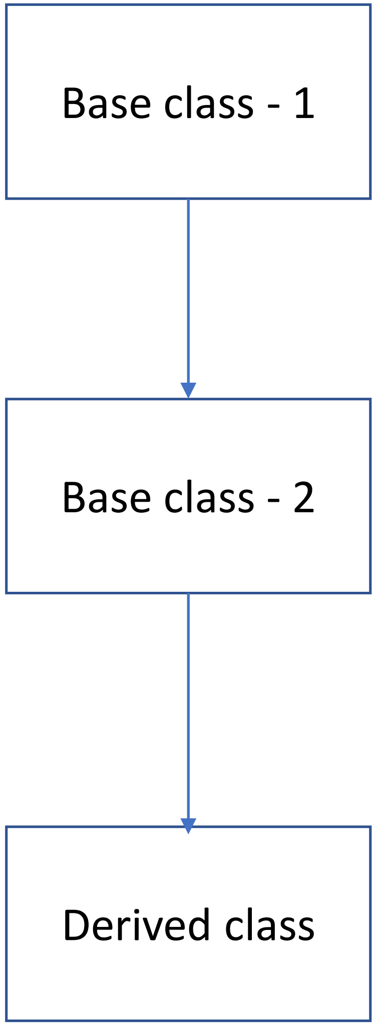
- In this every new class is derived from a single-parent class.

#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
```



#Grand parent class

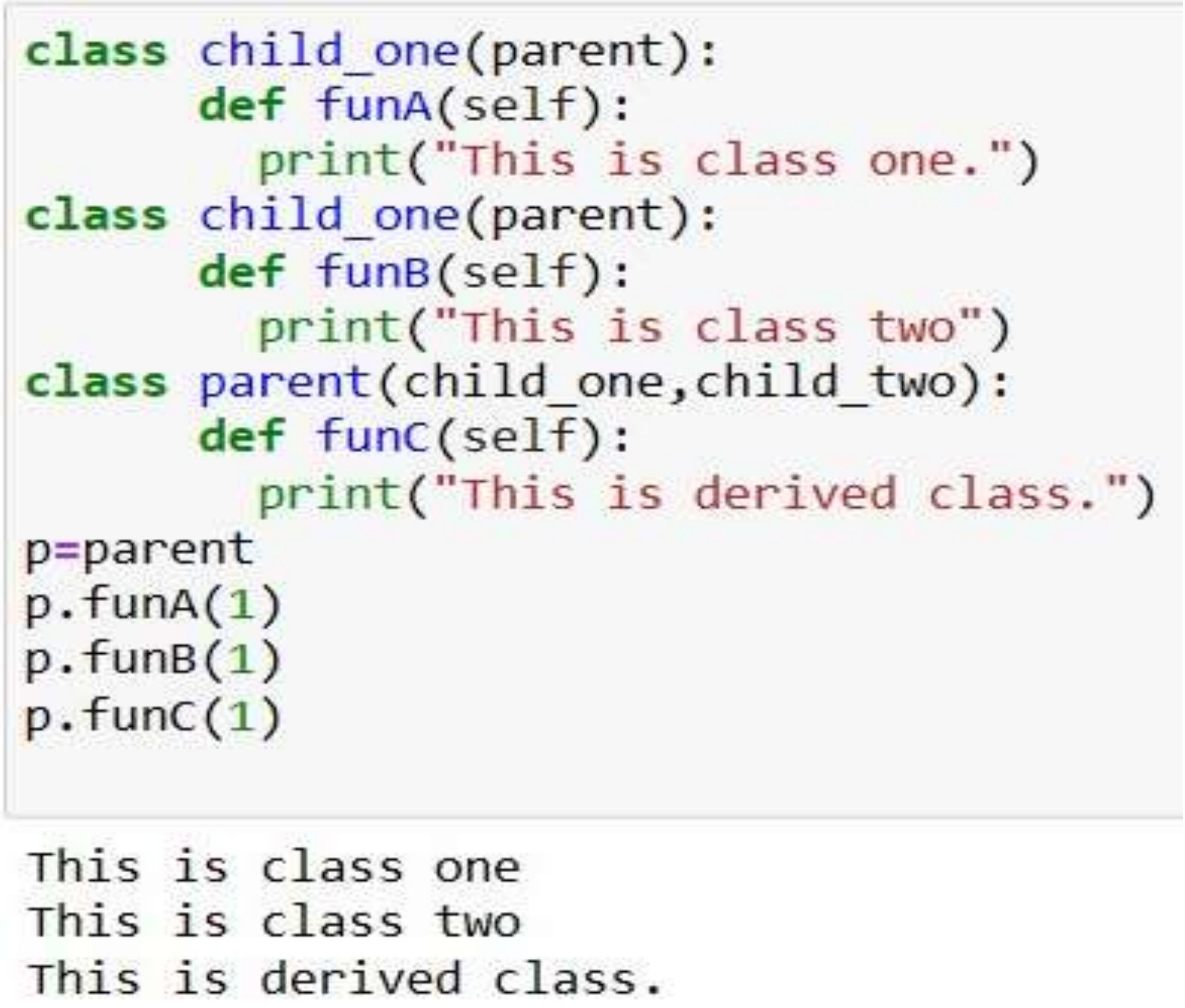
#Parent class

#Child class

➢ MULTIPLE-LEVEL INHERITANCE:-

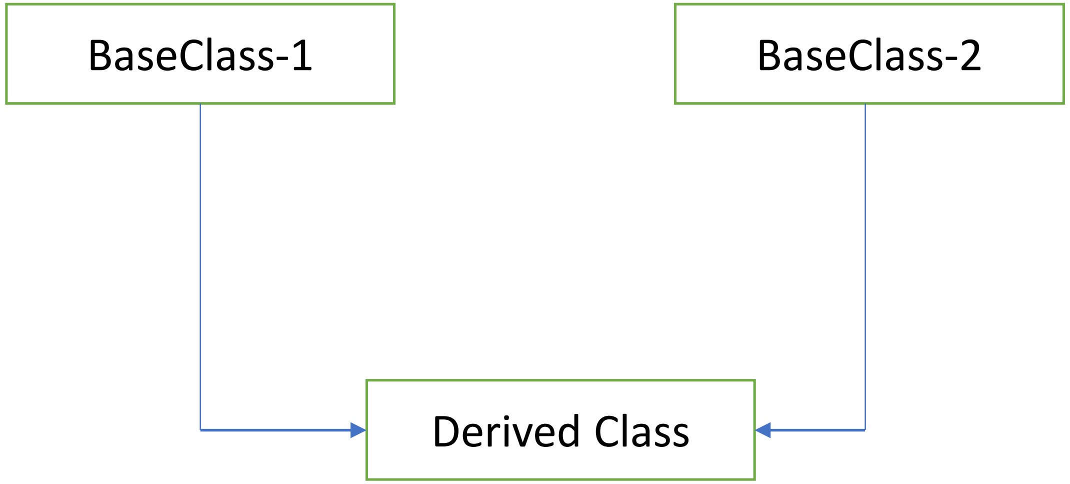
- In Multiple inheritances a derived class is created by using two or more base classes.

#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)
```

