Clasess & Objects

In [ ]:
```
class demo :
    pass
```

In [ ]:
```
obj = demo()
```

In [ ]:
```
print(obj)
```

In [ ]:
```
help(obj)
```

Attributes & Methods

In [ ]:
```
class Health :
    '''This is class "Health" with information and functions related with our health. '''
    # __init__ function
    def __init__(self, name, age, loc):
        '''
        Initialization of Health class.
        '''
        self.name = name
        self.age = age
        self.location = loc

    # another method to display data
    def print_info(self):
        print('Individual Helath Information :')
        print('Name     : ', self.name)
        print('Age      : ', self.age)
        print('Location : ', self.location)
```

In [ ]:
```
h1 = Health('John Davis', 54, 'Texas')
```

In [ ]:
```
h1.print_info()
```

Data attributes & class attributes

In [ ]:
```
class Health :
    '''This is class "Health" with information and functions related with our health. '''
    # class attribute
    counter = 0

    # __init__ function
    def __init__(self, name, age, loc):
        '''
        Initialization of Health class.
        '''
        self.name = name
        self.age = age
        self.location = loc
        Health.counter += 1

    # another method to display data
    def print_info(self):
        print('Individual Information :')
        print('Name     : ', self.name)
        print('Age      : ', self.age)
        print('Location : ', self.location)
        print('No. of Objects : ', Health.counter)
```

In [ ]:
```
obj_1 = Health('John Davis', 54, 'Texas')
```

In [ ]:
```
obj_1.print_info()
```

In [ ]:
```
obj_2 = Health('Dave Jones', 31, 'NewYork')
```

In [ ]:
```
obj_2.print_info()
```

Access Data & Class Attributes

In [ ]:
```
obj_1.name, obj_1.age
```

In [ ]:
```
obj_2.name, obj_2.age
```

In [ ]:
```
obj_1.counter , obj_2.counter
```

Data Attributes vs Class Attributes

In [ ]:
```
obj_1.name is obj_2.name
```

In [ ]:
```
obj_1.counter is obj_2.counter
```

In [ ]:
```
```

Access Modifiers

In [ ]:
```
class student :
    counter = 0
    # constructor
    def __init__(self, name, fathersname, age):
        self.name = name
        self.fathersname = fathersname
        self.age = age
        student.counter +=1
        self._roll_no = 'A2022/{}/{:003d}'.format(self.name[:3].upper() ,student.counter)
        self.__file_no = 'A2022/{}/{:003d}'.format(self.name.upper() ,student.counter)
    def print_info(self):
        print('Name          : ', self.name)
        print("Father's Name : ",self.fathersname )
        print("Age           : ", self.age)
        print("Roll No.      : ", self._roll_no)
        print("File No.      : ", self.__file_no)
```

In [ ]:
```
st1 = student('John Davis', 'Malcom Davis', 10)
```

In [ ]:
```
st1.print_info()
```

In [ ]:
```
class science(student):
    def display(self):
        print('Name          : ', self.name)
        print("Father's Name : ",self.fathersname )
        print("Age           : ", self.age)
        print("Roll No.      : ", self._roll_no)
        print("File No.      : ", self.__file_no)
```

In [ ]:
```
sci_st = science('John Davis', 'Malcom Davis', 15)
```

In [ ]:
```
sci_st.print_info()
```

In [ ]:
```
sci_st.display()
```

In [ ]:
```
```

In [ ]:
```
```

Inheritance

In [ ]:
```
class Parent :
    def __init__(self):
        print('Welcome to Parent Class')

    def parent_func(self):
        print('This is the parent Function')
```

In [ ]:
```
class Child(Parent):
    pass
```

In [ ]:
```
obj = Child()
```

In [ ]:
```
obj.parent_func()
```

Types of Inheritance

Single Inheritance

In [ ]:
```
# base class
class information:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def print_info(self):
        print('Name   : ', self.name)
        print('Age    : ', self.age)
        print('Gender : ', self.gender)
```

In [ ]:
```
# derived class
class learners(information):
    def set_learner_data(self, exp = None, qual = None):
        self.qual = qual
        self.exp = exp

    def display(self):
        self.print_info()
        print('Qual   : ', self.qual)
        print('Exp    : ', self.exp)
```

In [ ]:
```
obj1 = learners('John Davis', 34, 'M')
```

In [ ]:
```
obj1.print_info()
```

In [ ]:
```
obj1.set_learner_data(5, 'Graduate')
```

In [ ]:
```
obj1.display()
```

Multilevel Inheritance

In [ ]:
```
# base class
class information:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def print_info(self):
        print('Name   : ', self.name)
        print('Age    : ', self.age)
        print('Gender : ', self.gender)
```

In [ ]:
```
# derived class
class learners(information):
    def set_learner_data(self, exp = None, qual = None):
        self.qual = qual
        self.exp = exp

    def display(self):
        self.print_info()
        print('Qual   : ', self.qual)
        print('Exp    : ', self.exp)
```

In [ ]:
```
class profile(learners) :
    pass
```

In [ ]:
```
obj_2 = profile('John Davis', 34, 'M')
```

In [ ]:
```
obj_2.print_info()
```

In [ ]:
```
obj_2.set_learner_data(3, 'Graduate')
```

In [ ]:
```
obj_2.display()
```

Hierarchical Inheritance

In [ ]:
```
# base class
class information:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def print_info(self):
        print('Name   : ', self.name)
        print('Age    : ', self.age)
        print('Gender : ', self.gender)
```

In [ ]:
```
# derived class 1
class learners(information):
    def set_learner_data(self, exp = None, qual = None):
        self.qual = qual
        self.exp = exp

    def display(self):
        self.print_info()
        print('Qual   : ', self.qual)
        print('Exp    : ', self.exp)
```

