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

Answer:

Class is abstraction of a real world entity. It consists of attributes and methods. It is a blueprint to create objects of similar types. Instance is an object of a class. Relationship between a class and its instances is a one to many partnership.

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

Answer:

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

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

Answer:

A class holds the code or the logic to define each instance of the class as well as the information about its methods. It also holds the data of class attributes. It 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. It is like a blueprint for an object.

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

Answer:

The methods with a class can be used to access the instance 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.

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

Answer:

Python language supports inheritance.

Python Inheritance Syntax

---------------------------

Class BaseClass:

{Body}

Class DerivedClass(BaseClass):

{Body}

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

Answer:

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.

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

Answer:

The class attribute is available to all the instance objects of that class. The instance variable is accessible only to the object or instance of that class.

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

Answer:

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

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

Answer:

\_\_add\_\_ stands for addition and \_\_radd\_\_ stands for right-addition. For example, given x + y, if x.\_\_add\_\_(y) returns unimplemented, y.\_\_radd\_\_(x) is attempted instead.

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

Answer:

Reflection method frequently 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 cannot be determined when encoding the code, and need to be input in the form of passing strings through parameters.

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

Answer:

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

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

Answer:

\_\_init\_\_ method inherited by subclasses while it can be overloaded. To customize its behaviour within a subclass we can use super() method.