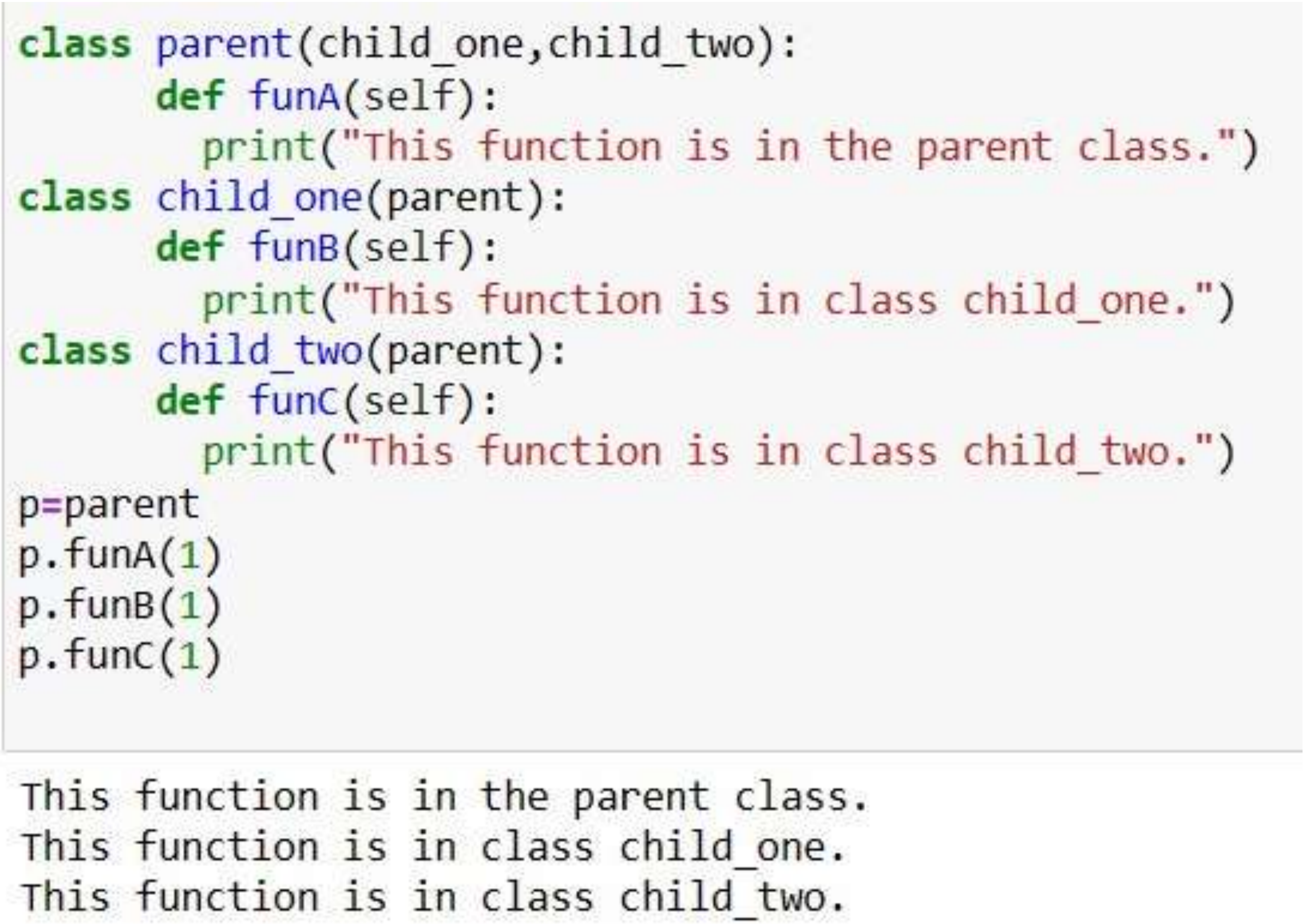
```
This is class one
This is class two
This is derived class.
```



