K. Praveen Kumar

Assignment -6 (18-12-2023)

**OOPS in Python**:  
**1. Class:** A class is a collection of objects and class contains the blueprints or the prototype from which the objects are being created.

It is a legal entity that contains some attributes and methods.

Syntax:

Class classname:

#statements

class Dog():

    pass

**2.Objects:** It is an entity that has a state and behaviour associated with it.

> It is an instance of class.

**Creating an object:**

**obj=Dog()**

class Dog():

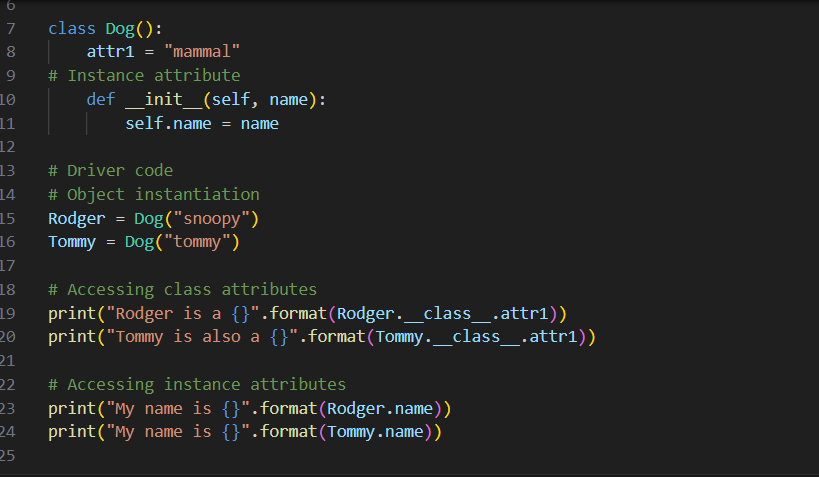
    pass

obj=Dog()

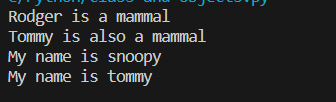
**\_\_init\_\_method:**

It is similar to constructors in c++ and java. It runs as soon as an object of a class is instantiated. This method is useful to do any initialization you want to do with your object.

Example:



Output:



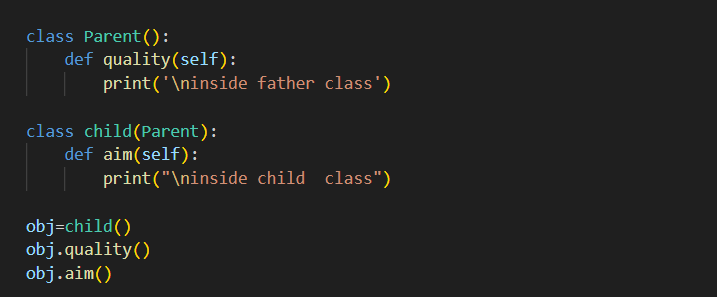
**3. Inheritance:**

Inheritance has the capability of acquiring the properties from another class.

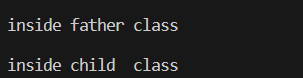
The class that derives properties is called derived class and the class from which the properties are being derived is called parent class or base class.

**Types:**  
**1. Single Inheritance:** In this it enables a derived class to inherit properties from a single-parent class.

Example:

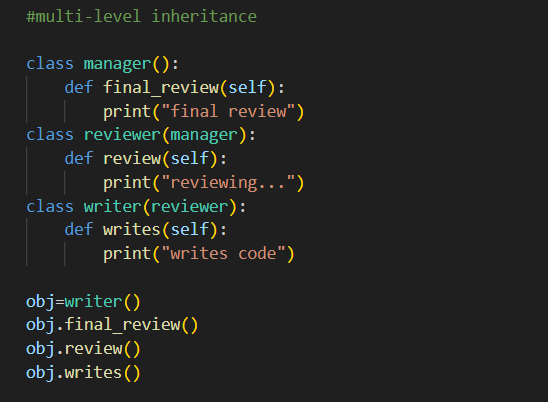


Output:



**2.Multi-level Inheritance:** In this there will be one base class and then the derived class and this derived class will be acting as a base class for the final derived class.

