Python Advanced Assignment 02

Q1. What is the relationship between classes and modules?

Ans: In object-oriented programming, classes and modules are both mechanisms for organizing and encapsulating code, but they serve different purposes and have different relationships. A class is a blueprint or template for creating objects. It defines the attributes (data) and behaviors (methods) of objects that belong to the class. On the other hand, a module is a file containing Python code, typically organized around a particular functionality or set of related functionalities. A module can contain classes, functions, variables, and other objects. In Python, modules can be imported and used in other modules or scripts to reuse code and avoid duplication. A class can also be defined within a module and then imported and used in other modules or scripts.

Q2. How do you make instances and classes?

Ans: A class can be made using keyword ‘class’

```
class Name_of_class:
        def __init__():
                pass
        def function():
                pass
```

instance of this class can be made using

```
object_name = Name_of_class()
```

calling a class function can be done using

```
object_name.function()
```

Q3. Where and how should be class attributes created?

Ans: Class attributes are variables that are associated with a class and are shared among all instances of that class. They are defined within the class definition but outside of any instance methods.

Q4. Where and how are instance attributes created?

Ans: Instance attributes are variables that are specific to an instance of a class. They are created and initialized when an instance of the class is created and are defined within the __init__ method of the class.

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

Ans: In Python, the self keyword is a convention that is used to refer to the instance of a class within the class itself. It is the first parameter of instance methods and is always named as ‘self’, although we can use any valid variable name instead of self.

Q6. How does a Python class handle operator overloading?

Ans: Python classes can handle operator overloading by defining special methods that correspond to the operator that we want to overload. These special methods are identified by their leading and trailing double underscores (__add__ for the + operator) and they are called automatically by Python when the corresponding operator is used on instances of the class.

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

Ans: When we are dealing with some special mathematical terminologies like complex numbers, matrices or convolution then they do not follow common operations of addition, multiplication etc, and so we allow operator overloading and define their operations accordingly.

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

Ans: The most common form of operator overloading is may be __add__ or __multiplication__.

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

Ans: The most comprehend concepts of OOP for it's good understanding are **Classes** and **Objects.** Once we get the idea of these two concepts then others are easy to understand.