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 Class is an abstraction of a real-world entity. It consists of attributes and methods.

Instance is an object of a class. It one to many relationships between class and its instances.

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

Ans It stores the values of the class attributes. When we give the attributes in the **init**function as an argument then the values which are stored in the object will get passed to it and attributes gets initialized and later on init function passed these values to the class

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

Ans Class creates a user-defined data structure, which holds its data members and member functions,

which can be accessed and used by creating an instance of that class.

A class is like a blueprint for an object.

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

Ans A method is refer to the function only but a difference is that, a method can be perform inside the class by using the class object. It cannot be perform outside the class.

While a standard function is a function which can be accessed outside any where. Working of both the method and std function is same but there is a difference in accessibility.

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

Ans Python do support inheritance in Oops concept. It is possible in python to inherit the behavior and properties of some other class to avoid writing the same code again and again when it is required. We can even do multiple inheritances. Meaning we can inherit the property of more than one class into a single class also

syntax

class A:

pass

class B(A):

pass

class C(B):

pass

class D(B,A):

pass

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

Ans Encapsulation prevents from accessing accidentally, but not intentionally.The private attributes and methods are not hidden. The private attributes can be accessed within the object method.

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

Ans **Class variable** It is a variable that is declared below the class and outside the **init** function, whereas, **instance variable** are declared inside the init function.

**Class variable** need a class name to get called as well as it can be called by the class instance name also, whereas a **instance variable** can be called by an object name but not with the class name.

**Class variable\*** is Global to all the objects of a class means it is shared across every object, while, instance variable is only for the unique data values which are stored by the class instance.

**Class variable** could not override the instance variable but instance variable can override the class variable

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

Ans First of all self is nothing but an object pointer which is pointing to object itself. We need an object namespace to call the func/method which is define inside the class. So that is why there is a pointer 'self' which is refering to an class object pointer. With this **self.method** we can use the method

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

Ans **add**is simply an built method which add the two items when it is passed, whose class is having this method. Like integer and string class have this method and in case of integer this method will add up the two values and in case of string class this will concatenate the two items.

**radd**This is inside the integer class but not in string class. This also add up the two number. But the the difference is, **radd**is only called if the left object does not have an **add** method, or that method does not know how to add the two objects. If both the classes have **add**method then this will not called.

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 facility where you can query an object about its attributes at runtime. Meaning, we can modify the object variable, create new class atrributes during the run without initialising inside the constructor. It is then display by invoking it as an instance variable. This is an reflection facility. This is needed when we have sudden requirement of any new attributes that has to work with the class.

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

Ans \_\_\_add\_\_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. **init** method is basically used for constructing the object variables meaning, initialization of object variables happen, where the class attributes gets the values from the class instance where the values are stored. So when we want to inherit the attributes/variables of the class then we need to inherit the init method also. This implies that whatever the variables were initialise in the parent class, those will also get initialise in the subclass. We uses **super()** method to call the parent class.

We can customize the behavior of the subclass by declaring some new other attributes and initializing them. However these new attributes will only get accessed by the subclass object. We could use **overriding** technique also for changing the behavior of the init method without defining it separately inside the subclass