What is object oriented programming(oops)

- OOPs allows decomposition of aproblem into a no. of units called Objects.
- python is an object oriented programming language.

Why we have to use OOPs?

- It provides a clear programming structure.
- It makes the development and maintanence easier.
- code reusability.

CLASS

- class is a collection of variables and functions.

  Syntax: class ClassName:

  ```
              list of variables
              list of methods
  ```

OBJECT

- An object is also called an instance of a class.
- An object is a collection of data and methods.

  Syntax: objectname = className

if a function is called in class it is called method(here function is called collection of statements)

In [16]:

```python
# example for class creation

class Hi:
    a,b=10,12
    def display():
        print("Hi,I am from display function")
        return 9

obj = Hi
print(obj.a)
print(obj.b)
print(obj.display())
```

```
10
12
Hi,I am from display function
9
```

In [4]:

```python
class math:
    def add(n1,n2):
        return n1+n2
    def mul(n1,n2):
        return n1*n2

obj = math
print(obj.add(12,13))
print(obj.mul(2,3))

```

```
25
6
```

constructor

- Its a task is to initialize to the data members of a class when an object of a class is created.

```
Syntax:
        class className:
            def__init__(self): It is constructor
            def__init__(self,a,b):
            def__init__(a,b,self):
```

- The self parameter is a reference to the current instance of the class,and is used to access variable that belongs to the class

In [7]:

```python
class math:
    def __init__(self,n1,n2):
        self.n1 = n1
        self.n2 = n2
    def show(self):
        print(self.n1)
        print(self.n2)
obj = math(2,5)
obj.show()

```

```
2
5
```

In [9]:

```
class math:
    def __init__(abc,n1,n2):
        abc.n1 = n1
        abc.n2 = n2
    def show(abc):
        print(abc.n1)
        print(abc.n2)
obj = math(2,5)
obj.show()

```

```
2
5
```

In [10]:

```
class Myclass:
    x = 5

print(Myclass)
```

```
<class '__main__.Myclass'>
```

inheritance

- acquiring properties from parent class to child class

single inheritance

- one child class and one parent class

In [15]:

```
class A:  #parent class
    a,b = 10,12
    def display():
        print("Iam from class A")
class B(A): #child class
    c,d = 13,15
    def show():
        print("Iam from class B")
obj = B #we have to create object for only child class,by creating for it we may able t
print(obj.a)
obj.display()
obj.show()
```

```
10
Iam from class A
Iam from class B
```

In [16]:

```
class A:  #parent class
    a,b = 10,12
    def display():
        print("Iam from class A")
class B(A): #child class
    c,d = 13,15
    def show():
        print("Iam from class B")
obj = A
print(obj.a)  ## by creating object for parent class we unable to access the methods
                #in child class(show method which in child class)
obj.display()
obj.show()
```

```
10
Iam from class A

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-16-8c7765006ed4> in <module>
     10 print(obj.a)
     11 obj.display()
---> 12 obj.show()

AttributeError: type object 'A' has no attribute 'show'
```

Multilevel inheritance

- one or more parent classes and one or more child classes

In [19]:

```
class A: # parentclass to B and C
    def classA():
        print("Iam from classA")
class B(A): # childclass to A and parent class to C
    def classB():
        print("Iam from classB")
class C(B): # child class both A and B
    def classB():
        print("Iam from classB")
obj = C # we may able to access methods and variables in both A nd B Classes
obj.classA()
obj.classB()
```

```
Iam from classA
Iam from classB
```

MULTIPLE INHERITANCE

- more than one parent class and one child class

In [22]:

```
class A:
    def classA():
        print("Iam from cse-A")
class B:
    def classB():
        print("Iam from cse-B")
class C(A,B):
    def classC():
        print("Iam from CSE")
obj = C
obj.classA()
obj.classB()
```

```
Iam from cse-A
Iam from cse-B
```

In [ ]:

```

```