Mutable vs. Immutable Objects

September 26, 2023

mutable and immutable objects in OOP in python with 30 examples

In Python, objects are categorized into two main types: mutable and immutable. Understanding the difference between these two types is crucial for effective programming. Immutable objects cannot be changed once they are created, while mutable objects can be modified after creation. Here are 30 examples to help you understand mutable and immutable objects in Python:

Immutable Objects (Cannot be changed after creation):

1. **Integers**

```python
x = 5
x += 1  # Creates a new integer object
```

2. **Strings**

```python
s = "Hello"
s += " World"  # Creates a new string object
```

3. **Tuples**

```python
t = (1, 2, 3)
# t[0] = 4  # Raises an error, tuples are immutable
```

4. **Floats**

```python
f = 3.14
# f += 1.0  # Creates a new float object
```

5. **Booleans**

```python
b = True
# b = False  # Creates a new boolean object
```

6. **Frozen Sets**

```python
fs = frozenset([1, 2, 3])
# fs.add(4)  # Raises an error, frozen sets are immutable
```

Mutable Objects (Can be modified after creation):

1. **Lists**

```python
lst = [1, 2, 3]
lst.append(4)  # Modifies the existing list
```

2. **Dictionaries**

```python
d = {'name': 'Alice', 'age': 30}
d['age'] = 31  # Modifies the value associated with the 'age' key
```

3. **Sets**

```python
```

```
s = {1, 2, 3}
s.add(4)  # Modifies the existing set
```

4. **Byte Arrays**

```python
ba = bytearray([1, 2, 3])
ba[0] = 4  # Modifies the existing byte array
```

5. **Custom Mutable Objects (Classes)**

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

person = Person("Bob", 25)
person.age = 26  # Modifies the age attribute
```

6. **Lists of Lists**

```python
matrix = [[1, 2], [3, 4]]
matrix[0][0] = 5  # Modifies the existing list within a list
```

7. **Dictionaries of Dictionaries**

```python
data = {'user': {'name': 'Alice', 'age': 30}}
data['user']['age'] = 31  # Modifies the existing dictionary withir
```

8. **Custom Mutable Objects (Classes)**

```python
class ShoppingCart:
```

```
    def __init__(self):
        self.items = []

    def add_item(self, item):
        self.items.append(item)

cart = ShoppingCart()
cart.add_item("Product A")  # Modifies the list of items
```

9. **Lists of Custom Objects**

python Copy code

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

products = [Product("Item 1", 10), Product("Item 2", 15)]
products[0].price = 12  # Modifies the price attribute of the first
```

10. **Dictionaries with Lists**

python Copy code

```python
order = {'items': ['Item A', 'Item B']}
order['items'].append('Item C')  # Modifies the list of items in th
```

11. **Custom Mutable Objects (Classes) with Mutable Attributes**

python Copy code

```python
class BankAccount:
    def __init__(self, balance):
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
```

```
account = BankAccount(100)
account.deposit(50)  # Modifies the balance attribute
```

These examples demonstrate the concept of mutable and immutable objects in Python and how they behave differently when you attempt to modify them. Immutable objects, once created, cannot be changed, while mutable objects can be modified in place.