Q1. What is the purpose of Python's OOP?

- Object Oriented Programming allows us to create classes in Python Programming. These classes can used to create real world entities with features like inheritance, polymorphism, abstraction and encapsulation.
- OOP is very useful to bind the data and the methods that work on that together as a single unit.
- These class objects allow us write clean and easy to understand code.
- These classes are sharable with some level of privacy associated.
- These classes will help to reuse the code with less effort.

Q2. Where does an inheritance search look for an attribute?

- It searches in the upward direction from instances to super classes.
- It starts from the object attribute, then to the parent classes and the super classes.

Q3. How do you distinguish between a class object and an instance object?

- Class object:
  - Classes are the building blocks for OOP
  - In Python, everything is an object of a class.
  - Classes also allow us to create user defined data types.
  - It can hold information in the form of attributes and perform tasks using the method calls.
  - No memory is allocated when classes are created.
  - Classes can not be manipulated, until we redefine them.
  - Classes are defined with the ‘class’ keyword.
- Instance Object
  - Instance object is built on a class.
  - To access the information present in the classes, and perform tasks written in the method calls, we have to create instance of those classes.
  - When instance objects are created on a particular class, memory will be allocated.
  - Multiple instances can be created on a single class.
  - Objects can be manipulated.
  - Instance objects are created by using the class name with the required arguments/keyword arguments.

Q4. What makes the first argument in a class’s method function special?

- We use ‘self’ argument by convention.
- This argument (‘self’) is used to bind the attributes and the methods of the class with the instance object created on the class.

Q5. What is the purpose of the __init__ method?

- It is called as class constructor or initializer.
- This is used to assign attributes to the instances created on the class.

Q6. What is the process for creating a class instance?

- Class instances are created for accessing the methods defined in the given class and use the attributes of the class as they were declared during the class creation.
- This helps in writing clean and clear code
- Any number of class instances are created on a single declared class. Thus, it helps to reduce the size of the code.

Q7. What is the process for creating a class?

- Start with class keyword and write the name of the class you want to create followed by a colon for adding the code inside in this class.
- By convention, camel casing is used for class names.
- You can provide a doc string for explaining the purpose of the classes or any other details you want to mention.
- For defining the arguments required for the class creation and attributes created for the instance of the classes. Define an __init__ method with the first keyword representing the class instance. 'self' keyword is used by convention. This keyword representing the instance object is not considered as the argument required for the class creation. Add all the arguments next to self-keyword with comma separations for defining the required arguments for creating the class instance.
- Inside the __init__ method, define the instance attributes
- Create methods of the class as functions in the class declaration by giving first argument of function as keyword representing the class instance, which is 'self' by convention.

Q8. How would you define the superclasses of a class?

- By passing the name of the super class inside the parenthesis next to the class name we are defining.
- For using the same attributes and methods like the parent class, use the super() function.
- For using the same attributes use the following method inside the __init__ method: super().__init__()
- For using the same method inside another custom method name: super().<parent_class_method_name>()