<https://drive.google.com/file/d/10a2ZbFoudyQvm_uts-AaGi59JFqBSftP/view?usp=drive_link>

Following is the basic structure for creating a simple class and object.

----------------------------------------------------

// CLASS

Class student:

name = “ABC”

// Object

S1 = student()

Print(S1.name)

-----------------------------------------------------

Default name for all student object has become “ABC”.

**Constructor**

All classes have a function called \_\_init\_\_(), which is always executed when the object is being

initiated.

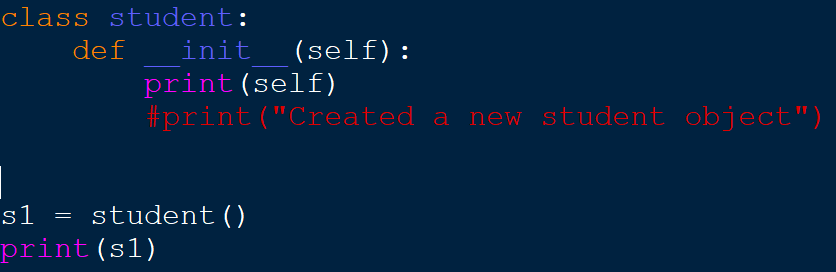
If not explicitly written python executes it’s default constructor.

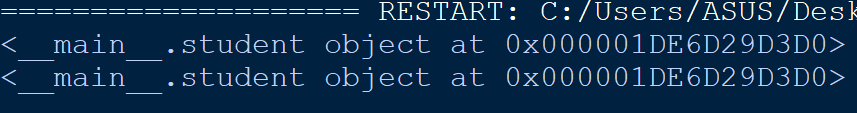
Class Student:

Def \_\_init\_\_(self):

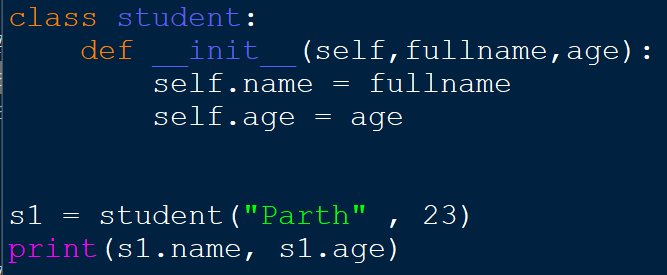
Print(“Object created”)

In constructor we pass an argument called self, self points to the object.





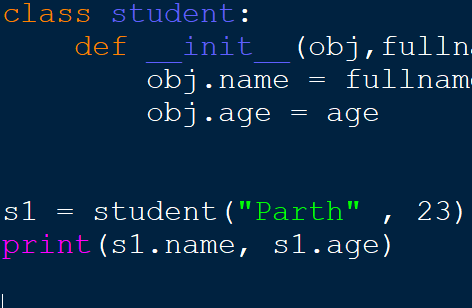
We can also pass other arguments in the constructor to provide values to its class variables or **attributes.**





We always need to pass self in constructor definition otherwise it will throw an error.

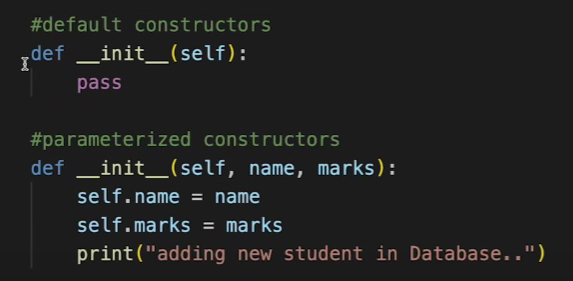
We can also use any other name instead of “self”. For e.g. below I have used *“obj”*. But usually self is used.



***The self parameter is a reference to the current instance of the class, and is used to access variables***

***that belongs to the class.***

We can have multiple constructors defined in our class. During object creation which constructor is called will be based on parameters provided.

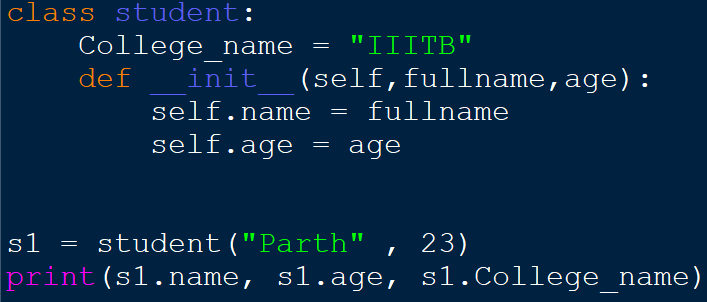


**Attributes**

There are 2 types of attributes Class and instance/object specific attributes.

Class attributes are those which are common to all instances of objects for e.g. COLLEGE NAME.

