Q1. Define the relationship between a class and its instances. Is it a one-to-one or a one-to-many partnership, for example?

Q2. What kind of data is held only in an instance?

Q3. What kind of knowledge is stored in a class?

Q4. What exactly is a method, and how is it different from a regular function?

Q5. Is inheritance supported in Python, and if so, what is the syntax?

Q6. How much encapsulation (making instance or class variables private) does Python support?

Q7. How do you distinguish between a class variable and an instance variable?

Q8. When, if ever, can self be included in a class's method definitions?

Q9. What is the difference between the \_ \_add\_ \_ and the \_ \_radd\_ \_ methods?

Q10. When is it necessary to use a reflection method? When do you not need it, even though you support the operation in question?

Q11. What is the \_ \_iadd\_ \_ method called?

Q12. Is the \_ \_init\_ \_ method inherited by subclasses? What do you do if you need to customize its behavior within a subclass?

Solution: 1

Relationship between a class and its instances is a one to many partnership.

Solution: 2

Instance objects contains the Instance variables which are specific to that specific Instance object.

Solution: 3

Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

Solution: 4

The methods with a class can be used to access the insatnce variables of its instance. So,the object's state can be modified by its method. Function can't access the attributes of an instance of a class or can't modify the state of the object.

Solution: 5

Yes,Python supports inheritance. The Types of Inheritence Supported by Python are:

1. Simple Inheritence
2. Multiple Inheritence
3. Multilevel lInheritence
4. Hybrid Inheritence
5. Hierracial Inheritence

Solution: 6

Encapsulation describes the idea of wrapping data and the methods that work on data within one unit. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data. To prevent accidental change, an objects variable can only be changed by an objects method.

Solution: 7

The **Class Attribute** is available to all the instance objects of that class. whereas **Instance Attributes** are accessible only to the object or Instance of that class.

A single copy of Class attributes is maintained by pvm at the class level. Whereas difference copies of instance attributes are maintained by pvm at objects/instance level.

Solution: 8

 Yes, self can included in class method definations to access the instance variables inside class methods.

Solution: 9

Entering **\_\_radd\_\_** Python will first try **\_\_add\_\_()**, and if that returns Not Implemented Python will check if the right-hand operand implements **\_\_radd\_\_**, and if it does, it will call **\_\_radd\_\_()** rather than raising a **TypeError**

**Solution: 10**

Reflection method we often encounter the requirement that a method in the executing object, or a variable in the calling object, or a field of the object should be assigned, while the method name or field name can not be determined when encoding the code, and need to be input in the form of passing strings through parameters.

Solution: 11

**\_\_iadd\_\_** method is called when we use implementation like a+=b which is **a.\_\_iadd\_\_(b)**

**Solution: 12**

Yes, **\_\_init\_\_** method will be inherited by subclasses. if we want to customize its behaviour within a subclass we can use **super()** method.