In [1]:
```
#Q1
```

In [2]:
```
#class is a classification of real world entity,class is a blue print.Class simplif
# An object is a single instance of a class. You can create many objects from the s
```

In [4]:
```
#EX
class pwskills:
    def welcome_msg(self):
        print("Welcome to pwskills")
pw=pwskills()
print(type(pw))
pw.welcome_msg()
```

```
<class '__main__.pwskills'>
Welcome to pwskills
```

In [5]:
```
#Q2
```

In [6]:
```
# 4 pillars of oops
# 1.Inheritance
# 2.Abstraction
# 3.Encapsulation
# 4.polymorphism
```

In [7]:
```
#Q3
```

In [9]:
```
#__init__ is also a constructor which is used to pass the data

#"__init__" is a reserved method in python classes.
#It is known as a constructor in OOP concepts. This method called when an object is
#Ex
class pwskills2:
    def __init__(self,phone_number,email_id,student_id):
        self.phone_number1=phone_number
        self.email_id1=email_id
        self.student_id1=student_id
pw=pwskills2(966695555,"sudh@gmail.com",102)
pw.phone_number1
```

Out[9]:
```
966695555
```

In [11]:
```
#Q4
#Self is a reference to current instance to the class.By using the self we can acce
#attributes and methods of the class in python.It binds the attributes with the giv
```

In [12]:
```
#Q5
#Inheritance means parent class property inherit by child class property.There are
#1.Single2.Multiple3.Multilevel4.hybrid5.Hierarchical
```

In [14]:
```
#Single
class test:
    def test_meth(self):
```

```
        return"this is my first class"
class test1(test):
    pass
test_obj=test1()
test_obj.test_meth()
```

Out[14]: 'this is my first class'

In [15]:
```
#multilevel
class Class1:
    def test_class1(self):
        return"this is a meth from class1"
class Class2(Class1):
    def test_class2(Class1):
        return"this is a meth from class2"
class Class3(Class2):
    pass
obj_class3=Class3()
obj_class3.test_class1()
```

Out[15]: 'this is a meth from class1'

In [17]:
```
#Multiple
class Class1:
    def test_class1(self):
        return"this is a class1"
class Class2:
    def test_class2(self):
        return"this is a class2"
class Class3(Class1,Class2):
    pass
obj_class3=Class3()
obj_class3.test_class1()
```

Out[17]: 'this is a class1'

In [18]:
```
# Hierarchical inheritance


# Base class
class Parent:
    def func1(self):
        print("This function is in parent class.")

# Derived class1


class Child1(Parent):
    def func2(self):
        print("This function is in child 1.")
        # Derivied class2


class Child2(Parent):
    def func3(self):
```

```
        print("This function is in child 2.")


# Driver's code
object1 = Child1()
object2 = Child2()
object1.func1()
object1.func2()
object2.func1()
object2.func3()
```

```
This function is in parent class.
This function is in child 1.
This function is in parent class.
This function is in child 2.
```

In [19]:

```
#Hybrid inheritance
class School:
    def func1(self):
        print("This function is in school.")


class Student1(School):
    def func2(self):
        print("This function is in student 1. ")


class Student2(School):
    def func3(self):
        print("This function is in student 2.")


class Student3(Student1, School):
    def func4(self):
        print("This function is in student 3.")


# Driver's code
object = Student3()
object.func1()
object.func2()
```

```
This function is in school.
This function is in student 1.
```

In [ ]: