Q1. Relationship Between a Class and Its Instances:

- It's a one-to-many relationship. A class serves as a blueprint, and instances are created based on this blueprint.

Q2. Data Held Only in an Instance:

- Instance-specific data is stored in instance variables.

Q3. Knowledge Stored in a Class:

- Class holds shared data (class variables) and behaviours (methods) common to all instances.

Q4. Method vs Regular Function:

- A method is a function associated with an object (instance or class), while a regular function is independent.

Q5. Inheritance in Python:

- Yes, Python supports inheritance.

- Syntax: `class DerivedClass(BaseClass):`

Q6. Encapsulation Support in Python:

- Limited support for encapsulation.

- Convention: Prefixing instance variables with a single underscore for a degree of privacy.

Q7. Class Variable vs Instance Variable:

- Class Variable: Shared by all instances of a class.

- Instance Variable: Specific to each instance.

Q8. self in Class's Method Definitions:

- It's a reference to the instance itself and is always the first parameter in a method definition.

- It's included in every method to access instance variables and other methods.

Q9. Difference Between \_ \_add\_ \_ and \_ \_radd\_ \_ Methods:

- \_ \_add\_ \_: Invoked for addition (e.g., `a + b`).

- \_ \_radd\_ \_: Invoked for right-side addition (e.g., `b + a`) if \_ \_add\_ \_ is not defined.

Q10. Necessity of Reflection Method:

- Reflection methods (e.g., \_ \_getattr\_ \_) are used for attribute access customization.

- Not needed when standard attribute access suffices.

Q11. \_ \_iadd\_ \_ Method:

- The in-place addition method, called when using the `+=` operator.

Q12. \_ \_init\_ \_ Inheritance and Customization:

- Inheritance: It's inherited by subclasses.

- Customization: Subclasses can override \_ \_init\_ \_ to customize its behaviour.