**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 user-defined data structure classes define a function called methods which can be use to get some information based on the function is create. A class can be describe a blueprint for some structure that the programmer is defined, it would take data from user and are able to return some value through methods as it created.

And the other side an instance is user object of that class, where the user will put their information in to the class and will get information based on there data. As many user will use the class an instance will be that many.

This is one-to-many relationship, one class can be use by many user. Like an example can be. If we have a Student class which take some input like roll no and student name, then one school will have many student and it is likely to have different name and roll number for each student. Here the Student class is one and its relationship is with many students.

**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 - Class creates a user-defined data structure, which holds its own 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 regular function is not dependent on any other object but a class method Is dependent on the class instance or class object. For using that class method we need class instance and we need to put class instance and dot followed by the function or method name. but in regular function does not need any of these we can simply call it by regular way.

**Q5. Is inheritance supported in Python, and if so, what is the syntax?**  
ANS - Yes inheritance is support in python and it is denoted inside of parenthesis followed by child class in the time of defining of child class.here is an example of syntax of inheritance of parent class inside of child class

class Parent(): # creating aparent class

pass

class Child(Parent): # creating child class inheritance with parent class

pass

**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 - Python supports encapsulation. It combine all the behaviour, properties and attributes of a class inside a single class object. So that we do not have to perform separate separate operation through different means. We just need a single object to perfomr all those task. It also hide the implementation of the task. Means we do ot required to focus on the backend task. We just need to foucs on the output, result when any operation si performed.

If we talk about making instance or class variables private, then yes. It makes them both of them private so that there is not chances of changing or modifying the code or some values. So we just make those attributes and methods private which have a severe impact on the codes, so that any other person can figure it it out by just looking into it.

**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 - add is simply an built method which add the two items when it is pased, 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 constructr. 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 - It is an inplace add method which is an built in method. Like that of add , it also adds the two value but unlike \_ add \_ , it stores that value which was calculated. iadd is actually storing value its adding into the instance variable and return the new value.

**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, initialisation 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 in order to call the parent class.

We can customise the behaviour of the subclass by declaring some new other attributes and initialising them. However these new attributes will only get accessed by the subclass object. We could use overridding technique also for changing the behaviour of the init method without defining it separately inside the subclass