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

Python provides all the standard features of object oriented programming. Developers often choose to use OOP in their Python programs because it makes code more reusable and makes it easier to work with larger programs

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

All of these objects are namespaces (packages of variables), and the **inheritance search** is simply a **search** of the tree from bottom to top **looking** for the lowest occurrence of an **attribute** name

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

A **class** is a template for creating **objects** in program whereas **the object** is an **instance** of a **class**. A **class** is a logical entity while **object** is a physical entity

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

When a def appears inside a **class**, it is usually known as a **method**. It automatically receives a **special first argument**, self, that provides a handle back to the instance to be processed. **Methods** with two underscores at the start and end of names are **special methods**.

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

"\_\_init\_\_" is a reseved method in python classes. It is called as a constructor in object oriented terminology. This method is called when an object is created from a class and it allows the class to initialize the attributes of the class.

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

**To create instances** of a **class**, you call the **class** using **class** name and pass in whatever arguments its \_\_init\_\_ **method** accepts.

Q7. What is the process for creating a class?

A variable declaration with a variable name with an object type. Instantiation − The 'new' keyword is used **to create** the object. Initialization − The 'new' keyword is followed by a call to a constructor. This call initializes the new object

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

To recap what you've seen before, **classes** can be derived from other **classes**. The derived **class** (the **class** that is derived from another **class**) is called a subclass. The **class** from which it's derived is called the **superclass**.