Object attributes are those which are unique or differ in objects e.g. STUDENT NAME.



Class attributes should be defined outside the constructor as every object will occupy space based on variables present in constructor definition. Now college\_name will occupy space only once and not everytime an object is created.

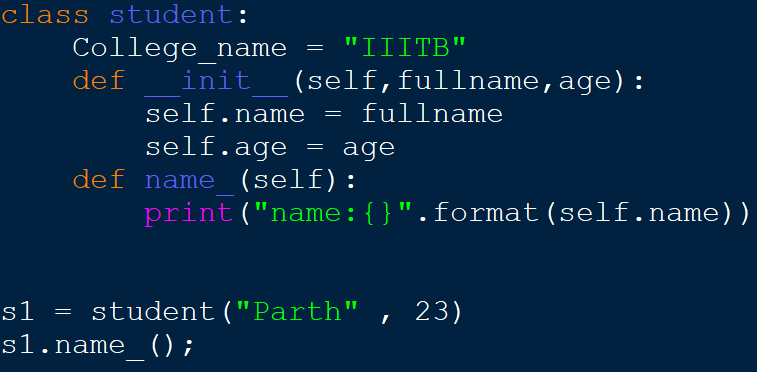
To access Class Attribute we can do it directly using class name too instead of using object. e.g. *student.College\_name.*

When we have class attribute and object attribute with same name then precedence of object attribute is greater than class attribute.

**METHODS**

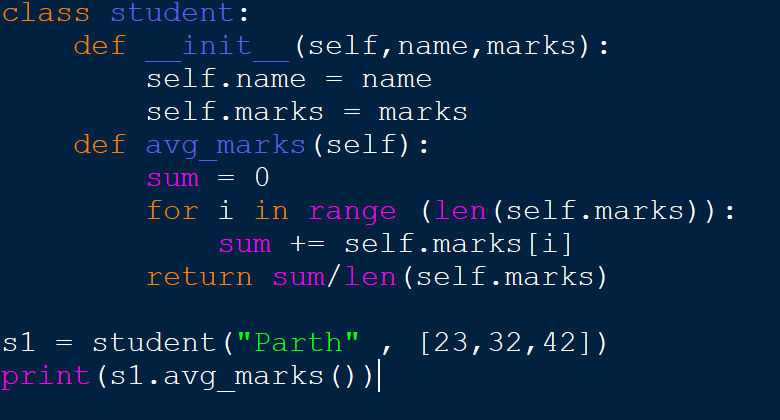
Functions that belong to object. For e.g. List class has SLICE() as a method.

First parameter of any method is self inside the class.



**Practice Question**

**Create a class Student passing name and list of marks as arguments. Define a method to return average marks.**

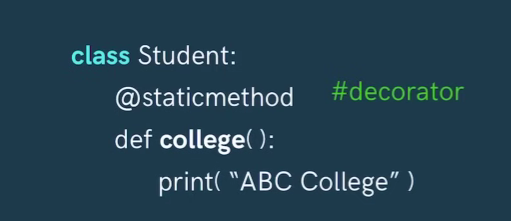
****

**Static Method**

Method that doesn’t use the self parameters ( work at class level )

Basically when a method doesn’t require object specific information we call it static method. If we don’t use self in normal method then it will throw an error so we need to write *@staticmethod* at the top of method so now when we define method without self it won’t give any error.

@Staticmethod is a decorator which wraps a function in order to extend its behaviour without permanently modifying it.