➢ HIERARCHICAL INHERITANCE:-

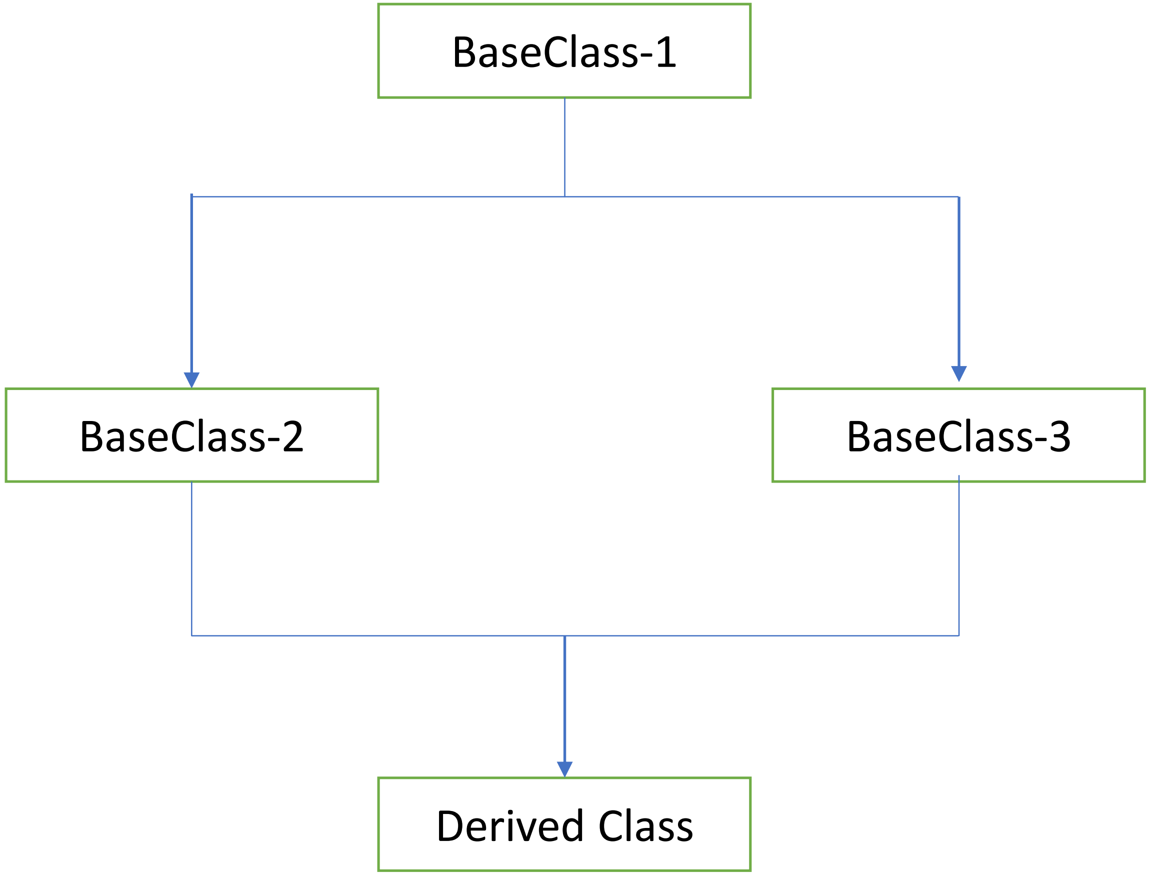
- Hierarchical inheritance is the process of creating a child class from a single-parent class and also multiple-base classes.

