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

Instances are objects of a class in Python. In other words, users can define an instance of a particular class as an individual object. Users can define the Instance methods inside a Python class, similar to how they define a regular function. First, users use the "def" keyword to define an instance method.

It is a one-to-many partnership.

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

Instance − An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle. Instantiation − The creation of an instance of a class. Method − A special kind of function that is defined in a class definition.

An instance of a class is an object. It is also known as a class object or class instance. As such, instantiation may be referred to as construction. Whenever values vary from one object to another, they are called instance variables. These variables are specific to a particular instance.

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

Classes commonly contains data field to store the data and methods for defining behaviors. Also every class in python contains a special method called initializer (also commonly known as constructors), which get invoked automatically every time new object is created.

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

The method operates the data in the class, while a function is used to return or pass the data. A function can be directly called by its name, while a method can't be called by its name. The method lies under Object-Oriented Programming, while a function is an independent functionality.

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

One of the core concepts in object-oriented programming (OOP) languages is inheritance. It is a mechanism that allows you to create a hierarchy of classes that share a set of properties and methods by deriving a class from another class. Inheritance is the capability of one class to derive or inherit the properties from another class.

Python Inheritance Syntax

Class BaseClass:

{Body}

Class DerivedClass(BaseClass):

{Body}

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

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It 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 object’s variable can only be changed by an object’s method. Those types of variables are known as private variables.

A class is an example of encapsulation as it encapsulates all the data that is member functions, variables, etc. The goal of information hiding is to ensure that an object’s state is always valid by controlling access to attributes that are hidden from the outside world.

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

A class variable is a variable that defines a particular property or attribute for a class. An instance variable is a variable whose value is specified to the Instance and shared among different instances. We can share these variables between class and its subclasses. We cannot share these variables between classes.

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

Self is the name variable of the MyName class. So if there is a variable within a method, SELF won't work. So SELF should be used when you define an instance method, as it's passed automatically as the first parameter when the method is called

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

\_\_add\_\_(y) and x. \_\_radd\_\_(y) is that the former calculates x + y whereas the latter calculates y + x — both calling the respective method defined on the object x .

Python \_\_add\_\_() function is one of the magic methods in Python that returns a new object(third) i.e. the addition of the other two objects. It implements the addition operator “+” in Python.

The radd() method adds each value in the DataFrame with a specified value. This method is called reverse add, and is similar to the add() method, but instead of calculating 100 + 15 it calculates 15 + 100 , which in this case will give the same result.

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

Reflection refers to the ability for code to be able to examine attributes about objects that might be passed as parameters to a function. For example, if we write type(obj) then Python will return an object which represents the type of obj.

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

The Python \_\_iadd\_\_() magic method implements in-place addition x += y that adds together the operands and assigns the result to the left operand. This operation is also called augmented arithmetic assignment. The method simply returns the new value to be assigned to the first operand.

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

The \_\_init\_\_() function is called automatically every time the class is being used to create a new object.

A subclass “inherits” all the attributes (methods, etc) of the parent class. This means that a subclass will have everything that its “parents” have. You can then change (“override”) some or all of the attributes to change the behavior.

An inherited class is not required to call the \_\_init\_\_() method of the parent class. If no \_\_init\_\_() method is implemented in the inherited class, then the parent \_\_init\_\_() will be called automatically when an object of the inherited class is created.