**Q1. What is the relationship between classes and modules?**

Modules are collections of methods and constants. They cannot generate instances. Classes may generate instances (objects), and have per-instance state (instance variables). A class may inherit from another class, but not from a module. A module may not inherit from anything, Modules may be mixed in to classes and other modules., Classes, however, cannot be mixed in to anything.

Import Math ## Module.

Class c:pass

**Q2. How do you make instances and classes?**

Instance − An individual object of a certain class. An object fuel\_type that belongs to a class Car, for example, is an instance of the class Car

Instance variable − A variable that is defined inside a method and belongs only to the current instance of a class.

class Car:

def \_\_init\_\_(self,fuel\_type,model):

self,.ft = fuel\_type

self.mod = model

innova = Car(Petrol,2022)

innova.model // 2022

**Q3. Where and how should be class attributes created?**

[**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.  All object refer to single copy

class classattr:

    cnt = 0     # class attribute

def inc(self):

        classattr.cnt += 1

x = classattr()

x. inc()

print(x.cnt)

s1 = sampleclass()

s1.increase()

print(s1.count)

s2 = sampleclass()

s2.increase()

print(s2.count)

print(classattr.count)

1

2

2

**Instancr Attributes** are not shared by objects. Every object has its own copy of the instance attribute

class Car:

def \_\_init\_\_(self,fuel\_type,model):

self,.ft = fuel\_type

self.mod = model

innova = Car(Petrol,2022)

toyoto – Car(Diesel,2011)

innova.mod // 2022

toyoto.mod //2011

**Q4. Where and how are instance attributes created?**

Defined inside a constructor using the self parameter , Specific to object , Accessed using object dot notation e.g. object.instance\_attribute , Changing value of instance attribute will not be reflected to other objects.

class Car:

def \_\_init\_\_(self,fuel\_type,model):

self,.ft = fuel\_type

self.mod = model

innova = Car(Petrol,2022)

toyoto – Car(Diesel,2011)

innova.mod // 2022

toyoto.mod //2011

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

The self keyword is used to represent an instance (object) of the given class , Act as a pointer to assign value to the instance attributes and used to access variables that belongs to the class

**Q6. How does a Python class handle operator overloading?**

Python Will not know how to perform operations based on class Pointers , So Python Special functions we can use to perform the same tasks.

Ex :

|  |  |  |
| --- | --- | --- |
| Operator | Expression | Internally |
| Addition | p1 + p2 | p1.\_\_add\_\_(p2) |
| Subtraction | p1 - p2 | p1.\_\_sub\_\_(p2) |
| Multiplication | p1 \* p2 | p1.\_\_mul\_\_(p2) |

class Point:

def \_\_init\_\_(self, x=0, y=0):

self.x = x

self.y = y

p1 = Point(1, 2)

p2 = Point(2, 3)

print(p1+p2)

Error – Not able to add using this .

Instead we can use the below :

class Point:

def \_\_init\_\_(self, x=0, y=0):

self.x = x

self.y = y

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

return "({0},{1})".format(self.x, self.y)

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

x = self.x + other.x

y = self.y + other.y

return Point(x, y)

p1 = Point(1, 2)

p2 = Point(2, 3)

print(p1+p2)

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

When we need to perform operations based on coordinates of the Point object

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

--str-- , --add—

To print the value , to add data which points to different class objects

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

Inheritance , Polymorphism