In [ ]:
```
# derived class 2
class trainer(information):
    def set_trainer_data(self, exp, charges ):
        self.exp = exp
        self.charges = charges

    def display_trainer(self):
        self.print_info()
        print('Experience        : ', self.exp)
        print('Hourly Charges    : ', self.charges)
```

In [ ]:
```
obj_learner = learners('John Davis', 21, 'M')
obj_learner.set_learner_data(3, 'Graduate')
```

In [ ]:
```
obj_learner.print_info()
```

In [ ]:
```
obj_learner.display()
```

In [ ]:
```
obj_trainer = trainer('Michael Dave', 43, 'Doctrate')
obj_trainer.set_trainer_data(15, 3500)
```

In [ ]:
```
obj_trainer.print_info()
```

In [ ]:
```
obj_trainer.display_trainer()
```

Multiple Inheritance

In [ ]:
```
# base class 1
class information:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def print_info(self):
        print('Name   : ', self.name)
        print('Age    : ', self.age)
        print('Gender : ', self.gender)
```

In [ ]:
```
# base class 2
class course_inf:
    def __init__(self):
        self.inf = {'Weekend' : {'Big Data & AI':12,
                                 'Cloud Computing' : 8,
                                 'Data Science with Python' :6,
                                 'Data Science with R' : 5 },
                    'Weekday' : {'Big Data & AI':6,
                                 'Cloud Computing' : 5,
                                 'Data Science with Python': 3,
                                 'Data Science with R' : 2.5 }}
```

In [ ]:
```
# derived class
class learners(information, course_inf):

    def __init__(self,name, age, gender,course, pref):
        information.__init__(self, name, age, gender)
        course_inf.__init__(self)
        self.course = course
        self.pref = pref

    def display(self):
        information.print_info(self)
        print('Course : ', self.course)
        print('Pref   : ', self.pref)
        weeks = self.inf[self.pref][self.course]
        print('# of Weeks : ', weeks)
```

In [ ]:
```
obj_learn = learners('Ava Jones', 33, 'F', 'Data Science with R', 'Weekend')

obj_learn.display()
```

In [ ]:
```
obj_learn.inf
```

Polymorphism

Method Overloading

In [ ]:
```
class information:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def print_info(self):
        print('Name   : ', self.name)
        print('Age    : ', self.age)
        print('Gender : ', self.gender)
```

In [ ]:
```
class learners:
    def __init__(self, name, age, gender, exp, qual):
        self.name = name
        self.age = age
        self.gender = gender
        self.qual = qual
        self.exp = exp

    def print_info(self):
        print('Name\t: ', self.name)
        print('Age\t: ', self.age)
        print('Gender\t: ', self.gender)
        print('Qual   : ', self.qual)
        print('Exp    : ', self.exp)
```

In [ ]:
```
obj1 = information('John Davis', 34, 'M' )
obj2 = learners('John Davis', 34, 'M', 3.5, 'Graduate')
```

In [ ]:
```
obj1.print_info()
```

In [ ]:
```
obj2.print_info()
```

Operator Overloading

In [ ]:
```
x = 8; y = 15
x + y
```

In [ ]:
```
x = 'FirstName'
y = 'LastName'
x + ' ' + y
```

Method Overriding

In [ ]:
```
class information:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def print_info(self):
        print('Name   : ', self.name)
        print('Age    : ', self.age)
        print('Gender : ', self.gender)
```

In [ ]:
```
class learners(information):
    def __init__(self, name, age, gender, exp, qual):
        information.__init__(self, name, age, gender)
        self.qual = qual
        self.exp = exp

    def print_info(self):
        '''print_info function of derived class overrides the print_info function of parent class'''
        print('Qual   : ', self.qual)
        print('Exp    : ', self.exp)
```

In [ ]:
```
obj = learners('John Davis', 45, 'M', 4.5, 'Graduate')
```

In [ ]:
```
obj.print_info()
```

In [ ]:
```
class learners(information):
    def __init__(self, name, age, gender, exp, qual):
        information.__init__(self, name, age, gender)
        self.qual = qual
        self.exp = exp

    def print_info(self):
        '''
        print_info function of derived class overrides the print_info function of parent class.
        To use the print ifno function of parent class, the function needs to be invoked explicitly.
        '''
        information.print_info(self)
        print('Qual   : ', self.qual)
        print('Exp    : ', self.exp)
```

In [ ]:
```
obj = learners('John Davis', 45, 'M', 4.5, 'Graduate')
```

In [ ]:
```
obj.print_info()
```

In [ ]:
```
```

In [ ]:
```
```

Abstraction

In [1]:
```
from abc import abstractmethod, ABC
```

In [2]:
```
class beverage(ABC):
    @abstractmethod
    def ingredients(self):
        print('base')
    def taste(self):
        pass
```

In [3]:
```
obj = beverage()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-570ae97a100b> in <module>
----> 1 obj = beverage()

TypeError: Can't instantiate abstract class beverage with abstract methods ingredients
```

In [4]:
```
# derived class 1
class mango_shake(beverage):
    def ingredients(self):
        print('Mango, Milk and Sugar')

    def taste(self):
        print('Yumm!!')
```

In [5]:
```
# derived class 1
class orange_juice(beverage):
    def ingredients(self):
        print('Orange, Water and Sugar')

    def taste(self):
        print('Sweet!!')
```

In [6]:
```
obj1 = mango_shake()
obj1.ingredients()
obj1.taste()
```

```
Mango, Milk and Sugar
Yumm!!
```

In [7]:
```
obj2 = orange_juice()
obj2.ingredients()
obj2.taste()
```

```
Orange, Water and Sugar
Sweet!!
```

In [ ]:
```
```