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

- OOP is about using objects to model real-world things and their interactions in code. It's different from procedural programming, which focuses more on step-by-step instructions.

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

- **Encapsulation:**Involves bundling data and the methods that operate on the data within one unit, a class
- **Inheritance:** Allows a class to inherit properties and behavior from another class.
- **Polymorphism:** Allows objects to be treated as instances of their parent class.

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

- In Python, self is a way to refer to the particular instance of a class.

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

- It's when a new class changes the behavior of a method it inherited from a parent class.

```python
class Parent:
    def show(self):
        print("Parent method")

class Child(Parent):
    def show(self):
        print("Child method")

obj = Child()
obj.show()  # Output: Child method
```

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

- **Class variables:** Shared by all objects of a class.
- **Instance variables:** Specific to each object (instance) of the class.

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

- Abstract classes are classes that cannot be instantiated and often contain abstract methods (methods without implementation).

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

- super() is used to access methods and properties from the parent class within a subclass.

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

- Python supports multiple inheritance, allowing a subclass to inherit from multiple parent classes. Challenges can arise in cases of method name conflicts or complex inheritance hierarchies, which can make code harder to understand and maintain.

**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 behavior of other functions or methods.
- They can be used in OOP to add functionality to methods or classes dynamically.

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

- Descriptive statistics involves summarizing and describing the features of a dataset, such as mean, median, mode, standard deviation, etc

```
data = [5, 7, 10, 3, 8]
mean = sum(data) / len(data)
```

**11. What do you understand by Inferential Statistics? Explain by Example11. What do you understand by Inferential Statistics? Explain by Example**

- Inferential statistics is like making good guesses about a big group based on what we know about a smaller part of that group.
- For Example : imagine we have a big jar of marbles, and we want to know the proportion of red marbles in the entire jar. It's too time-consuming to count all of them, so we take a small handful, count the red marbles in that handful, and use that information to estimate the proportion of red marbles in the entire jar.