Python Inheritance:

Inheritance is the capability of one class to derive or inherit the properties from another class. The class that derives properties is called the derived class or child class and the class from which the properties are being derived is called the base class or parent class.

In [1]:
```
#Lets look basic example first
```

In [2]:
```
class Person:
    def __init__(self,name,idnumber):
        self.name=name
        self.idnumber=idnumber

    def display(self):
        print("Person name is {}".format(self.name))
        print("Person idnumber is {}".format(self.idnumber))

#create object

person1=Person("Ram",12034)
```

In [3]:
```
print(person1.name)
```

```
Ram
```

In [4]:
```
person1.display()
```

```
Person name is Ram
Person idnumber is 12034
```

In [5]:
```
#this is normal class and object we were working on without inheritance, now lets create parent class & child c
```

In [8]:
```
class Person:
    def __init__(self,name,idnumber):
        self.name=name
        self.idnumber=idnumber

    def display(self):
        print("Person name is {}".format(self.name))
        print("Person idnumber is {}".format(self.idnumber))

class Man(Person):
    def __init__(self,name,idnumber,salary,address):
        self.salary=salary
        self.address=address

        Person.__init__(self,name,idnumber)

    def details(self):
        print("Man1 name is {}".format(self.name))
        print("Man1 salary is {}".format(self.salary))


#create object

person1=Man("Ram",12034,20000,"highway")
```

In [9]:
```
person1.details()
```

```
Man1 name is Ram
Man1 salary is 20000
```

In [10]:
```
#this is called inheritance. We will discuss more and look more example, it just a glance look to understand th
```

Python Polymorphism

Polymorphism simply means having many forms. For example, we need to determine if the given species of birds fly or not, using polymorphism we can do this using a single function.

One more Example: Like if we want to book tickets then there will be two option- Online and Physical we can use Polymorphism using single function

In [11]:
```
class Bird:

        def intro(self):
                print("There are many types of birds.")

        def flight(self):
                print("Most of the birds can fly but some cannot.")

class sparrow(Bird):

        def flight(self):
                print("Sparrows can fly.")
```

```
class ostrich(Bird):

        def flight(self):
                print("Ostriches cannot fly.")

obj_bird = Bird()
obj_spr = sparrow()
obj_ost = ostrich()

obj_bird.intro()
obj_bird.flight()

obj_spr.intro()
obj_spr.flight()

obj_ost.intro()
obj_ost.flight()
```

```
There are many types of birds.
Most of the birds can fly but some cannot.
There are many types of birds.
Sparrows can fly.
There are many types of birds.
Ostriches cannot fly.
```

Python Encapsulation

In [12]:
```
class Person:
    def __init__(self,name,idnumber):
        self.name=name
        self._idnumber=idnumber

    def display(self):
        print("Person name is {}".format(self.name))
        print("Person idnumber is {}".format(self.idnumber))

class Man(Person):
    def __init__(self,name,idnumber,salary,address):
        self.salary=salary
        self.address=address

        Person.__init__(self,name,idnumber)

    def details(self):
        print("Man1 name is {}".format(self.name))
        print("Man1 salary is {}".format(self.salary))
```

In [13]:
```
person1=Man("Ram",12034,20000,"highway")
```

In [14]:
```
print(person1.name)
```

```
Ram
```

In [15]:
```
print(person1.idnumber)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[15], line 1
----> 1 print(person1.idnumber)

AttributeError: 'Man' object has no attribute 'idnumber'
```

In [16]:
```
#see it give error because it is private attribute and we cannot access it.
```

This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data.