#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.
```



BaseClass-1

BaseClass-2

BaseClass-3

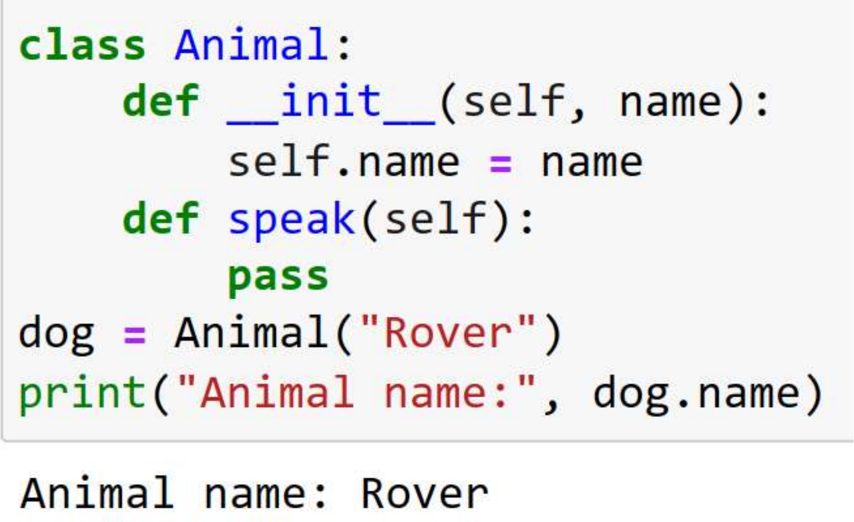
Derived Class

6. What is the Constructor and method overriding explain with an example program.

Ans:

- **Constructor**: A constructor is a special method in Python classes that is automatically called when an object of the class is created. The purpose of the constructor is to initialize the object's attributes. In Python, the constructor is defined using the __init__ method.

Example for constructor:-

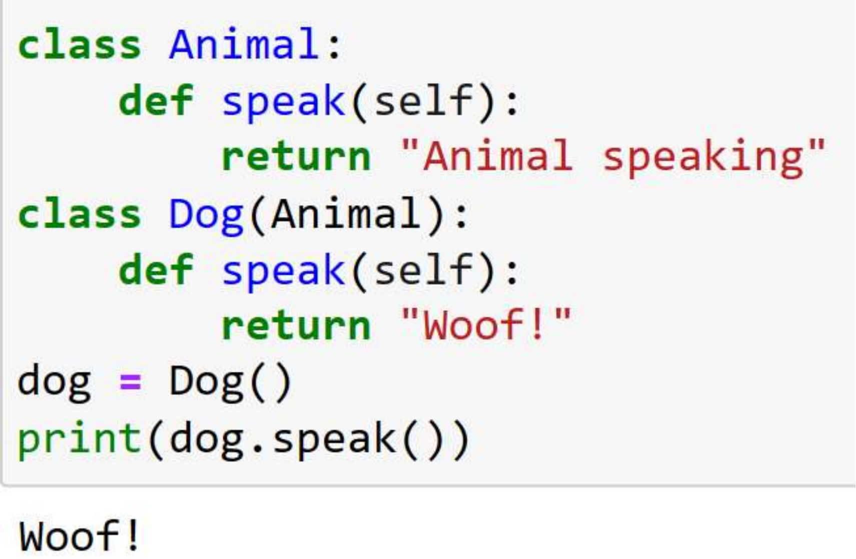


