Q1. What is the relationship between classes and modules?

Ans: Classes in python are templates for creating objects. They contain variables and functions which define the class objects. At the same time, modules are python programs that can be imported into another python program. Importing a module enables the usage of the module’s functions and variables into another program.

Q2. How do you make instances and classes?

Ans:

A class is a template for storing data and functions in an object. A class can be instantiated to create a unique instance. A class can accept parameters in the constructor.

**class** C:

**pass**

an\_instance = C()

**print**(an\_instance)

Q3. Where and how should be class attributes created?

Ans:

[Class attributes](https://www.geeksforgeeks.org/g-fact-34-class-or-static-variables-in-python/) belong to the class itself they will be shared by all the instances. Such attributes are defined in the class body parts usually at the top, for legibility.

class Student:

marks=50 #class attribute

def get\_marks(self):

return self.marks

Q4. Where and how are instance attributes created?

Ans: **Instance attributes** are attributes or properties attached to an instance of a class. Instance attributes are defined in the constructor.

class Student:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

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

Ans: self represents the instance of the class. By using the “self” we can access the attributes and methods of the class in python.

Q6. How does a Python class handle operator overloading?

Operators in Python work for **built-in** classes, like **int**, **str**, **list**, etc. But you can extend their **operability** such that they work on objects of user-defined classes too.

class bubble:

def \_\_init\_\_(self, volume):

self.volume = volume

def \_\_str\_\_(self):

return "volume is " + str(self.volume)

def \_\_add\_\_(self, other):

volume = self.volume + other.volume

return bubble(volume)

b1 = bubble(20)

b2 = bubble(30)

b3 = b1 + b2

print(b3)

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

Ans: Consider that we have two objects which are a physical representation of a class (user-defined data type) and we have to add two objects with binary ‘+’ operator it throws an error, because compiler don’t know how to add two objects. So we define a method for an operator and that process is called operator overloading. We can overload all existing operators but we can’t create a new operator. To perform operator overloading, Python provides some special function or magic function that is automatically invoked when it is associated with that particular operator. For example, when we use + operator, the magic method \_\_add\_\_ is automatically invoked in which the operation for + operator is defined.

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

Considering two items depicting a specific class, where you have to insert two objects using the binary ‘+’ operator. Perhaps it will show an error the compiler does not understand how to add. So, we describe an operator mechanism named overloading of the operator. Python includes a magic feature to conduct operator overloading which is immediately activated once paired with this same specific operator.

* For instance, when you are using the ‘+’ operator, the \_\_add\_\_ magical form can automatically describe the ‘+’ operator operation.
* With built-in sets, the Python operator functions well. But for different forms, operators behave accordingly. For example, in two numbers, the ‘+’ operator can apply addition, combine two lists, or merge multiple strings.
* Program to add without overloading the ‘+’ operator.

class circle:

    def \_\_init\_\_(object, radius):

        object.radius = radius

b1 = circle(10)

b2 = circle(20)

print(b1 + b2)

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

Python Inheritance

Inheritance enables us to define a class that takes all the functionality from a parent class and allows us to add more. In this tutorial, you will learn to use inheritance in Python.

It refers to defining a new [class](https://www.programiz.com/python-programming/class) with little or no modification to an existing class. The new class is called **derived (or child) class** and the one from which it inherits is called the **base (or parent) class**.

Python Inheritance Syntax

class BaseClass:

Body of base class

class DerivedClass(BaseClass):

Body of derived class

Derived class inherits features from the base class where new features can be added to it. This results in re-usability of code.

## Python Polymorphism

The literal meaning of polymorphism is the condition of occurrence in different forms.

Polymorphism is a very important concept in programming. It refers to the use of a single type entity (method, operator or object) to represent different types in different scenarios.

We know that the + operator is used extensively in Python programs. But, it does not have a single usage.

For integer data types, + operator is used to perform arithmetic addition operation

num1 = 1

num2 = 2

print(num1+num2)

Similarly, for string data types, + operator is used to perform concatenation.

str1 = "Python"

str2 = "Programming"

print(str1+" "+str2)

As a result, the above program outputs Python Programming.

Here, we can see that a single operator + has been used to carry out different operations for distinct data types. This is one of the most simple occurrences of polymorphism in Python.