In [ ]:
```
OOPS--> object + oriented + programming + lanaguage
```

In [ ]:
```
Five pillar of OOPS

- Class
- Object
- Inheritance
- Polymorphism
- Encapsulation
```

In [ ]:
```
Class - the class can be defined as a collection of objects or blueprint.
```

In [ ]:
```
# Syntax

class <classname>(parameter):
    <statement>
```

In [ ]:
```
class Parrot():

    # Class attribute
    name = "John"
    color = "green"
    age = 20

    def __init__(self):
        pass

    # class methods
    def fly(self):
        print("I can fly")
```

In [ ]:
```
# what is the differnce between function and method?
```

In [ ]:
```
# Object ->object is the instance of the class


<object_variable_name> = <class_name>
```

In [ ]:
```
obj = Parrot()
print(obj.name)
print(obj.color)
print(obj.age)

obj.fly()
```

In [ ]:
```
obj2 = Parrot()
print(obj2.name)
print(obj2.color)
print(obj2.age)

obj2.fly()
```

In [ ]:
```
obj3 = Parrot()
print(obj3.name)
print(obj3.color)
print(obj3.age)

obj3.fly()
```

In [ ]:
```
obj4 = Parrot()
print(obj4.name)
print(obj4.color)
print(obj4.age)

obj4.fly()
```

In [ ]:
```
class Person():

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def run(self):
        print("I can run")

    def __str__(self):
        return self.name

    def __len__(self):
        return len(self.name)


p = Person("John",20)
print(p.name)
print(p.__str__())
print(p.__len__())
p.run()
```

In [ ]:
```
## Write a class for human,Birds,Snake and use above all the methods,attribute
```

In [ ]:
```
## Inheritance -->

## types of Inheritance
1- Single Inheritance
2- Multilevel Inheritance
3- Multiple Inheritance
4- hierarchical Inheritance
5- Hybrid Inheritance
```

In [ ]:
```
## Single Inheritance

class parentClass():  ## Base  class
    name = "God Fellow"
    age = 38


    def display(self):
        print("I can display")


class childCLass(parentClass): ## Derived class
    pass
```

In [ ]:
```
c = childCLass()
print(c.name)
print(c.age)
c.display()
```

In [ ]:
```
## Multilevel Inheritance

class GrandFather():

    def name(self):
        print("I am Grandfather")


class Father(GrandFather):

    def play(self):
        print("I love to hockey")


class Child(Father):
    pass



c1 = Child()
c1.play()
c1.name()
```

In [ ]:
```
## Multiple Inheritance


class human():

    def work(self):
        print("i can work")

class noHuman():

    def donotwork(self):
        print("I do not work like human")


class Work(human,noHuman):
    pass


w = Work()
w.work()
w.donotwork()
```

In [ ]:
```
# Hierarchical Inheitance
```

In [7]:
```
class Main():

    def run(self):
        print("I can run ")

class h1(Main):
    def display(self):
        print("I am the H1 class")

class h2(Main):
    def display(self):
        print("I am the H2 class")

class h3(Main):
    def display(self):
        print("I am the H3 class")

m1 = h1()
m1.run()
m1.display()
```

```
I can run
I am the H1 class
```

Hybrid Inhritance

In [12]:
```
class Base():

    def b1(self):
        print("I am the Base Class")

class base2(Base):

    def b2(self):
        print("I am the base2 Class")

class base3(Base):

    def b3(self):
        print("I am the base3 Class")

class base4(base2,base3):
    pass


b = base4()
b.b1()
b.b2()
b.b3()
```

```
I am the Base Class
I am the base2 Class
I am the base3 Class
```

In [ ]:
```
# Polymorphism

#poly = many
# morphism = forms

1- overloading

2- overriding
```

In [13]:
```
class Parrot():

    def fly(self):
        print("Parrot can fly")

    def swim(self):
        print("Parrot can't swim")

class Penguin():

    def fly(self):
        print("I can't fly")

    def swim(self):
        print("I can swim")

def flying_test(bird):
    bird.fly()

p = Parrot()
p2 = Penguin()

flying_test(p)
flying_test(p)
```

```
Parrot can fly
Parrot can fly
```

In [17]:
```
## Overloading

class Parrot3():

    def fly(self):
        print("Parrot can fly ")

    def swim(self):
        print("Parrot can't swim")

    def fly(self):
        print("Parrot can fly but can fly very high altitude")

    def fly(self):
        print("Parrot can fly very high!!!")

p4 = Parrot3()
p4.fly()
```

```
Parrot can fly very high!!!
```

In [18]:
```
## Overriding

class Parrot1():

    def fly(self):
        print("Parrot1 can fly ")

    def swim(self):
        print("Parrot1 can't swim")

class Parrot2(Parrot1):

    def fly(self):
        print("Parrot2 can fly==============")

    def swim(self):
        print("Parrot2 can't swim")


p5 = Parrot2()
p5.fly()
```

```
Parrot2 can fly==============
```

In [ ]:
```

```