```python
class Animal:
    def __init__(self, name):
        self.name = name
    def speak(self):
        pass
dog = Animal("Rover")
print("Animal name:", dog.name)
```

```
Animal name: Rover
```

- **Method Overriding:** Method Overriding is a feature in object-oriented programming that allows a subclass to provide a different implementation of a method that is already defined in its parent class. The method in the subclass has the same name, parameters, and return type as the method in the parent class.

Example for overriding:-



```python
class Animal:
    def speak(self):
        return "Animal speaking"
class Dog(Animal):
    def speak(self):
        return "Woof!"
dog = Dog()
print(dog.speak())
```

```
Woof!
```

UNIT-5

1. **Define Exception? List any 6 types of Exceptions.**

Ans:

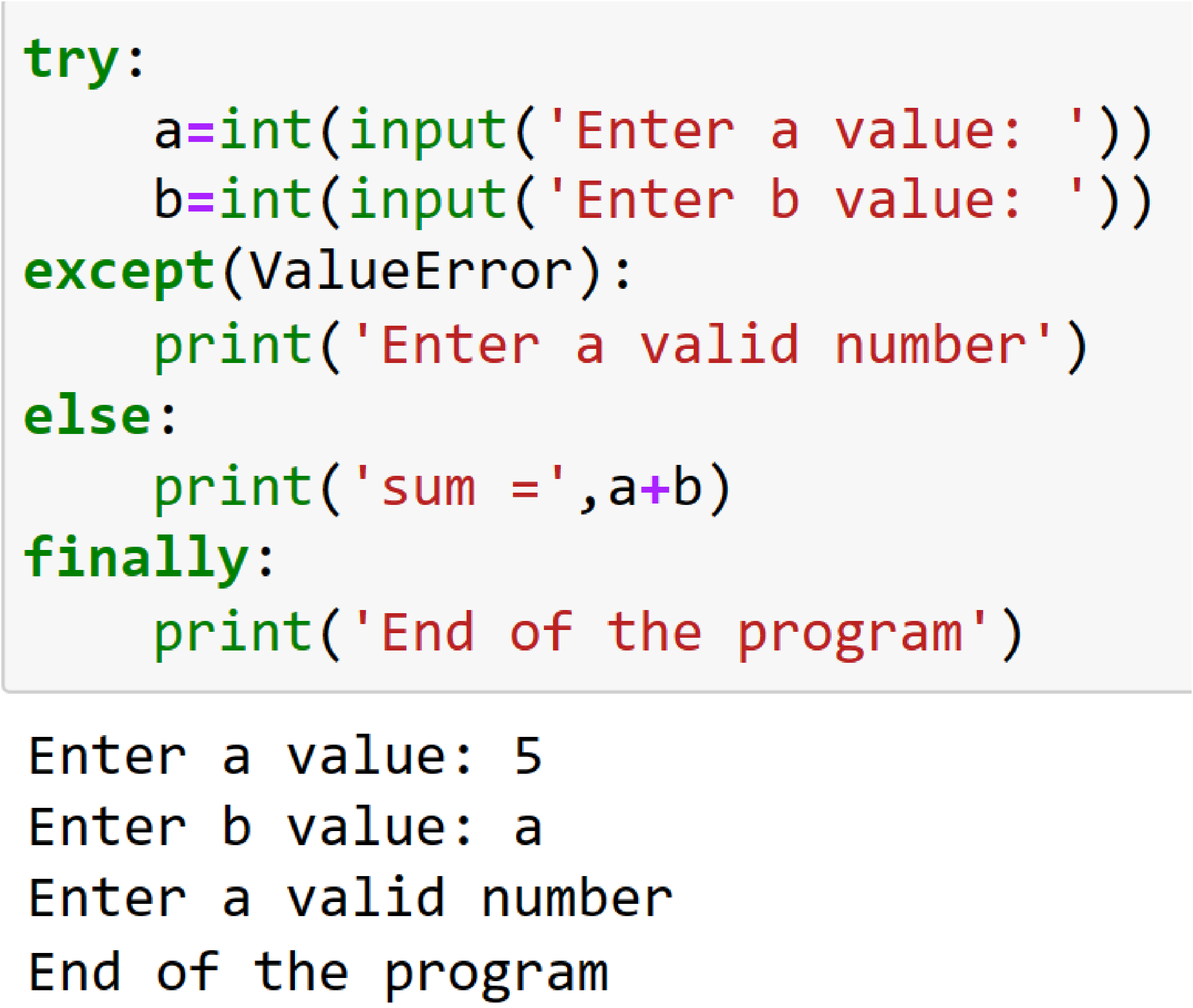
- An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instruction. In general, when a python script encounters a situation that it cannot cope with, it raises an exception. An exception is a python object that represents an error.

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.

2. Write the syntax and program to handle executions?

Ans:



