Object-oriented Programming In Python

Michael H. Goldwasser ; David Letscher

Python from Scratch - Lesson 4: Object Oriented Programming. In the previous chapters, we intentionally avoided object oriented programming (OOP). We think it’s easier and more fun to start learning Python without having to deal with the syntax of object-oriented programming. However, if you are interested in learning more about it, we recommend the book “Object-Oriented Programming in Python” by Michael H. Goldwasser and David Letscher. This book provides a comprehensive introduction to OOP and covers all the important concepts in detail.

In this article, we will explore some of the key ideas and features of OOP in Python. We will start by discussing the basic concepts of classes and objects, and then move on to more advanced topics such as inheritance, encapsulation, and polymorphism.

Classes and Objects

In object-oriented programming, a class is a blueprint for creating objects. An object is an instance of a class, and it contains all the attributes and methods that are defined in the class.

A simple example of a class definition in Python is:

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

    def introduce(self):
        print(f'Hello, my name is {self.name} and I am {self.age} years old.
```

In this example, we define a class called `Person` that has two attributes: `name` and `age`.

Methods

Methods are functions that are associated with a class and can be called on an object of that class.

For example, we can define a method called `introduce` that prints out a greeting:

```python
    def introduce(self):
        print(f'Hello, my name is {self.name} and I am {self.age} years old.
```

Creating Objects

To create an object of a class, we use the `__init__` method, which is called automatically when we create a new object.

For example, we can create a new `Person` object with the following code:

```python
person = Person('Alice', 25)
```

In this example, we create a new `Person` object called `person` with the name 'Alice' and age 25.

Accessing Attributes

We can access the attributes of an object by using the dot notation.

For example, we can print the name of the person:

```python
print(person.name)
```

Inheritance

One of the most important features of OOP is inheritance. Inheritance allows us to create new classes that inherit the properties and methods of existing classes.

For example, we can create a new class called `Student` that inherits from the `Person` class:

```python
class Student(Person):
    def __init__(self, name, age, student_id):
        super().__init__(name, age)
        self.student_id = student_id

    def introduce(self):
        print(f'Hello, my name is {self.name} and I am {self.age} years old. My student ID is {self.student_id}.
```

In this example, we define a new class called `Student` that inherits from the `Person` class.

Encapsulation

Encapsulation is the practice of hiding the implementation details of a class from the outside world.

For example, we can define a method called `set_age` that allows us to set the age of a person:

```python
    def set_age(self, new_age):
        self.age = new_age
```

In this example, we define a method called `set_age` that allows us to set the age of a person.

Polymorphism

Polymorphism refers to the ability of objects of different classes to respond to the same method in different ways.

For example, we can define a method called `introduce` that is inherited by both the `Person` and `Student` classes:

```python
    def introduce(self):
        print(f'Hello, my name is {self.name} and I am {self.age} years old.
```

In this example, we define a method called `introduce` that is inherited by both the `Person` and `Student` classes.

Conclusion

In this article, we explored some of the key concepts of object-oriented programming in Python. We discussed classes, objects, methods, inheritance, encapsulation, and polymorphism. We hope that this introduction will give you a good starting point for learning more about OOP in Python.

If you enjoyed this article, you might also like our other Python tutorials, such as "Python Data Structures" and "Python Networks and Web Scraping."

---

Additional Resources:
- "Object-Oriented Programming In Python" by Michael H. Goldwasser and David Letscher
- "Python from Scratch - Lesson 4: Object Oriented Programming" by Michael H. Goldwasser and David Letscher
- "Object-Oriented Programming in Python for Dummies" by Timophon S. Jones
- "Python OOP - The Complete Guide to Python Object-Oriented Programming" by Michael H. Goldwasser and David Letscher

---

Keywords:
- Python
- Object-Oriented Programming
- Classes
- Objects
- Inheritance
- Encapsulation
- Polymorphism
- Python Programming
- OOP