```
class A:
    def __init__(self):
        self.P=10
        self.__Q=20
    def getY(self):
        return self.__Q
a=A()
print(a.P)
print(a.__Q)
```

```
class A:
  def __init__(self):
    self.P=10
    self.__Q=20
  def getY(self):
    return self.__Q
a=A()
print(a.P)
print(a.__Q)
```

```
10
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-2-92da7fc0feb6> in <cell line: 9>()
      7 a=A()
      8 print(a.P)
----> 9 print(a.__Q)

AttributeError: 'A' object has no attribute '__Q'
```

EXPLAIN ERROR

The above error is because the __Q variable is considered as a private variable, which cannot be accessed outside the class in this manner. The variable __Q can be accessed either through a.getY() method or a._A__Q

Note: technically there is no private variables in python. Usually the __variable is masked under the name for '_className__Variable' for avoiding clashes in inheritance chain, so the variables can still be accessed just not through the normal method that is "object.__variable"

```
a._A__Q
```

```
20
```

(b)

```
class A:
    def __init__(self,P):
        self.P=P
    def print(self):
        print(self.P)
a=A()
a.print()
```

```
class A:
  def __init__(self,P):
    self.P=P
  def print(self):
    print(self.P)
a=A()
a.print()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-429a2038677d> in <cell line: 6>()
      4   def print(self):
      5     print(self.P)
----> 6 a=A()
      7 a.print()

TypeError: A.__init__() missing 1 required positional argument: 'P'
```

EXPLAIN ERROR

while creating the object of class A, The constructor is called which expects an argument which is not provided.

The correct way to create an object of class A: a = A(5)

(c) Write down the output:

```
class A:
    def __init__(self,P):
        self.P=P
    def print(self):
        print(self.P)
a=A("Welcome")
a.print()
```

```
class A:
  def __init__(self,P):
    self.P=P
  def print(self):
    print(self.P)
a=A("Welcome")
a.print()
```

```
 Welcome
```

2. Write a program to create a class named Demo. Define two methods Get_String() and Print_String(). Accept the string from the user and print it in upper case.

```
class Demo:
    def __init__(self):
        self.string = ""

    def Get_String(self):
        self.string = input("Enter a string: ")
        return self.string

    def Print_String(self):
        print("String in uppercase:", self.string.upper())


d1 = Demo()
val = d1.Get_String()
print(val)
d1.Print_String()
```

```
 Enter a string: hello World
 hello World
 String in uppercase: HELLO WORLD
```

3. Create a class Circle. Use constructor to read an attribute radius and use proper functions to return the radius and calculate the area of circle.

```
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def get_radius(self):
        return self.radius

    def get_area(self):
        return 3.14*self.radius**2


c1 = Circle(5)
radius = c1.get_radius()
print("Radius of the circle:", radius)
area = c1.get_area()
print("Area of the circle:", area)
```

```
Radius of the circle: 5
Area of the circle: 78.5
```

4. Implement amrita grading system using inheritance.

```
class Student():
    def __init__(self, name, roll):
        self.name = name
        self.roll = roll

    def display_info(self):
        print("Name : ", self.name)
        print("Roll number : ", self.roll)

class AmritaStudent(Student):
    def __init__(self, name, roll, year):
        super().__init__(name, roll)
        self.year = year

    def display_info(self):
        super().display_info()
        print("Year:", self.year)

    def calculate_grade(self, marks):
        if marks >= 90:
            return 'A'
        elif marks >= 80:
            return 'B'
        elif marks >= 70:
            return 'C'
        elif marks >= 60:
            return 'D'
        else:
            return 'F'


s1 = AmritaStudent("fufu", 88, 2022)
s1.display_info()
print("Grade : ", s1.calculate_grade(95))
```

```
Name :  fufu
Roll number :  88
Year: 2022
Grade :  A
```