```
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')
```

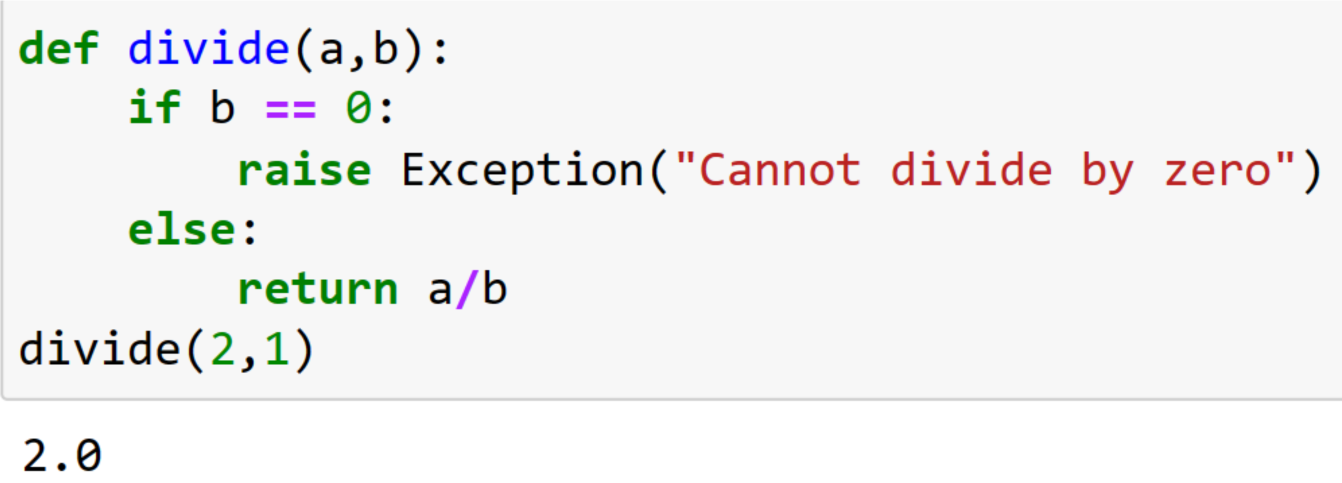
```
Enter a value: 5
Enter b value: a
Enter a valid number
End of the program
```

3. Explain about Raise an Exception?

Ans:

- The raise statement is used to raise an exception in Python. It is used to interrupt the normal flow of the program and transfer control to the exception-handling code. The raise statement allows you to throw a custom exception, and the type of the exception can be specified as an argument to the raise statement.

#code-



```
def divide(a,b):
    if b == 0:
        raise Exception("Cannot divide by zero")
    else:
        return a/b
divide(2,1)
```

