Q1. What is the relationship between classes and modules?

***Ans:***

modules is a file containing python definitions and statements that can be used in python files. Modules can define class, function and variables. Module is a container for classes and other objects. You can import module use its class in your own code.

On other hand, class in blueprint for creating objects that have specific properties and behaviours. Class are defined within the modules.

Q2. How do you make instances and classes?

***Ans:***  we define the classes using the “class” keyword, and we create instance by using the class name followed by parentheses.

So here is the example,

class cat:

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

self.name = name

self.age = age

def voice(self):

print(“meowww”)

my\_cat = cat(“john”, 2)

so here in above example we define class as cat which has two attributes “name” and “age”

and one method called “voice”.

So here we have created the instance my\_cat with name “john” and age “2”.

Q3. Where and how should be class attributes created?

***Ans:***

class cat:

age = “2”

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

self.name = name

def voice(self):

print(“meowww”)

my\_cat = cat(“john”)

here in the above example {age = “2”} is a class attribute. Which can be accessed by all the instance of the class becoz they defined at class level, outside of the any method. Class attribute can be useful for defining default values that are shared by all instances of the class.

Q4. Where and how are instance attributes created?

***Ans:***

class cat:

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

self.name = name

self.age = age

def voice(self):

print(“meowww”)

my\_cat = cat(“john”, 2)

instance attributes are attributes that are specific to each instance of a class. And they are defined within the “\_\_init\_\_” method of the given class.

Here is self.name and self.age are two instance attributes defined in \_\_init\_\_method of class cat. Instance attributes are specific to each instance of the class, which means that each instance can have its own values for those attributes.

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

***Ans:***

The ‘self’ is used to refer to the instance of the class. When you create an object or instance of a class in python, you can access its attributes and method using ‘self’ keyword.

class cat:

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

self.name = name

self.age = age

def introduce(self):

print(f “hello, my name is{self.name} and my age is{self.age} years old.”)

my\_cat = cat(“john”, 2)

so in the above example , the attributes name and age are assigned to instance using ‘self.name’ and ‘self.age’.

the method “introduce” in the class cat takes no parameter except “self” which part of the instance of the class. And hence name and age can be accessed using self.name and self.age

Q6. How does a Python class handle operator overloading?

***Ans:***

python classes can define their own custom behaviour for these operators by implementing special methods with special names. This is methods knows as magic methods or dunder methods.

For example to overload addition operator (+), a class can define a method called add(self, other), where self is the instance of the class and other is object being added and add method will return the addition.

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

***Ans:***

1. if the class have arithmetic or comparison operations defined.
2. If the operator overloading, make the code more readable and easier to understand.
3. If it introduces unexpected behaviours or side effects.
4. It may not be consistent with the behaviour of other built-in types.

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

***Ans:***

the arithmetic operator overloading is the most popular form of overloading. Which involves operator overloading such as +, -, \*, /.

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

***Ans:***

1. Classes: a class is blueprint for creating objects that define the properties and behaviours of the objects. It consist of data(attribute) and function(methods). Classes are building blocks of objects.

2. objects: an object is an instance of a class. It is created using the class as a blueprint and has its own set of data i.e., attributes and behaviours i.e., methods that are specific to that instance.