5. Implement hierarchical inheritance based on the following details:
Create class Shape. Derive two sub classes Rectangle and Triangle. Define the required attributes for each to calculate the area using constructors. Find the area of each using functions and objects.

```
class Shape:
    def __init__(self):
        pass

class Rectangle(Shape):
    def __init__(self, length, breadth):
        super().__init__()
        self.length = length
        self.breadth = breadth

    def calculate_area(self):
        return self.length * self.breadth

class Triangle(Shape):
    def __init__(self, base, height):
        super().__init__()
        self.base = base
        self.height = height

    def calculate_area(self):
        return 0.5 * self.base * self.height


rectangle = Rectangle(7, 5)
triangle = Triangle(10, 5)
print("Area of rectangle:", rectangle.calculate_area())
print("Area of triangle:", triangle.calculate_area())
```

```
 Area of rectangle: 35
 Area of triangle: 25.0
```

6. Try the programs in your tutorial PPT

```
class Employee1():#This is a parent class
    name="Dev"
    age=43
    def display(self):
        print("Function in Super class")

class childemployee(Employee1): #This is a child class
    def disp(self):
        print("My name is ", self.name, "age is ", self.age)

emp=childemployee()
emp. display()
emp.disp()
```

```
 Function in Super class
 My name is  Dev age is  43
```

```
# Base class1
class Mother:
    mothername = ""
    def mother(self):
        print(self.mothername)

# Base class2
class Father:
    fathername = ""
    def father(self):
        print(self.fathername)

# Derived class
class Son(Mother, Father):
    def parents(self):
        print("Father :", self.fathername)
        print("Mother :", self.mothername)

#Driver's code
s1 = Son()
s1.fathername = "RAM"
s1.mothername = "SITA"
s1. parents()
```

```
Father : RAM
Mother : SITA
```

```
class length:
    l = 0
    def length(self):
        return self.l

class breadth:
    b = 0
    def breadth(self):
        return self.b

class rect_area(length, breadth):
    def r_area(self):
        print("The area of rectangle with length " + str(self.l) + " units and breadth " + str(self.b) + " units is " + str(self.l * self.b) +

obj = rect_area()
obj.l = int(input("Enter the required length for rectangle: "))
obj.b = int(input("Enter the required breadth for rectangle: "))
obj.r_area()
```

```
Enter the required length for rectangle: 5
Enter the required breadth for rectangle: 2
The area of rectangle with length 5 units and breadth 2 units is 10 sq.units.
```

```
# Base class
class Shape:
    color="Yellow"
    def __init__(self,color):
        self.color=color
    # Derived class1
class Rect(Shape):
    def __init__(self,length,breadth):
        self.length=length
        self.breadth=breadth
        print("This function is in Rect.")
        print(self.color)
    def calc_area(self):
        area=self.length*self.breadth
        print("Area Rect.")
        print(area)
# Derivied class2
class Tri(Shape):
    def __init__(self,base,height):
        self.base=base
        self.height=height
        print("This function is in Triangle.")
        print(self.color)
        Shape.__init__(self,"Red")
        print(self.color)
    def calc_area(self):
        area=self.base*self.height*0.5
        print("Area Triangle.")
        print(area)
 # Driver's code
object1 = Rect(1,2)
object2 = Tri(1,2)
object1.calc_area()
object2.calc_area()
```

```
This function is in Rect.
Yellow
This function is in Triangle.
Yellow
Red
Area Rect.
2
Area Triangle.
1.0
```

```
class Parent:
    def __init__(self, name):
        self.name = name

    def getName(self):
        return self.name

class Child(Parent):
    def __init__(self, name, age):
        super().__init__(name)
        self.age = age

    def getAge(self):
        return self.age

class Grandchild(Child):
    def __init__(self, name, age, location):
        super().__init__(name, age)
        self.location = location

    def getLocation(self):
        return self.location
```

```
class Class1:
    def m(self):
        print("In Class1")
class Class2(Class1):
    def m(self):
        print("In Class2")
class Class3(Class1):
    def m(self):
        print("In Class3")
class Class4(Class2,Class3):
    pass

obj=Class4()
obj.m()
```

```
 In Class2
```

```
class  Class1:
    def m(self):
        print("In Class1")
        print("1")
class  Class2(Class1):
    def m(self):
        print("In Class2")
        print("2")
        Class1.m(self)
        print("3")
class  Class3(Class1):
    def m(self):
        print("In Class3")
        print("4")
        Class1.m(self)
```