**1. What is Object-Oriented Programming, and how does it differ from procedural programming?**

- Object-Oriented Programming (OOP) is a programming paradigm that organises code into objects, each representing an instance of a class. It differs from procedural programming by emphasising the encapsulation of data and methods within objects.

**2. Explain the principles of OOP and how they are implemented in Python. Describe the concepts of encapsulation, inheritance, and polymorphism in Python.**

- The principles of OOP in Python include:

- Encapsulation: Bundling data and methods within a class.

- Inheritance: A mechanism where a class inherits properties and behaviours from another class.

- Polymorphism: The ability of objects to take on multiple forms. In Python, this can be achieved through method overloading or overriding.

**3. What is the purpose of the self keyword in Python class methods?**

- The self keyword in Python class methods refers to the instance of the class and is used to access its attributes and methods within class definitions.

**4. How does method overriding work in Python, and why is it useful?**

- Method overriding in Python occurs when a subclass provides a specific implementation for a method that is already defined in its superclass. It is useful for customising the behaviour of inherited methods.

**5. What is the difference between class and instance variables in Python?**

-Class variables are shared among all instances of a class, while instance variables are specific to each instance.

**6. Discuss the concept of abstract classes and how they are implemented in Python.**

- Abstract classes in Python cannot be instantiated and may contain abstract methods. They serve as blueprints for other classes to inherit and provide concrete implementations.

**7. Explain the importance of the super() function in Python inheritance.**

- The "super()" function is important in Python inheritance as it is used to call methods from the parent class, ensuring proper inheritance and avoiding method name conflicts.

**8. How does Python support multiple inheritance, and what challenges can arise from it?**

- Python supports multiple inheritance, allowing a class to inherit from more than one parent class.

challenges:

-Inheritance can lead to tight coupling between the base class and the derived class. Changes in the base class may affect the derived class, and vice versa.

-Some programming languages support multiple inheritance, where a class can inherit from more than one class. However, this can lead to ambiguity when a class inherits from multiple classes that have methods or attributes with the same name.

**9. What is a decorator in Python, and how can it be used in the context of OOP?**

- Decorators in Python are functions that modify the behaviour of another function. In OOP, decorators can be applied to methods to extend or modify their functionality.

**10. What do you understand by Descriptive Statistics? Explain by Example.**

- Descriptive Statistics summarise and describe the main features of a dataset. An example is calculating the mean, median, and standard deviation of a set of exam scores.

**11. What do you understand by Inferential Statistics? Explain by Example.**

-Inferential Statistics make inferences and predictions about a population based on a sample. An example is using a sample mean to estimate the population mean. For example if the population is large, we take a small sample to analyse the trend of the whole population.