Object Oriented Programming

Class

In [8]:
```
class Human:
    color='Black'
    Height=5.8
    def detials(self):
        print("The name of the person")
    def action(self):
        print("Sleeping")
teja=Human()
teja.detials()
teja.action()
```

```
The name of the person
Sleeping
```

In [9]:
```
# In the above example we Just call the method by using the reference variabl
```

In [12]:
```
print(teja.color,teja.Height)
```

```
Black 5.8
```

In [13]:
```
# By looking the above example we can learn how to Call Method and Variables
```

We can create the Multiple Objects and pass the values through it

In [16]:
```
class Human1:
    color='Black'
    Height=5.8
    def detials(self,name,age):
        print("The name of the person:",name)
        print("My age is:",age)
teja1=Human1()
teja1.detials('Teja',25)
```

```
The name of the person: Teja
My age is: 25
```

In [17]:
```
class Human1:
    color='Black'
    Height=5.8
    def detials(self,name,age):
        print("The name of the person:",name)
        print("My age is:",age)
        print(color)
teja1=Human1()
teja1.detials('Teja',25)
```

```
The name of the person: Teja
My age is: 25

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_11108/405703685.py in <module>
      7         print(color)
      8 teja1=Human1()
----> 9 teja1.detials('Teja',25)

~\AppData\Local\Temp/ipykernel_11108/405703685.py in detials(self, name, ag
e)
      5         print("The name of the person:",name)
      6         print("My age is:",age)
----> 7         print(color)
      8 teja1=Human1()
      9 teja1.detials('Teja',25)

NameError: name 'color' is not defined
```

In [18]:
```
class Human1:
    color='Black'
    Height=5.8
    def detials(self,name,age):
        print("The name of the person:",name)
        print("My age is:",age)
        print(teja1.color)
teja1=Human1()
teja1.detials('Teja',25)
```

```
The name of the person: Teja
My age is: 25
Black
```

In [19]:
```
# I created Second object also in same way we can created Multiple objects
tejadub1=Human1()
tejadub1.detials('swaroop',28)
```

```
The name of the person: swaroop
My age is: 28
Black
```

Constructors are generally used for instantiating an object.

The task of constructors is to initialize(assign values) to the data members of the class when an object of the class is created. In Python the **init**() method is called the constructor and is always called when an object is created.

In [21]:
```
class HumanHero:
    # creating the constructor
    def __init__(self,name,age,height,color):
        self.name=name
        self.age=age
        self.height=height
        self.color =color
    # creating the method
    def details(self):
        print("My name is :",self.name)
        print("My age is :",self.age)
        print("My height is:",self.height)
        print("My color is :",self.color)
# Creating the objects
object1=HumanHero()
object1.details("Teja",25,5.7,"black")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_11108/215897170.py in <module>
     13         print("My color is :",self.color)
     14 # Creating the objects
---> 15 object1=HumanHero()
     16 object1.details("Teja",25,5.7,"black")

TypeError: __init__() missing 4 required positional arguments: 'name', 'ag
e', 'height', and 'color'
```

In [22]:
```
class HumanHero:
    # creating the constructor
    def __init__(self,name,age,height,color):
        self.name=name
        self.age=age
        self.height=height
        self.color =color
    # creating the method
    def details(self):
        print("My name is :",self.name)
        print("My age is :",self.age)
        print("My height is:",self.height)
        print("My color is :",self.color)
# Creating the objects
object1=HumanHero("Teja",25,5.7,"black")
object1.details()
```

```
My name is : Teja
My age is : 25
My height is: 5.7
My color is : black
```

Above we observe when-ever we using the constructor we should pass the values in the class only not method

In [23]:
```
# Create more objects
```

In [24]:
```
object2=HumanHero("Tswarup",26,5.8,"brown")
object2.details()
```

```
My name is : Tswarup
My age is : 26
My height is: 5.8
My color is : brown
```

In [25]:
```
object3=HumanHero("Ts",26,5.9,"white")
object3.details()
```

```
My name is : Ts
My age is : 26
My height is: 5.9
My color is : white
```

Inheritances

* Single Inheritances

* Multi-level Inheritances

* Multiple Inheritances

* Hierarchical Inheritances

* Hybrid Inheritances

In [29]:
```
# Single In heritances
#---------------------------------------------
class seeds:
    ass="apple seeds"
    bss="banana seeds"
    css="cheery seeds"
    def details(self):
        print("This all are seeds of fruits")
class tress(seeds):
    att="Apple Tree"
    btt="Banana Tree"
    ctt="Cheery Tree"
    def detailsTree(self):
        print("This all are seeds of fruits Trees")

# Creating the obj for child class and through child class invokinng the pare
objj1=tress()
print(objj1.att)
# ass is parent class variable
print(objj1.ass)
```

```
Apple Tree
apple seeds
```

In [33]:
```
# Hierarchical In-heritances
#-------------------------------------------
class seeds:
    ass="apple seeds"
    bss="banana seeds"
    css="cheery seeds"
    def details(self):
        print("This all are seeds of fruits")
class tress(seeds):
    att="Apple Tree"
    btt="Banana Tree"
    ctt="Cheery Tree"
    def detailsTree(self):
        print("This all are seeds of fruits Trees")
class wood(tress):
    aww="Apple Tree Wood"
    bww="Banana Tree Wood"
    cww="Cheery Tree Wood"
    def detailsTree(self):
        print("This all are seeds of fruits Trees Wood")

# Creating the obj for child class and through child class invokinng the pare
objj2=wood()
print(objj2.att)
# ass is parent class variable
print(objj2.ass)
# bww is
print(objj2.bww)
objj2.details()
```

```
Apple Tree
apple seeds
Banana Tree Wood
This all are seeds of fruits
```

In [34]:
```
# if method name is same it wil take first class method
```

In [42]:
```
# MultipleIn-heritances
#---------------------------------------------
class seeds:
    ass="apple seeds"
    bss="banana seeds"
    css="cheery seeds"
    def details(self):
        print("This all are seeds of fruits")
class tress(seeds):
    att="Apple Tree"
    btt="Banana Tree"
    ctt="Cheery Tree"
    def detailsTree(self):
        print("This all are seeds of fruits Trees")
class wood(seeds):
    aww="Apple Tree Wood"
    bww="Banana Tree Wood"
    cww="Cheery Tree Wood"
    def detailsTree(self):
        print("This all are seeds of fruits Trees Wood")

# Creating the obj for child class and through child class invokinng the pare
objj3=wood()
print(objj3.ass)

print("*"*20)

objj4=tress()
print(tress.btt)
```

```
apple seeds
********************
Banana Tree
```

In [47]:
```
# MultipleIn-heritances
#-------------------------------------------
class father:
    color ="black"
    def display(self):
        print("color is black")
class mother:
    height=5.7
    def display(self):
        print("height is 5.7")
class child(father,mother):
    pass
obey=child()
print(obey.color)
print(obey.height)
obey.display()
```

```
black
5.7
color is black
```

Hybrid Inheritances is the combination of all

In [ ]:
```

```