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&#39;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?

A1. In object-oriented programming, a class defines the blueprint or template for creating objects, while instances are specific objects created from that class. The relationship between a class and its instances is a one-to-many partnership, meaning that a single class can be used to create multiple instances.

A2. Instance variables hold data that is specific to each instance of a class. They are defined in the constructor method (**init**()) and can be accessed and modified using dot notation.

A3. A class stores knowledge in the form of class variables, which are shared by all instances of the class, and methods, which define the behavior of the class.

A4. A method is a function that is defined within a class and operates on its instances. It takes the instance itself as the first parameter, conventionally called "self". A regular function, on the other hand, is not defined within a class and operates on its input parameters.

A5. Yes, inheritance is supported in Python. The syntax for inheritance is to include the parent class in parentheses after the subclass name in the class definition. For example: class SubClass(ParentClass):

A6. Python supports encapsulation to some extent, but it does not have built-in support for making instance or class variables private. However, it is convention to prefix instance or class variables with an underscore (_) to indicate that they should not be accessed from outside the class.

A7. A class variable is a variable that is shared by all instances of the class, whereas an instance variable is specific to each instance. Class variables are defined within the class but outside of any method, while instance variables are defined within the constructor method.

A8. The self parameter is included in all class's method definitions, as it refers to the instance that is calling the method. The self parameter allows the method to access and modify instance variables and call other instance methods.

A9. The **add** method is called when the + operator is used to add two objects together, while the **radd** method is called when the object is on the right-hand side of the + operator. The **radd** method is used when the left-hand side object does not support addition with the right-hand side object directly.

A10. A reflection method is used to inspect or manipulate the attributes and methods of an object at runtime. It is necessary to use a reflection method when the specific attributes or methods of an object are not known beforehand. Reflection methods are not needed when the attributes or methods of an object are already known and can be accessed directly.

A11. The **iadd** method is called when the += operator is used to add an object to itself in place. It is a shorthand for x = x + y, where x is the object and y is the other operand.

A12. The **init** method is inherited by subclasses, but it can be overridden to customize its behavior within the subclass. To customize the **init** method within a subclass, the subclass should define its own **init** method with any additional parameters or functionality. The subclass can then call the parent class's **init** method using super().