Chapter 10: Object-Oriented Programming

Questions and Answers

Question 1: What is a class in Python and how do you define it?

Answer: A class in Python is a blueprint for creating objects. It defines a set of attributes and methods that the created objects will have.

Example:

```
class Employee:
    company = "Google"  # Class attribute

    def getSalary(self):
        print("Salary is not specified")
```

Explanation: In this example, `Employee` is a class with a class attribute `company` and a method `getSalary`.

Question 2: How do you create an object of a class in Python?

Answer: An object is created by instantiating a class.

Example:

```
shivam = Employee()
```

Explanation: In this example, `shivam` is an object of the `Employee` class.

Question 3: What is the purpose of the `self` parameter in Python classes?

Answer: The `self` parameter refers to the instance of the class. It is used to access attributes and methods of the class in Python.

Example:

```python
class Employee:
    def __init__(self, name):
        self.name = name

    def getSalary(self):
        print(f"{self.name}'s salary is not specified")

shivam = Employee("Shivam")
shivam.getSalary()  # Output: Shivam's salary is not specified
```

Explanation: In this example, `self.name` is used to refer to the `name` attribute of the specific instance (`shivam`).

Question 4: How do you define and use a static method in Python?

Answer: A static method is defined using the `@staticmethod` decorator. It does not take the `self` parameter.

Example:

```python
class Employee:
    @staticmethod
    def greet():
        print("Hello user")

Employee.greet()  # Output: Hello user
```

Explanation: In this example, `greet` is a static method that can be called without creating an instance of the `Employee` class.

Question 5: What is the `__init__()` method in Python and when is it called?

Answer: The `__init__()` method is a special method known as the constructor. It is called automatically when an object of the class is created.

Example:

```python
class Employee:
    def __init__(self, name):
        self.name = name

    def getSalary(self):
        print(f"{self.name}'s salary is not specified")

shivam = Employee("Shivam")
shivam.getSalary()  # Output: Shivam's salary is not specified
```

Explanation: In this example, the `__init__()` method initializes the `name` attribute when a new `Employee` object is created.