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

ANS:

A class defines object properties including a valid range of values, and a default value. A class also describes object behavior. An object is a member or an "instance" of a class. An object has a state in which all of its properties have values that you either explicitly define or that are defined by default settings.

A class defines object properties including a valid range of values, and a default value. A class also describes object behavior. An object is a member or an "instance" of a class. An object has a state in which all of its properties have values that you either explicitly define or that are defined by default settings.

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

ANS:

In [class-based](https://en.wikipedia.org/wiki/Class-based), [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming), an instance variable is a [variable](https://en.wikipedia.org/wiki/Variable_(programming)) defined in a [class](https://en.wikipedia.org/wiki/Class_(computer_programming)) (i.e., a [member variable](https://en.wikipedia.org/wiki/Member_variable)), for which each instantiated [object](https://en.wikipedia.org/wiki/Object_(computer_science)) of the class has a separate copy, or [instance](https://en.wikipedia.org/wiki/Instance_(computer_science)). An instance variable has similarities with a [class variable](https://en.wikipedia.org/wiki/Class_variable),[[1]](https://en.wikipedia.org/wiki/Instance_variable" \l "cite_note-1) but is non-[static](https://en.wikipedia.org/wiki/Static_variable). An instance variable is a variable which is declared in a class but outside of [constructors](https://en.wikipedia.org/wiki/Constructor_(object-oriented_programming)), [methods](https://en.wikipedia.org/wiki/Method_(computer_programming)), or [blocks](https://en.wikipedia.org/wiki/Block_(programming)). Instance variables are created when an object is instantiated, and are accessible to all the constructors, methods, or blocks in the class. [Access modifiers](https://en.wikipedia.org/wiki/Access_modifiers) can be given to the instance variable.

An instance variable is not a [class variable](https://en.wikipedia.org/wiki/Class_variable), although there are similarities. It is a type of [class attribute](https://en.wikipedia.org/wiki/Class_(computer_science)#Structure) (or class property, [field](https://en.wikipedia.org/wiki/Field_(computer_science)), or data member). The same dichotomy between instance and class members applies to methods ("member functions") as well; a class may have both [instance methods](https://en.wikipedia.org/wiki/Instance_method) and [class methods](https://en.wikipedia.org/wiki/Class_method).

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?

ANS:

### Explicit Knowledge

It is the most basic form of knowledge and is easy to pass along as it’s written down and very accessible. Explicit knowledge is called when data is processed, organized, structured, and interpreted. Explicit knowledge can be easily articulated, recorded, communicated, and stored, most notably in knowledge management.

For example, open a knowledge management platform and take a look around. The company data sheets, white papers, research reports, etc., are all explicit company knowledge.

Explicit knowledge consists of information like:

* SOPs
* Instruction manuals
* Guides

### Implicit Knowledge

Implicit knowledge is the practical application of explicit knowledge. There are likely instances of implicit knowledge all around the organization.

For example, let’s say, consider asking a team member how to perform a task. This could ignite a conversation about the range of options to perform the task and potential outcomes, leading to a thoughtful process to get the best course of action. The team member’s implicit knowledge educates the conversation of how to do something and what can happen further.

### Tacit Knowledge

Tacit knowledge is the knowledge that we procure from personal experience and context. It’s information that, if any time, would be very difficult to note down, articulate, or show in a tangible form.

For example, think of learning how to make your mother’s famous recipes. Indeed, she gave you the recipe book, but when you execute it on your own, you feel something is missing. After many years of experience, she has learned the right feel for the dough, or precisely how long something should be in the oven. It’s not the points she can write down; she can feel it.

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

ANS:

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?

ANS:

 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?

ANS:

Encapsulation is one of the key concepts of object-oriented languages like Python, Java, etc. Encapsulation is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

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

ANS:

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?

ANS:

In object-oriented programming, whenever we define methods for a class, we use self as the first parameter in each case. Let's look at the definition of a class called Cat . In this case all the methods, including \_\_init\_\_ , have the first parameter as self . We know that class is a blueprint for the objects.

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

ANS:

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?

ANS:

Reflection is a feature in the Java programming language. It allows an executing Java program to examine or "introspect" upon itself, and manipulate internal properties of the program. For example, it's possible for a Java class to obtain the names of all its members and display them.

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

ANS:

Syntax. 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?

ANS:

The \_\_init\_\_ method is the Python equivalent of the C++ constructor in an object-oriented approach. The \_\_init\_\_ function is called every time an object is created from a class. The \_\_init\_\_ method lets the class initialize the object's attributes and serves no other purpose. It is only used within classes.

The “\_\_init\_\_” is a reserved method in python classes. It is known as a constructor in Object-Oriented terminology. This method when called, allows the class to initialize the attributes of the class. Python super() The super() function allows us to avoid using the base class name explicitly.