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

Everything in Python is an object such as integers, lists, dictionaries, functions and so on. Every object has a type and the object types are created using classes. Instance is an object that belongs to a class. For instance, list is a class in Python. When we create a list, we have an instance of the list class

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

Instance variables are owned by instances of the class. This means that for each object or instance of a class, the instance variables are different. Unlike class variables, instance variables are defined within methods.

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?

A method, like a function, is a set of instructions that perform a task. The difference is that a method is associated with an object, while a function is not.

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

Unlike other object-oriented programming languages like Java, Python supports all types of inheritance, even multiple inheritance

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?

Class variables are defined within the class but outside of any class methods. Instance variables are defined within class methods, typically the constructor. Changes made to the class variable affect all instances. Changes made to the instance variable does not affect all instances.

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

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. You should also use self in Python when referencing a class attribute from inside an instance method.

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

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.

Using reflection, we can write one recursive reverse function that will work for strings, lists, and any other sequence that supports slicing and concatenation. If an obj is a reference to a string, then Python will return the str type object. Further, if we write str() we get a string which is the empty string. In other words, writing str() is the same thing as writing “”. Likewise, writing list() is the same thing as writing [].

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.

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

Inheritance is a concept in object-oriented programming where a child class (or subclass) derives attributes and behaviors from a parent or sibling class.

This eliminates the need to implement the methods inherited by a subclass, or child class, again. In other words, it enables a child class to inherit/reuse the attributes and methods of a parent class. In terms of real-world objects, it represents an IS-A relationship