***del*** Keyword

Used to delete object property or object itself

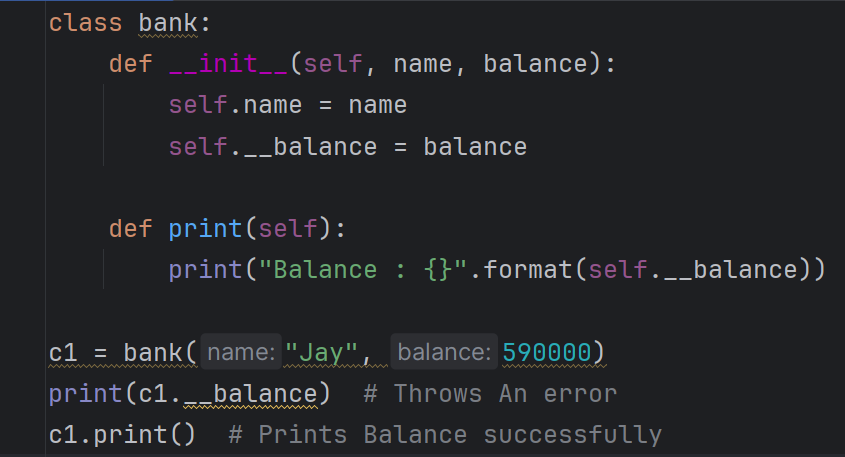
*del s1.name // To delete attribute*

*del s1 // To delete whole object*

Help in clearing up the space occupied by object or its attributes

***Private*** Attribute or Method

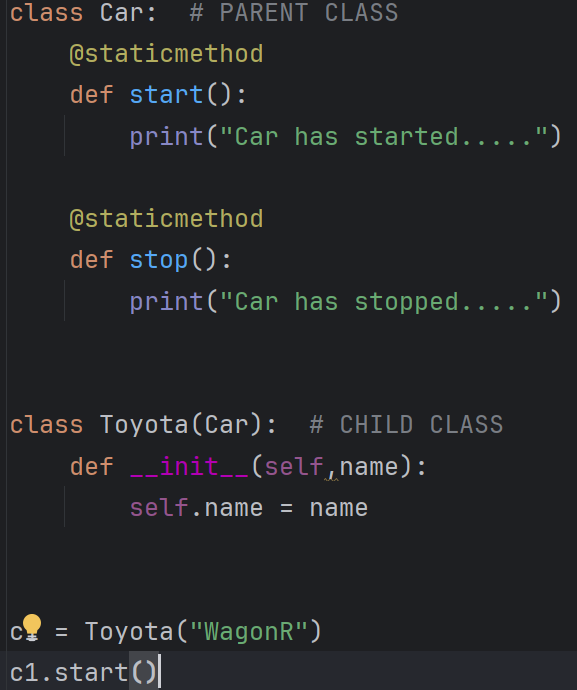
To make any attribute or method private i.e. not accessible outside the class we add “\_\_” ( 2 underscore) in front of attribute name or method name. Although they can be accessed using other methods of class.



Here we made balance attribute private by adding *‘\_ \_’*

***Inheritance***

When one class (child/derived) derives the properties & methods of another class (parent/base).

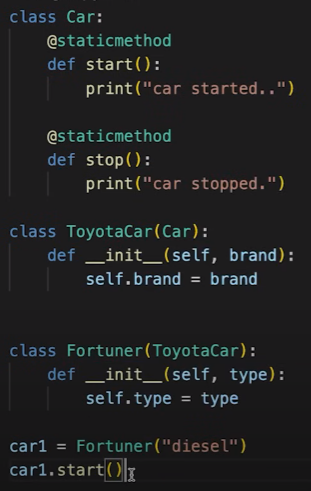


**Types of Inheritance**

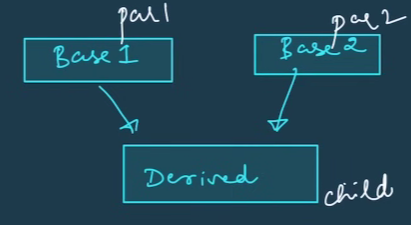
* Single level Inheritance (Above Example demonstrate single level inheritance)
* Multilevel Inheritance
* Multiple Inheritance

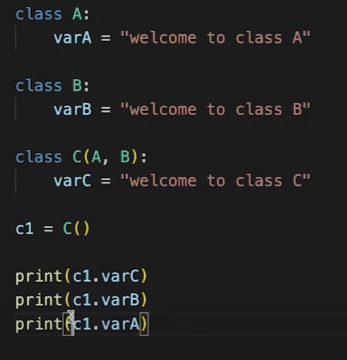
**Multiple inheritance**

*A -> B -> C ….*

**

**Multiple Inheritance**





**SUPER()**

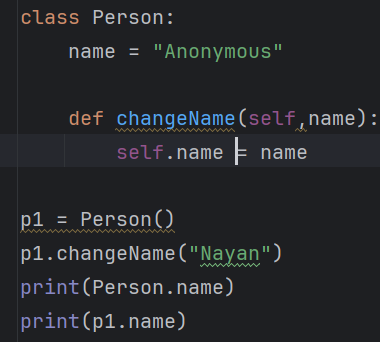
When we want to access attribute or method of parent class directly in child class definition then we use super().





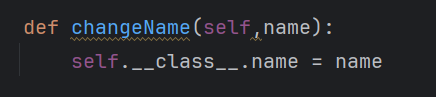
**Class Method**

Suppose I have provided a global *name* to aclass *Person.* Now I want to change the name not only of the object but for the whole class.

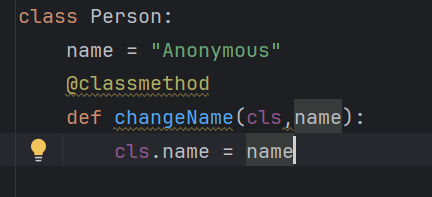
**

When I do this only name specific to object changes and for class it remains same.

So we could do something like this



Else We can use ***@classmethod***



In class method we pass *cls* as first argument which directly references to class and not the object.