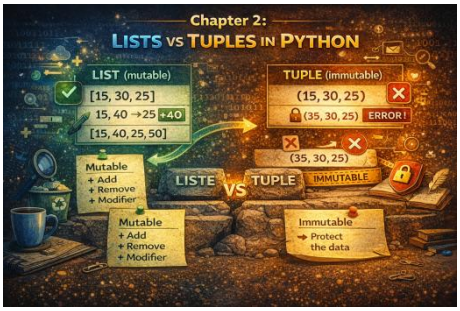
Chapter 2: Lists and Tuples in Python



Introduction

In Python, a program often needs to store **multiple values** together. Instead of creating many variables, Python provides **data structures**.

The two most important sequence data structures are:

- **Lists** → mutable (modifiable)
- **Tuples** → immutable (not modifiable)

This chapter explains:

- How to create, access, and manipulate lists
- How tuples work and why they are immutable
- When and why to use lists and tuples

1. Lists in Python

1.1 What is a List?

A **list** is an ordered collection of elements.

- Elements can be of **different types**
- Lists are **mutable**
- Defined using **square brackets `[ ]`**

1.2 Creating a List

```
numbers = [10, 20, 30, 40]
names = ["Ali", "Sara", "Omar"]
mixed = [1, "Python", 3.14, True]
```

Output:

```
print(numbers)
print(names)
print(mixed)
[10, 20, 30, 40]
['Ali', 'Sara', 'Omar']
[1, 'Python', 3.14, True]
```

Remark: A list can contain integers, strings, floats, and booleans at the same time.

1.3 Accessing List Elements (Indexing)

List indexing works like strings.

```
numbers = [10, 20, 30, 40]
print(numbers[0])
print(numbers[1])
print(numbers[-1])
```

Output:

```
10
20
40
```

Remarks:

- Index starts at **0**
- `-1` gives the last element

1.4 List Length

```
print(len(numbers))
```

Output:

```
4
```

`len()` returns the number of elements in the list.

2. List Slicing (Decoupage)

Slicing extracts part of a list.

```
numbers = [10, 20, 30, 40, 50]
print(numbers[1:4])
print(numbers[:3])
print(numbers[::2])
```

Output:

```
[20, 30, 40]
[10, 20, 30]
[10, 30, 50]
```

Remark: Slicing syntax:

```
list[start : end : step]
```

3. Modifying Lists (Manipulation)

3.1 Modifying an Element

```
numbers[1] = 25
print(numbers)
```

Output:

```
[10, 25, 30, 40, 50]
```

Lists are **mutable**.

3.2 Adding Elements

➢ **`append()` – Add at the end**

```
numbers.append(60)
print(numbers)
[10, 25, 30, 40, 50, 60]
```

➢ **`insert()` – Add at a specific position**

```
numbers.insert(1, 15)
print(numbers)
[10, 15, 25, 30, 40, 50, 60]
```

3.3 Removing Elements

➢ **`remove()` – Remove by value**

```
numbers.remove(30)
print(numbers)
[10, 15, 25, 40, 50, 60]
```

➢ **`pop()` – Remove by index**

```
numbers.pop(2)
print(numbers)
[10, 15, 40, 50, 60]
```

➢ **`del`**

```
del numbers[0]
print(numbers)
[15, 40, 50, 60]
```

4. Common List Operations

4.1 Looping Through a List

```
names = ["Ali", "Sara", "Omar"]
for name in names:
```

```
    print(name)
```
Output:
```
Ali
Sara
Omar
```
4.2 Checking Membership
```
print("Ali" in names)
print("John" in names)
```
Output:
```
True
False
```
5. Tuples in Python
5.1 What is a Tuple?
A **tuple** is an ordered collection of elements:

- Defined using **parentheses ( )**
- **Immutable** (cannot be modified)
- Faster and safer than lists

5.2 Creating a Tuple
```
coordinates = (10, 20)
colors = ("red", "green", "blue")
print(coordinates)
print(colors)
```
Output:
```
(10, 20)
('red', 'green', 'blue')
```
5.3 Accessing Tuple Elements
```
print(colors[0])
print(colors[-1])
```
Output:
```
red
blue
```
5.4 Immutability of Tuples
```
colors[0] = "yellow"
```

 Error:

```
TypeError: 'tuple' object does not support item assignment
```

Important Remark: Tuples **cannot be modified** after creation.

6. Lists vs Tuples (Comparison)

Feature	**List**	**Tuple**
Brackets	`[ ]`	`( )`
Mutable	Yes	No
Speed	Slower	Faster
Use case	Data changes	Fixed data