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 is a blueprint that is needed to make an object(= instance). The difference between an object and an instance is, an object is a thing and an instance is a relation. In other words, instance describes the relation of an object to the class that the object was made from. Relationship between a class and its instances is a one to many partnership.

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

Ans. Instance objects contains the Instance variables which are specific to that specific Instance object.

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

Ans. A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made.

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

Ans. A function can pass the data that is operated and may return the data. The method operates the data contained in a Class. Data passed to a function is explicit. A method implicitly passes the object on which it was called. 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. Inheritance allows us to define a class that inherits all the methods and properties from another class. Parent class is the class being inherited from, also called base class. Child class is the class that inherits from another class, also called derived class.

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

Ans. Encapsulation 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 objects variable can only be changed by an objects method.

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

Ans. Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block. Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.

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

Ans. Yes, self can included in class method definations to access the instance variables inside class methods.

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

Ans. Entering \_\_radd\_\_ Python will first try \_\_add\_\_(), and if that returns Not Implemented Python will check if the right-hand operand implements \_\_radd\_\_, and if it does, it will call \_\_radd\_\_() rather than raising a TypeError

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 method we often encounter the requirement that a method in the executing object, or a variable in the calling object, or a field of the object should be assigned, while the method name or field name can not be determined when encoding the code, and need to be input in the form of passing strings through parameters.

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

Ans. \_\_iadd\_\_ method is called when we use implementation like a+=b which is a.\_\_iadd\_\_(b)

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

Ans. Yes, \_\_init\_\_ method will be inherited by subclasses. if we want to customize its behaviour within a subclass we can use super() method.