Example:



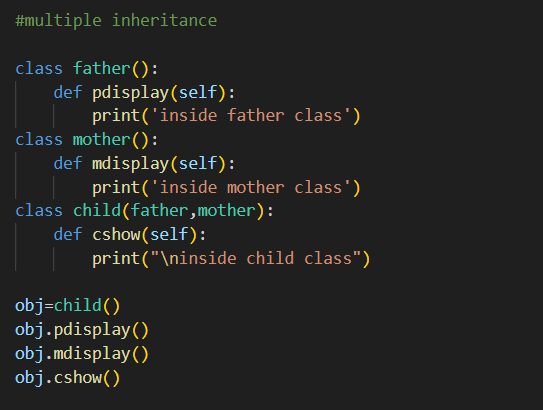
Output:



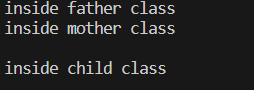
**3.Multiple Inheritance:** In this we cannot inherit the features of more classes into single class.

Suppose we are having two parent classes (father and mother) and one child class is derived from two parent classes.

Example:



Output:



**4.Heirarchial Inheritance:** In this, multiple classes can be derived from a single base class.

Example:

#heirarchial inheritance

class parent():

    def show(self):

        print("\ninside parent class")

class child1(parent):

    def c1display(self):

        print("inside child1 class")

class child2(parent):

    def c2dis(self):

        print("inside child2 class")

obj1=child1()

obj1.show()

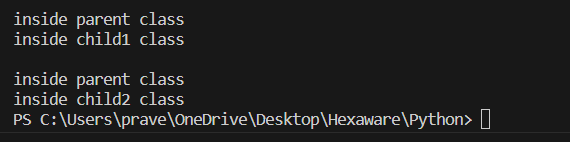
obj1.c1display()

obj2=child2()

obj2.show()

obj2.c2dis()

Output:



**4. Polymorphism:** Implementing the same thing in different ways.

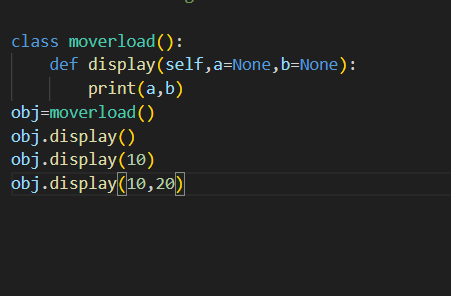
The polymorphism explains the concept of Method Overloading and Method Overriding.

1.Method Overloading:

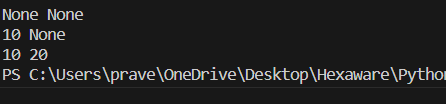
-> In this the method name should be same.

-> Arguments must be different.

Example:



Output:



2.Method Overriding:

-> In this the method name and also the number of arguments should be the same.

class bird():

    def intro(self):

        print("type of birds:")

    def flight(self):

        print("all birds cannot fly")

class sparrow(bird):

    def flight(self):

        print("sparrows can fly")

class ostrich(bird):

    def flight(self):

        print('ostrich cannot fly')

obj1=bird()

obj2=sparrow()

obj3=ostrich()

obj1.intro()

obj1.flight()

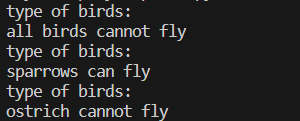
obj2.intro()

obj2.flight()

obj3.intro()

