30th July 2025

OOPS CONCEPT

- CLASS
- OBJECT
- METHOD
- INHERITENCE
- ENCAPSULATION
- ABSTRACTION
- POLYMORPHISM

PYTHON OBJECT ORIENTED PROGRAMMING

- We discussed what is function & how to pass variable to the function and then we moved to module
- Next move towards concept & one of the concept is OOPS concept
- Python famous cuz fuctional orinetied programming language, do support procedure oriented & object oriented programming language
- What is procedure - we call the function inside the function & breakdown the project to small module.
- What is mean by object -- in real time problem we are trying to solve virtual world solution
- E.g if i want to call anyone i need object which is phone, for type a code need laptop, laptop is object. in real world everything is object
- Every object has certain attribute & certain behaviour
- Attribute (my weight, my name, company which i am working is attribute)

- Behaviour - I am talking, I am explaining (these are behaviour)
- Action defines behaviour.
- Object is something where we can store the data & object will some thing will behaviour.
- Function in object oriented is called methods
- Any programming language you choose always class & object will together but why
- Company manufacture the mobile phone. many people use the phone those are the object
- Phone design is import.
- Phone design at once & manufacture plenty of phone
- Design in oops concept is called CLASS
- CLASS - DESIGN || OBJECT - INSTANCE
- We can called as INSTANCE OF THE CLASS
- Phone is the instance of the class Phone design
- class - blueprint which followed object
- class - design of the object. without design we cant build object
- before we build the tower or house we need the blueprint of the object

In [2]:
```
a = 5
type(a)
```

Out[2]:
```
int
```

In [3]:
```
a = 5
print(type(a))
```

```
<class 'int'>
```

In [5]:
```
l = [1,2,3,]
print(type(l))
```

```
<class 'list'>
```

In [7]:
```
class computer():  # computer is the class

    def config():  # config is the methods
        print('i5', '8gb')

com1 = computer()
com1
```

Out[7]:
```
<__main__.computer at 0x21a45bc23c0>
```

In [8]:
```
class computer():  # computer is the class

    def config():  # config is the methods
        print('i5', '8gb')

com1 = computer()
```

In [9]:
```
class computer():  # computer is the class

    def config():  # config is the methods
        print('i5', '8gb')

com1 = computer()

computer.config()
```

```
i5 8gb
```

In [10]:
```
class computer():  # computer is the class

    def config():  # config is the methods
        print('i5', '8gb')

com1 = computer()
```

```
com1.config()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 8
      4         print('i5', '8gb')
      6 com1 = computer()
----> 8 com1.config()

TypeError: computer.config() takes 0 positional arguments but 1 was given
```

In [11]:
```
class computer():  # computer is the class

    def config(self):  # config is the methods
        print('i5', '8gb')

com1 = computer()

com1.config()
```

```
i5 8gb
```

In [ ]:
```

```