Hi guys, thanks for staying this long with me. Akshansh (+91 8384891269, akshanshoffical@gmail.com (mailto:akshanshoffical@gmail.com)) here and today we are going to discuss one of the most important topic in programming.

You may have heard the name of Object Oriented Programming? We are going to understand what is it and how does it work. I'll call OOP to Object Oriented Programming because you will have lot of oop's moments with this particular topic.

When I was learning this, it bounced over my head and I tried to get this but unfortunately I wasn't able to.I hope I'll be able to make you understand what OOP is. If you don't get confused while working with OOP, thank me later as I've done lot of work to make this easy for you.

Let's Get Started

OOP is something truely important. It relates all the work of programming with real world around you.

We'll be covering these - 1) Classes 2) object instances 3) working with methods 4) OOP Inheritance

Class & Objects in python

Earlier I said, OOP makes you thing relate with real life example. Let's take a realistic example. Think of a girl (aha, not your crush, I know her face just pop up in your mind).

Think of a common girl, not just a particular girl. What do all girls have- name,age etc (have you ever seen a girl without name? without age, I can consider). These all things will be found in a girl, right. Suppose you are god and you are making a girl, you'll think that it should has hands,legs,head,face etc. Once you are done with it, you will decide what skintone,weight,height,smile,cuteness etc she is going to have. (I took example of girls instead of boys, because boys usually don't have qualities like cuteness skintone, they all are same :-P)

So girl will be the class (main focussed thing, as we usually do in real life too) here, and all the common things like name,age (all girls have these, no matter which skin tone she has, what is her heigh, what is her wieght) will be arguments.

arguments in OOP will be called attributes.

so when we write Girl, it is just a model (class) in which name, age, legs,hands,face,head etc will be required (those are arguments/attributes).

things those are common in every girl (name, age etc) will be called arguments/attributes in OOP.

If we think of my crush named "SANA", you'll notice that she is a girl too (definitely, I am not gay). Sana is a part of category Girl . She is a part of class (Girl), she will be called, object. Object is also called instance in OOP.

instance in OOP.

making an object is called instancialtion. In other words creating instance (part of class) is called instanciation.

each girl has her own quality, like Sana has - cuteness, beautiful, lovely, adorable etc. (don't get jealous man, she is crush he he he). And when we will be creating Sana as a part of Girl object, we will pass all these qualities to her. We will be doing this to all objects (parts).

creating a class

this is how you create a class, by writing class and then class_name. By convention we write first letter of the class_name in UpperCase. Then we put : to end the code-

In [7]:

```
class Girl:
    def __init__(self,name,age):
        self.name=name
        self.age=age
```

OBSERVATIONS-

you see class Girl: than in new line we have __init___ method (two undersocre before init and two underscore after the init). Then we passed self,name,age and you can pass anything that will be common between all the girls.

what does init do, and why we pass self into it as an argument

init is an initializer that is automatically called. If your class is a bike then think that init is a start kick. It starts class. It means all the parts of this class will be able to acess the parameter/argument/inputs which are in it (or in the class, in this case name and age are those parameters)

Question arises, what the fuck self does then? Well all the parts/objects/instances of this class will access arguments/attributes/input of the class by ownself. That is why we pass self into the class. self means object/instance needs no additional persmisson to access the inputs/attributes of the main class (Girl, in this case).

self.name=name means each object can access name by ownself. self.age=age each object can access age by own self. That means we do not need to write additional codes for those.

we have creted the class now. Let's create a part of it. Part means object/instance. Creating an instance is known as instancialtion.

In [12]:

```
class Girl:
    def __init__(self,name,age):
        self.name=name
        self.age=age

    #now creating two instances/objects
girl1=Girl("Nisha", 24)
girl2=Girl("Riya", 25)
```

Looks, girl1 and girl2 are parts (objects/instance) of class Girl and in arguments/input I've to pass name and age (as we have passed in class __init___(self,name,age).

make sure to pass name and age to each object like we did for girl1 and girl2 because when we were writing the class we passed name and age init

how do we access girl1 and girl2

Let's find the name of girls:

In [13]:

```
girl1.name
```

Out[13]:

```
'Nisha'
```

In [14]:

```
girl2.name
```

Out[14]:

```
'Riya'
```

remember the dot notation, we use dot notation to get into one thing. girl1.name means, in girl1 we are looking for the name

let's find the age:

In [15]:

```
1 girl1.age
```

Out[15]:

```
24
```

In [16]:

```
girl2.age
```

Out[16]:

```
25
```

In [18]:

```
girl1.name, girl1.age
```

Out[18]:

```
('Nisha', 24)
```

In [19]:

```
girl2.name, girl2.age
```

Out[19]:

```
('Riya', 25)
```

methods in OOP

remember the functions we talked in lecture09? It makes your coding work easy so that you don't have to repeat the code again and again. Yes we call function as methods in OOP.

let's create a function that will tell that your girl is singing-

In [23]:

```
def singing():
    print("""Boy, you know I want your love
Your love was handmade for somebody like me
Come on now, follow my lead
I may be crazy, don't mind me
Say, girl;, let's not talk too much
Grab on my waist and put that body on me
Come on now, follow my lead
Come, come on now, follow my lead""")
```

let's print this function, by simply calling the fucntion. If you remember we call function by name

In [24]:

```
singing()
```

```
Boy, you know I want your love
Your love was handmade for somebody like me
Come on now, follow my lead
I may be crazy, don't mind me
Say, girl;, let's not talk too much
Grab on my waist and put that body on me
Come on now, follow my lead
Come, come on now, follow my lead
```

how do we put it into the class?

