**1. Python - public, private and protected:**

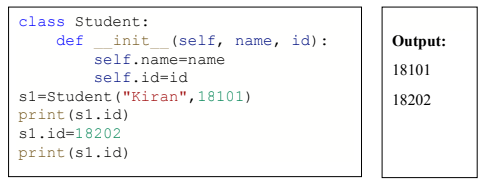
Object-oriented languages control the access to class resources by public, private and protected keywords but python doesn't have any mechanism that effectively restricts access to any instance variable or method.   
**Python prescribes a convention of prefixing the name of the variable/method with single or double underscore to emulate the behavior of protected and private access specifiers.**  
Before starting python encapsulation, let’s discuss how python handle public, private and protected.

**Members:-**

**1.1. Public member:**

Public members (generally methods declared in a class) are accessible from outside the class. The object of the same class is required to invoke a public method. All members in a Python class are public by default. Any member can be accessed from outside the class environment.   
Here, in the following example, variables can call outside of call and also can change the

variables.

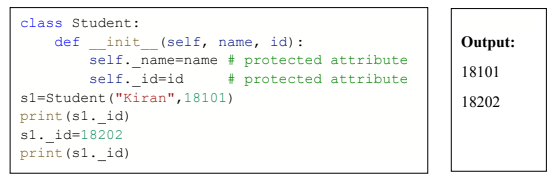


**1.2. Protected member:**

**Protected members of a class are accessible from within the class and are also available to its sub-classes. No other environment is permitted access to it.**  
This enables specific resources of the parent class to be inherited by the child class.

Python's convention to make an instance variable protected is to add a prefix “\_” (single underscore) to it. This effectively prevents it to be accessed, unless it is from within a sub-class. In fact, this doesn't prevent instance variables from accessing or modifying the instance.

**Here in following example, it can still perform the following operations and accessing and modifying instance variables prefixed with \_ from outside its class.**



**1.3. Private member:**

**Private members of a class are denied access from the environment outside the class. They can be handled only from within the class.**

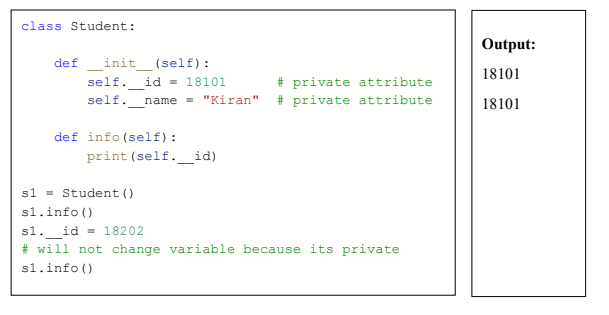
Private members are similar to protected members, the difference is that the class members declared private should neither be accessed outside the class nor by any base class. In Python, there is no existence of Private instance variables that cannot be accessed except inside a class.

However, to define a private member prefix the member name with double underscore “\_\_” (double underscore).

Here in following example, attribute cannot be change outside of the class as it is assigned as private attribute.

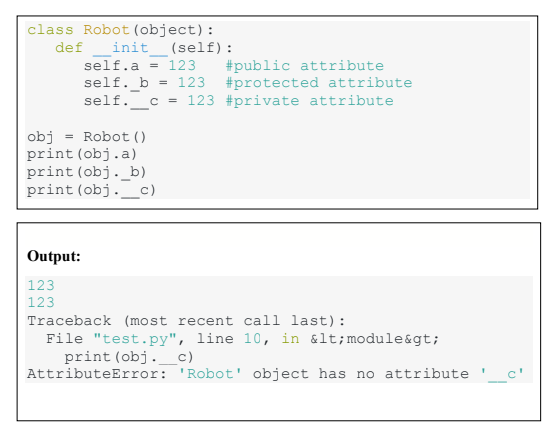
Also from the following example we can differentiate public, private and protected attributes.

Here public and protected attribute can access outside the class but in case of private attribute it shows an error.

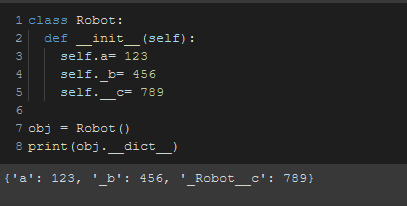


Also from the following example we can differentiate public, private and protected attributes.

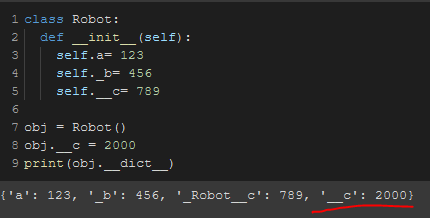
Here public and protected attribute can access outside the class but in case of private attribute it shows an error.



But it’s there, but the access is limited.



Even if we want to change a value, It create an entirely different instance and will save the value there.



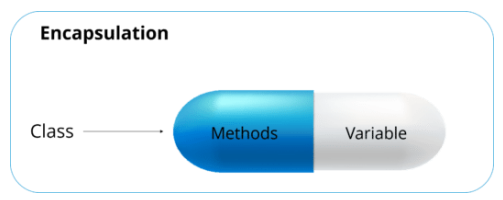
**Note:** for “c”, it’s showing **“\_Robot\_\_c”: 789** because the access is given to the user only and not outside of the class.

**Encapsulation**

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). **It describes the idea of wrapping data and the methods that work on data within one unit.**

This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data. To prevent accidental change, an object’s variable can only be changed by an object’s method. Those type of variables are known as private variable.

A class is an example of encapsulation as it encapsulates all the data that is member functions, variables, etc.



**Example:**

Consider a real-life example of encapsulation, in a company, there are different sections like the accounts section, finance section, sales section etc. The finance section handles all the financial

transactions and keeps records of all the data related to finance. Similarly, the sales section handles all the sales-related activities and keeps records of all the sales. Now there may arise a situation when for some reason an official from the finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of the sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is.

Here the data of the sales section and the employees that can manipulate them are wrapped under a single name “sales section”. As using encapsulation also hides the data.

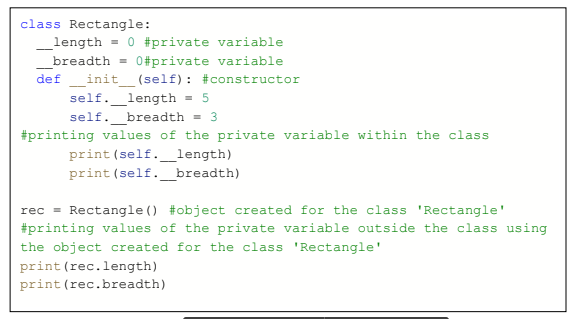
In this example, the data of any of the sections like sales, finance or accounts are hidden from any other section. So encapsulation provides security by hiding the data from the outside world.

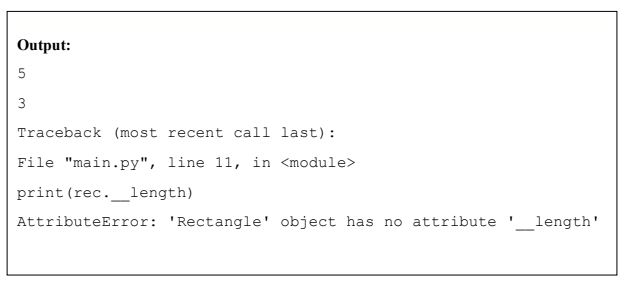
In Python, Encapsulation can be achieved by declaring the data members of a class either as private or protected. Let us see how access modifiers help in achieving Encapsulation.

**2.1 Encapsulation Using Private Members:**

If we declare any variable or method as private, then they can be accessed only within the class

in which they are defined. In the below example, 'length' and 'breadth' are the two private variables declared and can be accessed within the class 'Rectangle'.

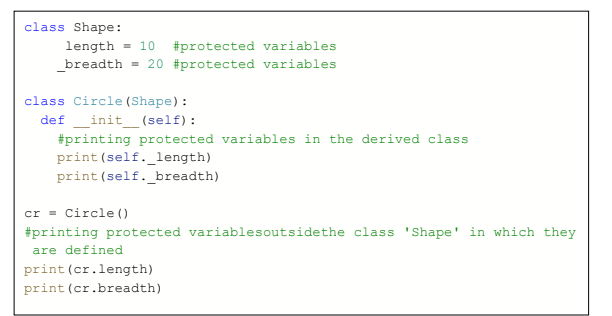


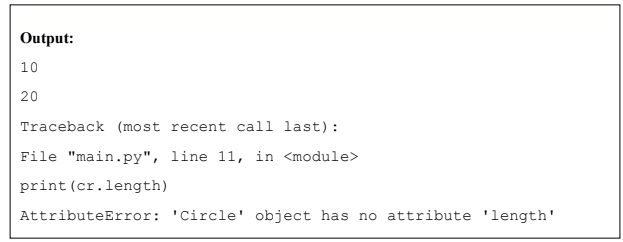


Since we have accessed private variables in the main() method, i.e., outside the class 'Rectangle', we got an error. Hence in the above program, Encapsulation is achieved using the private variables 'length' and 'breadth'.

**2.2 Encapsulation Using Protected Members:**

Protected members can be accessed within the class in which they are defined and also within the derived classes. In the below example, 'length' and 'breadth' are the two protected variables defined inside the class 'Shape'.

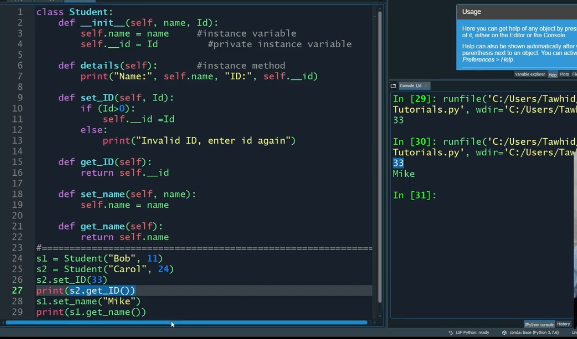




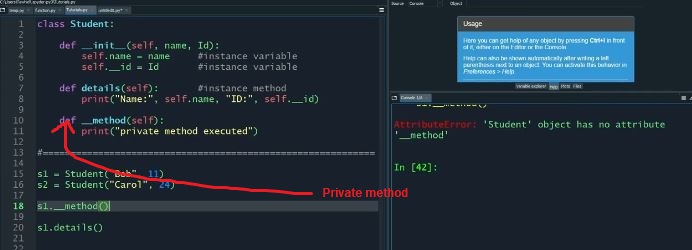
When we try to access protected variables in the derived class, we got output. But, in the main() method, we got an error. Hence in the above example, Encapsulation is achieved using the protected variables 'length' and 'breadth'.

**Get-set method:**

The get method **returns the value of the variable name** . The set method takes a parameter ( newName ) and assigns it to the name variable. The this keyword is used to refer to the current object.



# **Private method:**



We have to call this method from another method in that same class