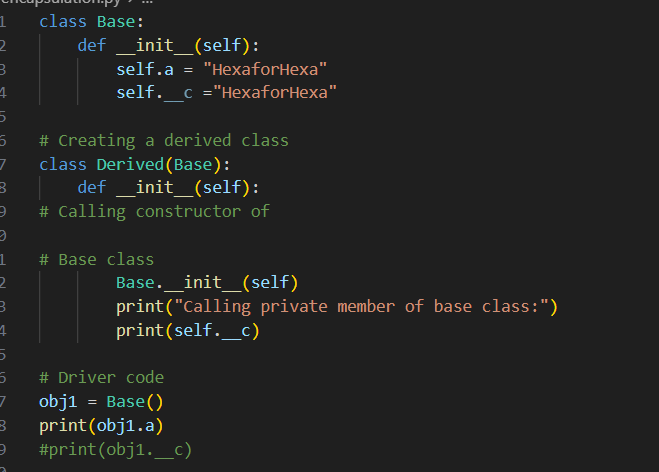
obj3.flight()

Output:

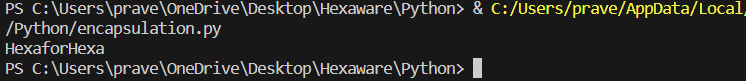


**5. Encapsulation:** It is defined as idea of wrapping of data and the methods that work on data within one unit. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data.

Example:



Output:



**6.Abstraction:** Data abstraction means it hides complex implementation details and expose only essential information and functionalities to the users.

In python, we can achieve data abstraction by using abstract classes and those classes can be created by using abc module and abstract method of abc module.

**File Handling in Python:**

It allows users to handle files that is to read and write files, along with many other file handling options to operate on files.

Advantages:

1.Versatility

2.Flexibility

3.User-friendly

4.Cross-platform

Disadvantages:

1.Error-prone

2.Security risks

3.Complexity

4.Performance

**Open() Function:**

**Syntax:** f=open(‘filename’,mode)

Opening a file in read mode:

file=open('file handling.txt','r')

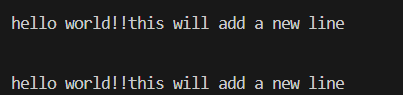
print(file.read())

**or**

f=open('file handling.txt','r')

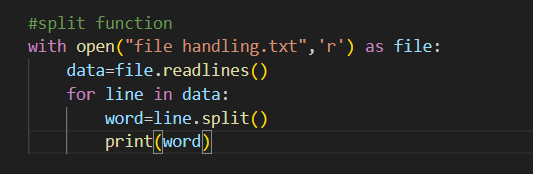
for each in f:

    print(each)

**output:**

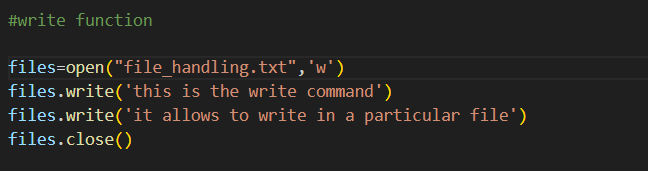
**Split function():**

By using this we can split the lines while reading the python files.

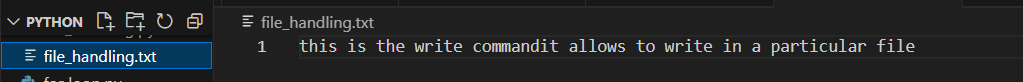


Output:



**Creating a file using write() function:**

**Output:**

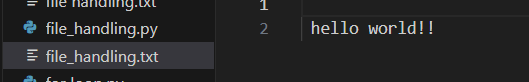


We can also use the written statement along with() function:

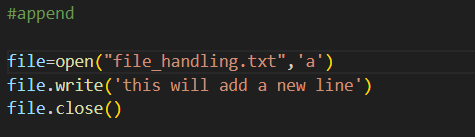
with open("file\_handling.txt",'w') as f:

    f.write("\n hello world!!")

