Data Structures and Object-Oriented Programming in Python

Shihab Ahmed

February 28, 2025

1 Introduction

Python provides various built-in data structures that allow efficient data storage and manipulation. In addition, Python supports object-oriented programming (OOP) through classes and objects.

2 Data Structures in Python

2.1 Lists

Lists are ordered, mutable collections that allow duplicate elements.

```python
my_list = [1, 2, 3, 4]
my_list.append(5)  # Add an element
print(len(my_list))  # Get length
print(sorted(my_list))  # Sort list
```

Listing 1: List Operations

2.2 Tuples

Tuples are ordered, immutable collections.

```python
my_tuple = (10, 20, 30)
print(my_tuple.index(20))  # Find index
print(my_tuple.count(10))  # Count occurrences
```

Listing 2: Tuple Operations

2.3 Sets

Sets are unordered collections that do not allow duplicate elements.

```python
my_set = {1, 2, 3, 3}
my_set.add(4)  # Add element
print(my_set)
print(my_set.union({5, 6}))  # Union operation
```

Listing 3: Set Operations

2.4 Dictionaries

Dictionaries store key-value pairs.

```python
my_dict = {'name': 'Alice', 'age': 25}
print(my_dict.keys())  # Get keys
print(my_dict.values())  # Get values
my_dict['city'] = 'New York'  # Add key-value pair
```

Listing 4: Dictionary Operations

3 Object-Oriented Programming in Python

3.1 Classes and Objects

Classes define a blueprint for objects, which are instances of classes.

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

    def greet(self):
        return f"Hello, my name is {self.name} and I am {self.age}
            years old."

person1 = Person("Alice", 25)
print(person1.greet())
```

Listing 5: Class and Object Example

3.2 Inheritance

Python supports inheritance, allowing new classes to derive from existing ones.

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

    def describe(self):
        return f"{self.name} is a {self.job_title}."

employee1 = Employee("Bob", 30, "Software Engineer")
```

```python
print(employee1.describe())
```

Listing 6: Inheritance Example

4 Conclusion

Python provides powerful data structures and an object-oriented approach to programming, making it versatile for various applications.