List

Kittikun Jitpairod

Introduction

In Python, a list is an ordered, mutable collection of items. Lists can contain elements of different types, including other lists. They are defined by enclosing comma-separated values in square brackets [ ].

Key characteristics of lists:

- Ordered: Elements maintain their order.
- Mutable: Can be modified after creation.
- Dynamic: Can grow or shrink in size.
- Heterogeneous: Can contain items of different types.

Creating Lists

Lists can be created in several ways:

```
empty_list = []
numbers = [1, 2, 3, 4, 5]
fruits = ["apple", "banana", "cherry"]
mixed = [1, "hello", 3.14, True]
nested = [1, [2, 3], [4, 5, 6]]

print(empty_list)
print(numbers)
print(fruits)
print(mixed)
print(nested)
```

```
[]
[1, 2, 3, 4, 5]
['apple', 'banana', 'cherry']
[1, 'hello', 3.14, True]
[1, [2, 3], [4, 5, 6]]
```

Accessing List Elements

Indexing

Python uses zero-based indexing. Negative indices count from the end of the list.

```
fruits = ["apple", "banana", "cherry"]
print(fruits[0])
print(fruits[-1])
```

```
apple
cherry
```

Slicing

Slicing allows access to a range of elements. The syntax is [start:end:step], where start is inclusive and end is exclusive.

```
numbers = [0, 1, 2, 3, 4, 5]
print(numbers[1:4])
print(numbers[:3])
print(numbers[2:])
print(numbers[::2])
```

```
[1, 2, 3]
[0, 1, 2]
[2, 3, 4, 5]
[0, 2, 4]
```

Modifying Lists

Changing Elements

Individual elements can be modified by assigning new values to specific indices.

```
fruits = ["apple", "banana", "cherry"]
fruits[1] = "blueberry"
print(fruits)
```

```
['apple', 'blueberry', 'cherry']
```

Adding Elements

Elements can be added to lists using various methods:

```
fruits = ["apple", "banana"]
fruits.append("cherry")        # Adds to the end
fruits.insert(1, "blueberry")  # Inserts at specific index
fruits.extend(["date", "fig"]) # Adds multiple elements
print(fruits)
```

```
['apple', 'blueberry', 'banana', 'cherry', 'date', 'fig']
```

Modifying Lists

Removing Elements

Elements can be removed from lists in several ways:

```
fruits = ["apple", "banana",
          "cherry", "date"]

# Remove by value
fruits.remove("banana")
print(fruits)
```

```
['apple', 'cherry', 'date']
```

```
# Remove by index
popped = fruits.pop(1)
print(f"Removed {popped}")
print(fruits)
```

```
Removed cherry
['apple', 'date']
```

```
# Remove last item
last = fruits.pop()
print(f"Removed {last}")
print(fruits)
```

```
Removed date
['apple']
```

```
# Clear all items
fruits.clear()
print(fruits)
```

```
[]
```

List Operations

Concatenation

Lists can be combined using the + operator:

```
combined = [1, 2, 3] + [4, 5, 6]
print(combined)
```

```
[1, 2, 3, 4, 5, 6]
```

Repetition

Lists can be repeated using the * operator:

```
repeated = [1, 2, 3] * 3
print(repeated)
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Membership Test

The `in` keyword checks if an item exists in a list:

```
fruits = ["apple", "banana", "cherry"]
print("banana" in fruits)
print("mango" in fruits)
```

```
True
False
```