OOPs Concept

By Sudhanshi

In [1]:
```
class Square():          #Square is a name of the class
    def __init__(self): #Defining constructor
        self.length = 24
        self.breadth = 24
```

Note 1: If we want to call out the instance variable which is under the class then we have to create the instance

In [2]:
```
sq = Square() #Here I m creating a class instance or instanting a class
sq.length  # Here I am calling the instance variable
```

Out[2]:
```
24
```

In [4]:
```
sq = Square ()
print("Length = ",sq.length , "\nBreadth = ",sq.breadth)
```

```
Length =  24
Breadth =  24
```

Note 2: If you define the parameters then rectangle() won't work until you didn't give the arguments

In [5]:
```
class Rectangle():
    def __init__(self,length,breadth):
        self.length = length
        self.breadth = breadth
```

In [6]:
```
rect = Rectangle()
rect.length
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-279e450803e5> in <module>
----> 1 rect = Rectangle()
      2 rect.length

TypeError: __init__() missing 2 required positional arguments: 'length' and 'breadth'
```

In [7]:

```
rect= Rectangle(30,24)
print("Length = ",rect.length, "\nbreadth = ",rect.breadth)
```

```
Length =  30
breadth =  24
```

Note 3: Parameters and variables order should be same

In [8]:

```
class employees():           #Here i create a class called employees
    def __init__(self,Name,Age,Salary,Department,Gender,Experience):
        self.Name =Name
        self.Age =Age
        self.Salary =Salary
        self.Department =Department
        self.Gender= Gender
        self.Experience = Experience

Emp_001 = employees('Sudhanshi','Female','IT','8 Years',42,520000)
Emp_002 = employees('Nivetha','Female','Marketing','5 Years',32,20000)
Emp_003 = employees('Akash','Male','IT','9 Years',44,580000)

print(Emp_001.Gender)
```

```
42
```

Note 4: We can simply change the class variable by assigning a new value

In [9]:

```
class circle():
    pie = 3.14
    def __init__(self,radius):
        self.radius = radius
```

In [10]:
```
circle_1= circle(8)
print("Radius = {} \t pi = {}".format(circle_1.radius,circle_1.pie))
```

```
Radius = 8       pi = 3.14
```

In [11]:
```
circle_1= circle(8)
circle_1.pie
```

Out[11]:
```
3.14
```

In [12]:
```
circle.pie = 3.1436
circle_1= circle(8)
circle_1.pie
```

Out[12]:
```
3.1436
```

Methods in OOPs

In [13]:
```
#Define a class addition
class addition:

    #define a constructor
    def __init__ (self,num1,num2):
        self.num1 = num1
        self.num2 =num2

        # define add() instance method
    def add(self):
        return self.num1 + self.num2
```

In [14]:
```
# create an instance that is object of class Addition
obj1 = addition(160, 20)

# call add method with obj1 object
```

```
sum = obj1.add()
print("Sum = ", sum)
```

```
Sum =  180
```

In [18]:
```
class rectangle():
    def __init__ (self,length,breadth):
        self.length = length
        self.breadth = breadth

    #Area of a rectangle
    def multiplication(self):
        return self.length*self.breadth
```

In [19]:
```
Area = rectangle (200,160)
multiply = Area.multiplication()
print("Area = ",multiply)
```

```
Area =  32000
```

In [ ]:

In [ ]:

In [ ]: