ADVANCED ASSIGNMENT NO 2

Nirbhay Ahir

[Nirbhay.ahir36@gmail.com](mailto:Nirbhay.ahir36@gmail.com)

Q1. What is the relationship between classes and modules?

Ans: Classes are the template/blueprint to create objects. Once a class is made, we can create multiple instances of that class. Every instance of the class will have access to the attributes and methods defined in the class.

Modules are the python files with .py extension in their names. These modules contain python codes. Functions and classes from one module can be imported to another module to use it.

In simple terms, we can consider a module to be the same as a code library or a file that contains a set of functions/Classes that we want to include in your application.

Q2. How do you make instances and classes?

Ans: A class instance is created by calling the class name and then inside the parenthesis we pass the arguments, which is accepted by the init method of the class.

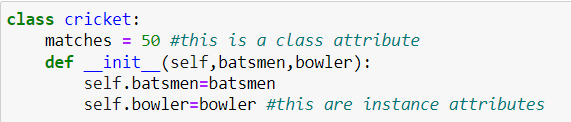
Example: cricket = player(virat, rohit). Here cricket is the instance, player is the class name, virat and rohit are the attributes.

We use the class keyword and then the class name and then put a colon to create a class.

Example: class cricket:. Here cricket is the class name.

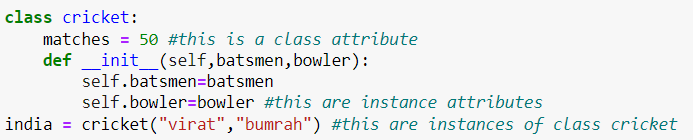
3. Where and how should be class attributes created?

Ans: Class attributes are those attributes that are shared by all the instances of the class. Class attributes should be created after the definition of the class name and outside all the class's methods.



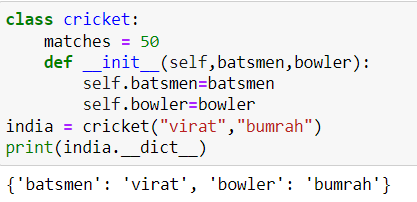
Q4. Where and how are instance attributes created?

Ans: Instance attributes are usually defined within the \_\_**init\_\_** method of the class. The values of these attributes are passed as an argument during the creation of the instance using class name. Instance attributes are not shared by the objects of the class.



Q5. What does the term "self" in a Python class mean?

Ans: self represents the instance of the class i.e. It refers to the object itself. By using the “self” keyword we can access the attributes and methods of the class within the class in python. It binds the attributes with the given arguments.

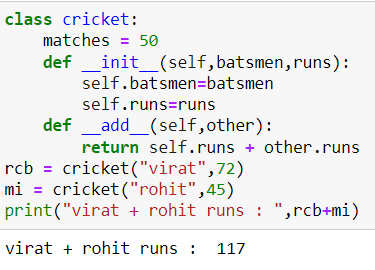


Q6. How does a Python class handle operator overloading?

Ans: Python class handles operator overloading using special methods called dunder or magic methods. These methods begin and end with \_\_(double underscore).

Example:

1. \_\_add\_\_: used to implement addition operator +.
2. \_\_sub\_\_: used to implement subtraction operator -.
3. \_\_mul\_\_: used to implement multiplication operator \*.
4. \_\_div\_\_: used to implement division operator /.



Q7. When do you consider allowing operator overloading of your classes?

Ans: We consider allowing operator overloading either when the use of the operator is not defined on the class objects or when we want to have a different meaning for the same operator depending on the context.

For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because ‘+’ operator is overloaded by int class and str class.

Q8. What is the most popular form of operator overloading?

Ans: The most popular form of operator overloading is by using special methods called dunder or magic methods. These methods begin and end with \_\_(double underscore).

Q9. What are the two most important concepts to grasp in order to comprehend Python OOP code?

Ans: The two most important concepts to grasp in order to comprehend Python OOP code are classes and objects. Apart from classes and objects few other important concepts of OOP are Inheritance, Polymorphism, Abstraction and Encapsulation.