super()

super() object is used to call or invoke members of super class within sub class.

Example:
```
class A:
    def __init__(self):
        self.x=100

class B(A):
    def __init__(self):
        super().__init__()
        self.y=200

def main():
    objb=B()
    print(objb.x)
    print(objb.y)
main()
```

Output:
```
======== RESTART: F:/python6pmaug/ooptest30.py =======
100
200
```

Example:
```
#Single Level Inheritance
class Person:
    def __init__(self):
        self.__name=None
    def setName(self,n):
        self.__name=n
    def getName(self):
        return self.__name
class Employee(Person):
    def __init__(self):
        super().__init__()
        self.__job=None
    def setJob(self,j):
        self.__job=j
    def getJob(self):
        return self.__job
def main():
    emp1=Employee()
    emp1.setName("Naresh")
    emp1.setJob("Manager")
    print(f'''Name {emp1.getName()}
Job {emp1.getJob()}''')

main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest31.py ========
Name Naresh
Job Manager
```

Example:

```
class A:
    def __init__(self):
        self.x=100 # public
        self._y=200 # protected
        self.__z=300 # private
class B(A):
    def __init__(self):
        super().__init__()
        self.p=400 # public


def main():
    objb=B()
    print(objb.x)
    print(objb.p)
    print(objb._y)
main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest32.py ========
100
400
```

```
200
```

What is difference between private,protected,public?

Private	**Protected**	**Public**
These members prefix with __	These members are prefix with _	This members are not prefix with any underscore
These members are accessible within class but cannot accessible within derived class or outside the class	These are accessible with class and derived class but not accessible outside derived class	These members can be accessible any where.

Example:

```
class A:
    def __init__(self):
        self.x=100
        self._y=200
        self.__z=300

class B(A):
    def __init__(self):
        super().__init__()
    def m1(self):
        print(f'public x={self.x}')
        print(f'protected y={self._y}')
        #print(f'private z={self.__z}')

def main():
    objb=B()
    objb.m1()
    print(objb.x)
main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest33.py ========
public x=100
protected y=200
100
```

Multilevel inheritance
More than one level of inheritance is called multilevel inheritance.
If class is derived from another derived class it is called multilevel inheritance.

```
class Person:
    def __init__(self,n):
        self.__name=n
    def getName(self):
        return self.__name

class Employee(Person):
    def __init__(self,n,j):
        super().__init__(n)
        self.__job=j
    def getJob(self):
        return self.__job

class SalariedEmployee(Employee):
    def __init__(self,n,j,s):
        super().__init__(n,j)
        self.__salary=s
    def getSalary(self):
        return self.__salary
def main():
    emp1=SalariedEmployee("naresh","manager",50000)
    print(f'''Name {emp1.getName()},Job {emp1.getJob()},
Salary {emp1.getSalary()}''')

main()
```

Output:
```
======== RESTART: F:/python6pmaug/ooptest34.py ========
Name naresh,Job manager,
Salary 50000
```

Multiple Inheritance
If a class derived from more than one base class is called multiple inheritance.

Example:

```
#multiple inheritance
class A:
    def __init__(self):
        self.x=100

class B:
    def __init__(self):
        self.y=200

class C(A,B):
    def __init__(self):
        super().__init__()
        B.__init__(self)
        self.z=300

def main():
    objc=C()
    print(f'''x={objc.x},y={objc.y},z={objc.z}''')
main()
```

Output:

```
x=100,y=200,z=300
```

Example:

```
# Creating a multiple inheritance using more than two classes.

class Car():
    def Benz(self):
        print(" This is a Benz Car ")
class Bike():
    def Bmw(self):
        print(" This is a BMW Bike ")
class Bus():
    def Volvo(self):
        print(" This is a Volvo Bus ")
class Truck():
    def Eicher(self):
        print(" This is a Eicher Truck ")
class Plane():
    def Indigo(self):
        print(" This is a Indigo plane ")
class Transport(Car,Bike,Bus,Truck,Plane):
    def Main(self):
        print("This is the Main Class")
B=Transport()
B.Benz()
B.Bmw()
B.Volvo()
B.Eicher()
B.Indigo()
B.Main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest36.py ========
 This is a Benz Car
 This is a BMW Bike
 This is a Volvo Bus
 This is a Eicher Truck
 This is a Indigo plane
This is the Main Class
```

Example:

```
class Person:
    def __init__(self):
        self.__name=None
    def setName(self,n):
        self.__name=n
    def getName(self):
        return self.__name

class Account:
    def __init__(self):
        self.__accno=None
    def setAccno(self,a):
        self.__accno=a
    def getAccno(self):
        return self.__accno

class SavingAccount(Person,Account):
    def __init__(self):
        Person.__init__(self)
        Account.__init__(self)
        self.__balance=None
    def deposit(self,t):
        if self.__balance==None:
            self.__balance=t
        else:
            self.__balance=self.balance+t
    def getBalance(self):
        return self.__balance

def main():
    acc1=SavingAccount()
    acc1.setName("naresh")
    acc1.setAccno(11111)
    acc1.deposit(50000)
    print(f'''AccountNo {acc1.getAccno()},
Name {acc1.getName()},
Balance {acc1.getBalance()}''')

main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest37.py ========
AccountNo 11111,
Name naresh,
Balance 50000
```

Method Overriding

6:00pm – 9:00pm (FRI – SAT)

pythonbygupta@gmail.com