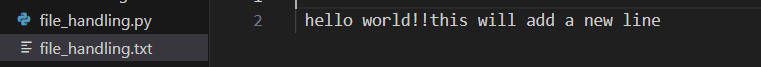
output:



**Append:**

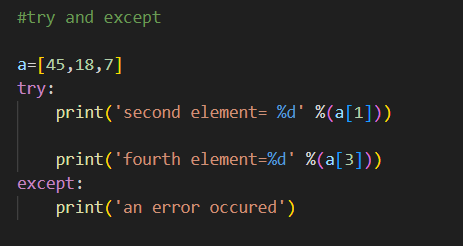
****

**Output:**

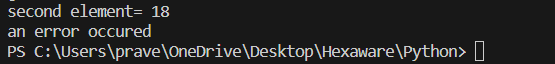
****

**Try and Except statements:**

Used to catch and handle exceptions in python.



Output:



**Python Modules:** A module is a file containing python definitions and statements. It can define functions, classes and Variables.

Creating of python module:

Let us create a module for simple operations on variables and name the file as calc.py

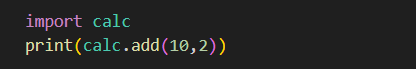
def add(x,y):

    return (x+y)

def subtract(x,y):

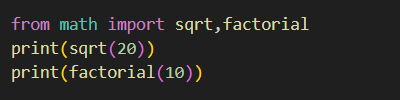
    return (x-y)

we can import calc module and perform operations like:



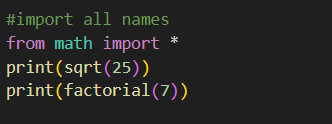
Output:

12



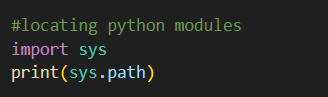
Output:  


**Import all names:** \* symbol used with import to import all the names from a module to a current namespace.

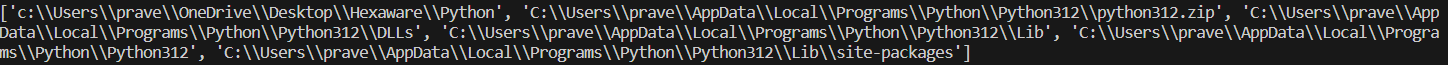


Output:



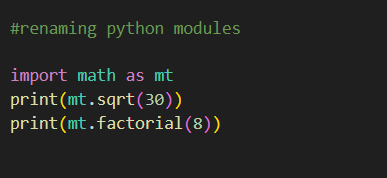
Locating python modules:  


Output:



**Renaming a Python module:**

**Syntax:** import module\_name as alias\_name





**Built-in modules:**

#built-in modules

import math

print(math.sqrt(15))

print(math.pi)

print(math.degrees(45))

print(math.radians(50))

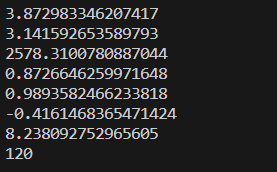
print(math.sin(8))

print(math.cos(2))

print(math.tan(1.45))

print(math.factorial(5))

**Output:**

****

import random

print('\n',random.randint(0,100))

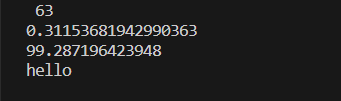
print(random.random())

print(random.random()\*100)

l=[1,4,5,"hello","world",False]

print(random.choice(l))

output:



#importing built in datetime '

import datetime

from datetime import date

import time

print('\n',time.time())

print(date.fromtimestamp(652252))

output:



Creating our module:

We have created a module which is used to convert the temperature to centigrade and Fahrenheit and name it as ‘tempcoversion’.

def to\_centigrade(x):

    return 5\*(x-32)/9.0

def to\_fahrenheit(x):

    return 9\*x/0.5+32

#water freezing temp in celsius

freezing\_c=0.0

#water freezing temp in fahrenheit

freezing\_f=32.0

we can use typecoversion module to convert the temperature.

import tempcoversion

print(tempcoversion.to\_centigrade(15))

print(tempcoversion.freezing\_f)

output:



#importing particular components of module

from tempcoversion import to\_fahrenheit

print(to\_fahrenheit(30))

from tempcoversion import freezing\_c

print(freezing\_c)

output:

