Q1. What is the relationship between classes and modules?

The difference between a class and a module in python is that a class is used to define a blueprint for a given object, whereas a module is used to reuse a given piece of code inside another program.

A class can have its own instance, but a module cannot be instantiated. We use the ‘class’ keyword to define a class, whereas to use modules, we use the ‘import’ keyword. We can inherit a particular class and modify it using inheritance. But while using modules, it is simply a code containing variables, functions, and classes.

Modules are files present inside a package, whereas a class is used to encapsulate data and functions together inside the same unit.

Q2. How do you make instances and classes?

Class Employee:

company\_name='natwest'

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

self.name=name

self.salary=salary

here ‘Employee’ is a class and under which self.name and self.salary two instance object is created

Q3. Where and how should be class attributes created?

A class attribute is a Python variable that belongs to a class rather than a particular object. It’s shared between all the objects of this class and is defined outside the constructor function, \_\_init\_\_(self,...), of the class.

class ExampleClass(object):

class\_attr = 0

def \_\_init\_\_(self, instance\_attr):

self.instance\_attr = instance\_attr

class\_attr is a class attribute

Q4. Where and how are instance attributes created?

instance attributes are defined in the constructor. Defined directly inside a class. Defined inside a constructor using the self parameter.

class ExampleClass(object):

class\_attr = 0

def \_\_init\_\_(self, instance\_attr):

self.instance\_attr = instance\_attr

instance\_attr is an instance attribute defined inside the constructor.

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

SELF represents the instance of class. This handy keyword allows you to access variables, attributes, and methods of a defined class in Python.

The self parameter doesn’t have to be named “self,” as you can call it by any other name. However, the self parameter must always be the first parameter of any class function, regardless of the name chosen. So instead of self, you could use “Mine” or “Ours” or anything else.

Q6. How does a Python class handle operator overloading?

The operator overloading in Python means provide extended meaning beyond their predefined operational meaning. Such as, we use the "+" operator for adding two integers as well as joining two strings or merging two lists. We can achieve this as the "+" operator is overloaded by the "int" class and "str" class.

print (14 + 32)

# Now, we will concatenate the two strings

print ("Java" + "Tpoint")

# We will check the product of two numbers

print (23 \* 14)

# Here, we will try to repeat the String

print ("X Y Z " \* 3)

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

By overloading standard operators on a class, you can exploit the intuition of the users of that class. This lets users program in the language of the problem domain rather than in the language of the machine.

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

The most frequent instance is the adding up operator '+'

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

Both inheritance and polymorphism are fundamental concepts of object-oriented programming. These concepts help us to create code that can be extended and easily maintainable