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

Sol. In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

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

Sol. An inheritance search looks for an attribute first in the instance object, then in the class the instance was created from, then in all higher superclasses, progressing from the bottom to the top of the object tree, and from left to right (by default). The search stops at the first place the attribute is found. Because the lowest version of a name found along the way wins, class hierarchies naturally support customization by extension.

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

Sol. Both class and instance objects are namespaces (packages of variables that appear as attributes). The main difference between them is that classes are a kind of factory for creating multiple instances. Classes also support operator overloading methods, which instances inherit, and treat any functions nested within them as special methods for processing instances.

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

Sol. The first argument in a class method function is special because it always receives the instanceobject that is the implied subject of the method call. It’s usually called self by convention.Because method functions always have this implied subjectobject context by default, we say they are “object-oriented”—i.e., designed to process or change objects

Q5. What is the purpose of the \_\_init\_\_ method?

Sol. If the \_\_init\_\_ method is coded or inherited in a class, Python calls it automatically each time an instance of that class is created. It’s known as the constructor method;it is passed the new instance implicitly, as well as any arguments passed explicitly to the class name. It’s also the most commonly used operator overloading method. If no \_\_init\_\_ method is present, instances simply begin life as empty namespaces.

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

Sol. You create a class instance by calling the class name as though it were a function; any arguments passed into the class name show up as arguments two and beyondin the \_\_init\_\_ constructor method. The new instance remembers the class it was created from for inheritance purposes.

Q7. What is the process for creating a class?

Sol. You create a class by running a class statement; like function definitions, these statements normally run when the enclosing module file is imported.

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

Sol. You specify a class’s superclasses by listing them in parentheses in the class statement, after the new class’s name. The left-to-right order in which the classes are listed in the parentheses gives the left-to-right inheritance search order in the class tree.