```
2.0
```

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

Ans:

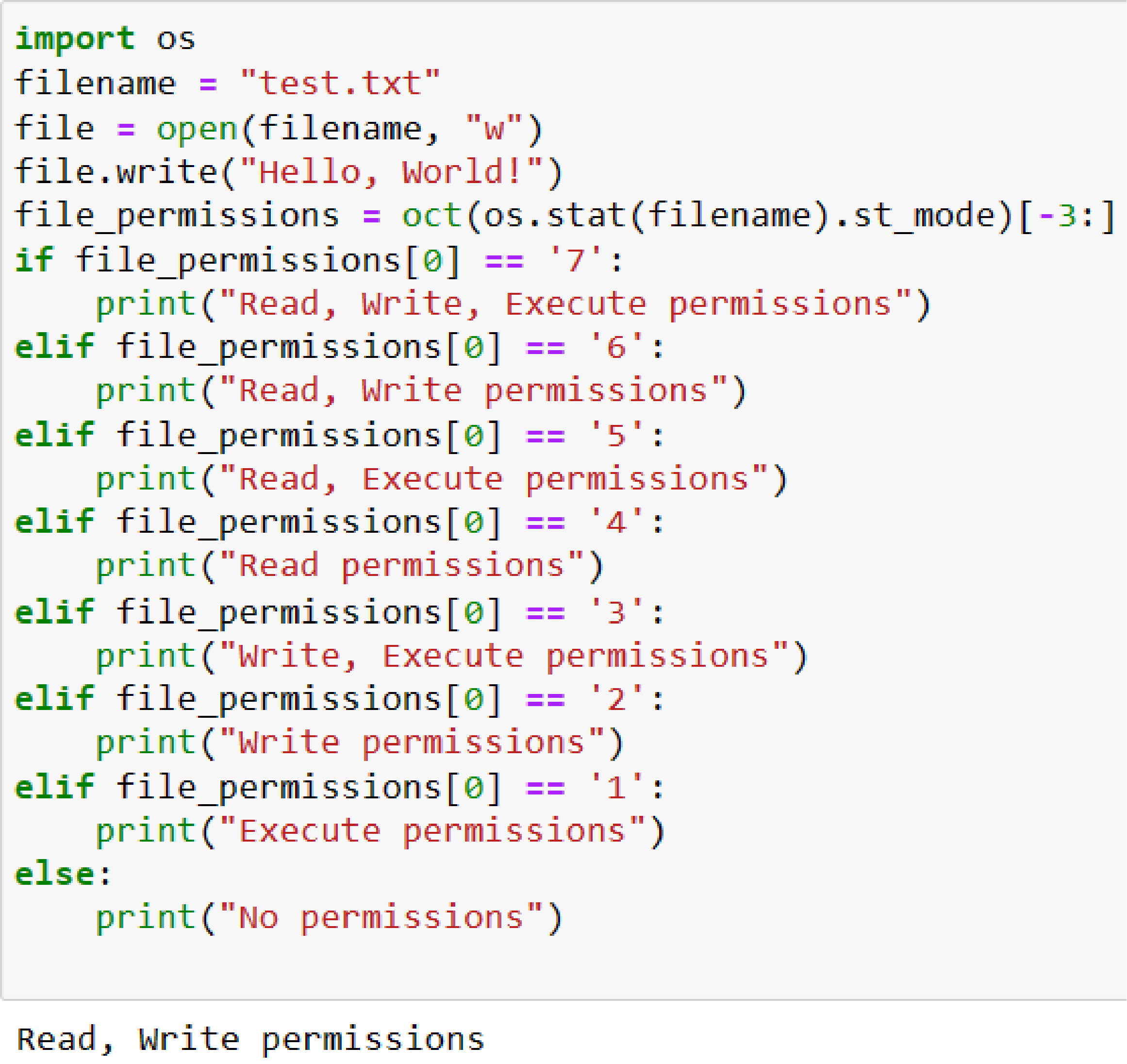
File: A file is a collection of data stored on a computer's storage device with a unique name and location. In Python, a file can be opened using the open function and can be read, written, or appended to, depending on the mode in which it was opened.

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.

5. **Write a python program to open a file and check what are the access permissions acquired by that file using OS module?**

Ans:



```python
import os
filename = "test.txt"
file = open(filename, "w")
file.write("Hello, World!")
file_permissions = oct(os.stat(filename).st_mode)[-3:]
if file_permissions[0] == '7':
    print("Read, Write, Execute permissions")
elif file_permissions[0] == '6':
    print("Read, Write permissions")
elif file_permissions[0] == '5':
    print("Read, Execute permissions")
elif file_permissions[0] == '4':
    print("Read permissions")
elif file_permissions[0] == '3':
    print("Write, Execute permissions")
elif file_permissions[0] == '2':
    print("Write permissions")
elif file_permissions[0] == '1':
    print("Execute permissions")
else:
    print("No permissions")
```

```
Read, Write permissions
```