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

In procedural programming the importance is given to functions whereas in Object-Oriented Programming the concept of class and objects are involved.
The reusability of code is in both procedural and OOP but the difference is procedural programming becomes difficult to handle when the code becomes larger but OOP handles easily because of techniques such as encapsulation, inheritance.
The data and functions are separated in procedural programming but in OOP both are bound together by encapsulation.

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

Everything in OOP is treated as an object. An object is an instance of class that operates on attributes and methods.
The four main principles behind OOP are abstraction, encapsulation, inheritance and polymorphism.
Encapsulation: It is a way of wrapping data(attributes) and functions(methods) as a single unit.
Inheritance: This allows a new class (child class) to inherit properties such as attributes and methods from an existing class (parent class), promoting code reuse.
Polymorphism: This means the ability of object to take different forms or the same code that can do different kinds of functions.

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

Self represents the instance of class . By using self we can access attributes and methods in python class.

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 allows a subclass to provide a specialized version of a method that is already defined in its superclass. When an object of the subclass calls that method, the overridden version in the subclass is executed instead of the one in the superclass.

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

Class variables are variables that are defined within a class but outside of any methods. They are shared by all instances (objects) of the class.
Instance variables are variables that are unique to each instance (object) of a class. They are defined within methods using the self keyword.

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

The concept of abstract class is to hide the complex implementation but to show only the essential information such as structure of the program(like blueprint).
Implementation:

```
class Shape:
  def area(self):
    pass
  def perimeter(self):
    pass
class Rectangle(Shape):
  def __init__(self, length, breadth):
    self.length = length
    self.breadth = breadth
  def area(self):
    return self.length*self.breadth
  def perimeter(self):
    return 2*(self.length*self.breadth)
class Square(Shape):
  def __init__(self, side):
    self.side = side
  def area(self):
    return self.side**2
  def perimeter(self):
    return 4*self.side
rectangle = Rectangle(4,2)
print(rectangle.area())
```

```
print(rectangle.perimeter())
square = Square(5)
print(square.area())
print(square.perimeter())
```

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

It is used to call a method from the superclass, allowing you to extend or override the behavior of that method in the subclass. The primary purposes of the super() function are to maintain code reusability, facilitate cooperative multiple inheritance, and ensure that the superclass's functionality is not completely overridden.

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

Python supports multiple inheritance, allowing a class to inherit attributes and methods from more than one parent class. In multiple inheritance, a class can have multiple base classes and it inherits features from all of them. The challenges that arise if two or more parent classes define methods or attributes with the same name. Multiple inheritance can make the code more complex and harder to understand. It may become challenging to trace the origin of methods and attributes, especially in large codebases.

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

Decorator is a special type of function or class that can be used to modify the behavior of other functions or methods. It often provides additional functionality

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

Descriptive statistics is a branch of statistics where it deals with summarizing and presenting data in a meaningful way. The three main types in descriptive statistics are measure of central tendency, measures of variability and measure of frequency distribution.
Example:
Taking score of students in a class [60, 80, 85, 60, 90]
Mean: The average of the students score ie. 375/5=75
Median: The middle value of the dataset when it is ordered ie, [60, 60, 80, 85, 90] here the median will be 80.
Mode: The most repeating value ie, 60 (which is repeated 2 times)

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

Inferential statistics draw conclusions about population based on sample data. The various aspects of inferential statistics are population inference, hypothesis testing and cross validation.
Example:
If we have a dataset , we split the dataset into train and test data. Then we train the ml model in training data and use test data to test the performance of the model. The test data which is a subset of the dataset is used to determine the performance of the whole population.
In hypothesis testing we make assumptions on the population then we draw conclusions from the sample of data.
In cross validation, we partition data into testing and training, we test the model performance on unseen data(testing data)