7. OOPS

```
1. Class
2. Object / Instance
3. Encapsulation
4. Abstraction
5. Constructors & Destructors
6. Magic Methods (__str__)
7. Methods and Variable
8. Messege Passing / Shared Memory
9. Getter & Setter Properties
10. Composition
11. Inheritance
12. MRO(Method Resolution Order)
13. Super() Method
14. Aggregation
15. Polymorphism - Over-riding & Overloading
16. Duck Typing - methods are more important than class type
17. Access Specifier - Data Hiding / Name Mangling
18. Abstract Class and Abstract Methods
19. Interfaces
20. Dynamic Binding / Monkey Patching
21. Properties
22. Slots
23. Singleton Class
24. Meta Class
```

1. Class

What is Class ?

```
⚽ In Python every thing is an object. To create objects we required some Model or
Plan or Blue
print, which is nothing but class.
⚽ We can write a class to represent properties (attributes) and actions
(behaviour) of object.
⚽ Properties can be represented by variables
⚽ Actions can be represented by Methods.
⚽ Hence class contains both variables and methods.
```

How to define a class ?

```
class className:
    ''' documenttation string '''
    variables: instance variables,static and local variables
    methods: instance methods,static methods,class methods
```

```
Documentation string represents description of the class. Within the class doc
string is always optional. We can get doc string by using the following 2 ways.
```

1. print(classname.__doc__)
2. help(classname)

In [2]:
```
class Student:
    """This is Student Class"""

print(Student.__doc__)
help(Student)
```

```
This is Student Class
Help on class Student in module __main__:

class Student(builtins.object)
 |  This is Student Class
 |
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

Types of variables

1. Instance Variables (Object Level Variables)
2. Static Variables (Class Level Variables)
3. Local variables (Method Level Variables)

Types of Methods

1. Instance Methods
2. Class Methods
3. Static Methods

In [7]:
```
l1 = [1, 2, 3, 4]

help(list.append) #class.method
print('-'*50)

help(l1.append) #object.method
print('-'*50)

list.append(l1,10)
print(l1)

l1.append(50)
print(l1)
```

```
Help on method_descriptor:

append(self, object, /)
    Append object to the end of the list.

--------------------------------------------------
Help on built-in function append:

append(object, /) method of builtins.list instance
    Append object to the end of the list.

--------------------------------------------------
```

```
[1, 2, 3, 4, 10]
[1, 2, 3, 4, 10, 50]
```

2. Object

What is an Object ?

```
Pysical existence of a class is nothing but object. We can create any number of
objects for a class.

real Time Entity

Represent any real time object in program
```

Examples

Person

```
(Data) Attributes - name, age, color, height, weight, education, ... (Property,
info)

(Function) Methods - learn, Fight, cry, sing, walk, ....
```

Customer

```
Attributes - name, acc no, balance, password, ...
Methods - credit, debit, balance_enquiry
```

Syntax: referencevariable = classname()

```
Example:-

s = Student()
```

Reference Variable

```
The variable which can be used to refer object is called reference variable.
By using reference variable, we can access properties and methods of object.
```

In [33]:

```
class Person:
    def laugh(self): #Instance / Object method
        print("Ha ha ha haha ha aha hah a")
    def cry(self): #Instance / Object method
        print("ohuuu ohuu ohuuuuuu")
    def function(self): #Instance / Object method
        """Just Printing self to undestand object reference"""
        print(self)
        print(id(self))
```

In [48]:

```
print(Person)
print(dir(Person))
help(Person.function)
```

```
<class '__main__.Person'>
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__
ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le_
_', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr_
_', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'cry', 'fun
ction', 'laugh']
Help on function function in module __main__:

function(self)
    Just Printing self to undestand object reference
```

In [34]:
```
p1 = Person() #p1 is reference variable to object of class Person
p2 = Person() #p2 is reference variable to object of class Person
```

In [36]:
```
print(p1, id(p1))
print(p2, id(p2))

p1.function() #calling through instance
Person.function(p1) #calling through class

p2.function()
```

```
<__main__.Person object at 0x000001FEFB8BF130> 2194653573424
<__main__.Person object at 0x000001FEFB88CE80> 2194653367936
<__main__.Person object at 0x000001FEFB8BF130>
2194653573424
<__main__.Person object at 0x000001FEFB8BF130>
2194653573424
<__main__.Person object at 0x000001FEFB88CE80>
2194653367936
```

In [37]:
```
p1.laugh() #instance method of p1
p2.laugh() #instance method of p2
```

```
Ha ha ha haha ha aha hah a
Ha ha ha haha ha aha hah a
```

In [39]:
```
print(p1.laugh) #instance method
print(p2.laugh) #instance method
print(Person.laugh) #class method
```

```
<bound method Person.laugh of <__main__.Person object at 0x000001FEFB8BF130>>
<bound method Person.laugh of <__main__.Person object at 0x000001FEFB88CE80>>
<function Person.laugh at 0x000001FEFB95FCA0>
```

3. Features of OOPs

1. Encpsulation

```
    Process by which we can bind together data memebers and member function in a
single unit known as class
```

In [29]:
```
class Animal:
    def laugh(self):
        print('Hooooo hoooooo hooooo')

dog = Animal()
```

In [30]:
```
p1.laugh()
dog.laugh()
```

```
Ha ha ha haha ha aha hah a
Hooooo hoooooo hooooo
```

2. Abstraction

```
   Only Showing Essential details to user while hiding background information
```

In [40]:
```python
print("Hello World")
```

```
Hello World
```

In [45]:
```python
p1.cry()
p1.laugh()
```

```
ohuuu ohuu ohuuuuuu
Ha ha ha haha ha aha hah a
```

In [44]:
```python
r = range(1,10)
print(type(r))
print(*r)
```

```
<class 'range'>
1 2 3 4 5 6 7 8 9
```

Access Specifiers

3. Data Hiding

```
    Hiding Some Information (attributes) for Direct Access from outside the class
```

In [50]:
```python
class A:
    def __hi(self):
        print("Hi World!!")
    def bye(self):
        print("Bye World!!")
    def hello(self):
        print("Hello World!!")
        self.__hi()
```

In [51]:
```python
a = A()
```

In [52]:
```python
a.__hi() # Direct Access to __method is not allowed outside the class, here __hi is a hid
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_2188/145997960.py in <module>
----> 1 a.__hi() # Direct Access to __method is not allowed outside the class, here __hi i
s a hidden method

AttributeError: 'A' object has no attribute '__hi'
```

In [54]:
```python
a.hello() #we can access through another method
```

```
Hello World!!
Hi World!!
```

In [ ]:
```python

```