In [28]:

```
print("""oh pythonistas, I know you want it done.
You're looking somebody like me and bold.
Come on now, follow my code.
I may be silent but funny or both, don't mind me
Say, sir;, let's not talk too much
take my notes. and put the code in it before you gone
come on now, follow my code
come on now, follow my code""")
```

```
oh pythonistas, I know you want it done.
You're looking somebody like me and bold.
Come on now, follow my code.
I may be silent but funny or both, don't mind me
Say, sir;, let's not talk too much
take my notes. and put the code in it before you gone
come on now, follow my code
come on now, follow my code
```

ha ha ha that was funny, :-P. Let's put this into our class.

In [30]:

```
class Girl:
    def __init__(self,name,age):
        self.name=name
        self.age=age
    #making a function a method of this class. Like we used to do in function
    def singing(self):
        print("""Boy, you know I want your love
Your love was handmade for somebody like me
Come on now, follow my lead
I may be crazy, don't mind me
Say, girl;, let's not talk too much
Grab on my waist and put that body on me
Come on now, follow my lead
Come, come on now, follow my lead""")

girl1=Girl("Nisha",24)
girl2=Girl("Riya",25)
```

if you remember how we used to access things in class by using dot notation

In [31]:

```
girl1.name
```

Out[31]:

```
'Nisha'
```

let's call method, we just put into the class. Please note that argument of singing method/function is self so that it can access the class by itself

In [32]:

```
girl1.singing()
```

```
Boy, you know I want your love
Your love was handmade for somebody like me
Come on now, follow my lead
I may be crazy, don't mind me
Say, girl;, let's not talk too much
Grab on my waist and put that body on me
Come on now, follow my lead
Come, come on now, follow my lead
```

Let's take another example of class ¶

I will be creating a class of Car. We know that one thing will be common between all the cars is (maker,model,year). So common things those all cars will have, will be passed as attributes/inputs init.

In [33]:

```
class Car:
    def __init__(self,maker,model,year):
        self.maker=maker
        self.model=model
        self.year=year
```

let's create a method/function those cars will have- like starting, breacking, speeding

In [34]:

```
class Car:
    def __init__(self,maker,model,year):
        self.maker=maker
        self.model=model
        self.year=year
    #creating starting method/function
    def starting(self):
        print(self.maker,"is starting - GRRrrr GRrrrr ===3")
    #creating speeding method
    def speeding(self):
        print(self.maker,"is speeding- vrroom vrrrooom vrooom ppshhhhhhhhhh >>>>>")
    #creating breaking method
    def breaking(self):
        print(self.maker,"is now applying break khachhaaackkkkkkk grrrrrrrrr sssssh
```

see that we created class and then we created methods into that class. Methods are nothing new, they all are functions and they work as fucntions too. You just have to pass self into them as parameter so that it can access the class by it ownself. Are you getting me, good?

let's create 3 objects those will be cars and we have to pass three arguments maker,model,year

In [35]:

```
car1=Car("Audi","a4",2018)
car2=Car("BMW","c6", 2016)
car3=Car("maruti", "wagonR", 2015)
```

now access the name of maker, model and year of manufacturing

In [36]:

```
car1.maker, car1.model, car1.year
```

Out[36]:

```
('Audi', 'a4', 2018)
```

In [37]:

```
car2.maker, car2.model, car2.year
```

Out[37]:

```
('BMW', 'c6', 2016)
```

In [38]:

```
car3.maker, car3.model, car3.year
```

Out[38]:

```
('maruti', 'wagonR', 2015)
```

let's start all the cars

In [39]:

```
car1.starting()
```

```
Audi is starting - GRRrrr GRrrrr ===3
```

In [40]:

```
car2.starting()
```

```
BMW is starting - GRRrrr GRrrrr ===3
```

In [41]:

```
car3.starting()
```

```
maruti is starting - GRRrrr GRrrrr ===3
```

lets speed all the cars

In [42]:

```
car1.speeding()
```

```
Audi is speeding- vrroom vrrrooom vrooom ppshhhhhhhhhh >>>>>
```

In [43]:

```
car2.speeding()
```

```
BMW is speeding- vrroom vrrrooom vrooom ppshhhhhhhhhh >>>>>
```

In [44]:

```
car3.speeding()
```

```
maruti is speeding- vrroom vrrrooom vrooom ppshhhhhhhhhh >>>>>
```

let's apply breaks on all the cars

In [45]:

```
car1.breaking()
```

```
Audi is now applying break khachhaaackkkkkk grrrrrrrrr sssssh!!!!
```

In [46]:

```
car2.breaking()
```

```
BMW is now applying break khachhaaackkkkkk grrrrrrrrr sssssh!!!!
```

In [47]:

```
car3.breaking()
```

```
maruti is now applying break khachhaaackkkkkk grrrrrrrrr sssssh!!!!
```

I hope you guys found it helpful. It is just basics in OOP. I'll follow up with few more lectures in this topic. Keep pythoning and stay safe. It was pleasure to let you know the python the way I do. Have a great day everyone.

In [ ]:

```

```