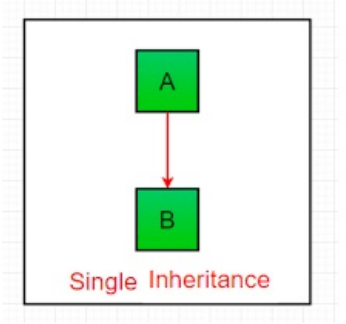
Single Inheritence In Python

By Sudhanshi

* Inheritence is one of the most importance concept in oops(object-oriented Programming)

* When a class inherits another class then it's called single inheritance.

*The most important advantage of inheritance is reusability of code.



In [1]:
```
class Parent:  #Parent class or base class
    def funct1(self):
        print("This function is in Parent class")

class Child(Parent):   #Drive class or child class
    def funct2(self):
        print("This function is in Child class")
```

In [2]:
```
object=Child()
object.funct1()
```

```
This function is in Parent class
```

In [3]:
```
object.funct2()
```

```
This function is in Child class
```

In the example given below, Persondetails class inherits the Student class, so there is the single inheritance.

In [25]:
```
class Student:
    def Student(self,ID):
        self.ID = ID

class Persondetails(Student):
    def Persondetails(self,name,Class,Language):
        self.name= name
        self.Class = Class
        self.Language = Language

    def display(self):
        print('ID :',self.ID, '\nNAME :',self.name, '\nClass : ',self.Class,'\nLANGUAGE: ',self.Language)


s1 = Persondetails()

s1.Student(101)
s1.Persondetails('Rohit','Vth','Marathi')
```

In [26]:
```
s1.display()
```

```
ID : 101
NAME : Rohit
Class :  Vth
```

LANGUAGE: Marathi

Note 1: The number of argunments in parameters should be same or else u will get an error

In [8]:
```
class Employee:
    def __init__(self,id,name,salary,role,skills):
        self.id=id
        self.name=name
        self.salary=salary
        self.role= role
        self.skills = skills
    def empdetails(self):
        return ('Employee id:',self.id,
              'Name:',self.name,
              'Salary:',self.salary,
              'Role:',self.role)


class Programmer(Employee):
    def printprog(self):
        return('Employee id:' ,self.id,
              'Name:',self.name,
              'Salary:',self.salary,
              'Role:',self.role,
              'Skills:',self.skills)
```

In [10]:
```
Tom=Employee(11101,"Tom",25000,'Marketing')
Dick=Employee(11102,'Dick',52000,'IT')
Harry=Employee(11103,'Harry',59000,'IT')

karthik=Programmer(11104,'Karthik',36550,'Programmer',"Python")
Sanjana=Programmer(11105,'Sanjana',48000,'Programmer','Java')
Newton=Programmer(11106,'Newton',56000,'Programmer','C++')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-10-39420286ef2c> in <module>
----> 1 Tom=Employee(11101,"Tom",25000,'Marketing')
      2 Dick=Employee(11102,'Dick',52000,'IT')
      3 Harry=Employee(11103,'Harry',59000,'IT')
      4
      5 karthik=Programmer(11104,'Karthik',36550,'Programmer',"Python")

TypeError: __init__() missing 1 required positional argument: 'skills'
```

NOTE 2: If we want to retrieve those data-set which is present in child inheritence class then we have to comment parent class skills

In [13]:
```
class Employee:
    def __init__(self,id,name,salary,role,skills):
        self.id=id
        self.name=name
        self.salary=salary
        self.role= role
        self.skills = skills
    def empdetails(self):
        return ('Employee id:',self.id,
              'Name:',self.name,
              'Salary:',self.salary,
              'Role:',self.role)
              #'Skills:',self.skills)


class Language(Employee):
    def printlang(self):
        return('Employee id:' ,self.id,
              'Name:',self.name,
              'Salary:',self.salary,
              'Role:',self.role,
              'Skills:',self.skills)
```

In [16]:
```
Tom=Employee(11101,"Tom",25000,'Marketing','Digital Marketing')
Dick=Employee(11102,'Dick',52000,'IT','SQL')
Harry=Employee(11103,'Harry',59000,'IT','Tableau')

karthik=Language(11104,'Karthik',36550,'Programmer',"Deutsch")
```

```
Sanjana=Language(11105,'Sanjana',48000,'Programmer','French')
Newton=Language(11106,'Newton',56000,'Programmer','Spansih')
```

In [17]:
```
print(karthik.printlang())
```

```
('Employee id:', 11104, 'Name:', 'Karthik', 'Salary:', 36550, 'Role:', 'Programmer', 'Skills:', 'Deutsch')
```

In [18]:
```
print(Dick.empdetails())
```

```
('Employee id:', 11102, 'Name:', 'Dick', 'Salary:', 52000, 'Role:', 'IT')
```

In [19]:
```
class Person:
    def __init__ (self,name,age):
        self.name=name
        self.age=age
    def display(self):
        print('Name :',self.name)
        print('Age :',self.age)

class Student(Person):
    def __init__(self,rollno,name,age,per):
        self.rollno=rollno
        Person.__init__(self,name,age)
        self.per=per
    def display(self):
        print('Roll no :',self.rollno)
        Person.display(self)
        print('Percentage :',self.per)

S1= Student(101,"Adwait",15,89.50)
print('Student Details :',end='')
S1.display()
```

```
Student Details :Roll no : 101
Name : Adwait
Age : 15
Percentage : 89.5
```

In [20]:
```
class Employee:
    def Empdetails(self,id,name,salary,department):
        self.id=id
        self.name=name
        self.salary=salary
        self.department=department

class Programmer(Employee):
    def skill(self,skills):
        self.skills=skills

    def display(self):
        print( 'Employee id is:',self.id,'\nEmployee name is :',self.name,'\nSalary is :',self.salary,'\nDepartme
```

In [21]:
```
Details = Programmer()

Details.Empdetails(101,'Tom',45000,'IT')
Details.skill('Python')
Details.display()
```

```
Employee id is: 101
Employee name is : Tom
Salary is : 45000
Department is: IT
Skills : Python
```

In [22]:
```
class Addition:
    num1= 200
    num2= 300
    def add(self):
        sum=self.num1+self.num2
        print("Addition of num1 and num2 is:",sum)

class Myclass(Addition):
```

```
    num3 = 5
    def multiply(self):
        multiply = self.num1+self.num2 * self.num3
        print("Output is:",multiply)
```

In [23]:

```
operation= Myclass()
operation.add()
```

```
Addition of num1 and num2 is: 500
```

In [24]:

```
operation.multiply()  # 200+300*5
                      # 200+1500
                      #1700
```

```
Output is: 1700
```

In [ ]: