Python Notes — Lists, Tuples, Strings & Dictionaries

---

1. Introduction

Python provides several **data structures** to store and organize data efficiently:

- **List** → Ordered, changeable collection
- **Tuple** → Ordered, unchangeable collection
- **String** → Sequence of characters
- **Dictionary** → Key–value pairs

These are among Python's most powerful and commonly used structures.

---

2. Lists

Definition:

A **list** is an **ordered**, **mutable (changeable)** collection of items, enclosed in **square brackets [ ]**.

Example:

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

Output:

```
['apple', 'banana', 'cherry']
```

Creating Lists:

```
numbers = [10, 20, 30, 40]
mixed = [25, "hello", 3.14, True]
empty = []
```

Accessing List Elements:

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

Modifying a List:

```
fruits[1] = "kiwi"
print(fruits)
```

Adding Elements:

```
fruits.append("mango")       # add at end
fruits.insert(1, "orange")   # insert at position 1
```

Removing Elements:

```
fruits.remove("apple")   # remove by value
fruits.pop(1)            # remove by index
del fruits[0]            # delete element
```

Useful List Functions:

Function	Description	Example
`len()`	Count elements	`len(fruits)`
`sum()`	Sum numbers	`sum(numbers)`

`max()`	Maximum value	`max(numbers)`
`min()`	Minimum value	`min(numbers)`
`sort()`	Sort list ascending	`fruits.sort()`
`reverse()`	Reverse order	`fruits.reverse()`

Looping Through a List:

```
for fruit in fruits:
    print(fruit)
```

List Comprehension:

A **short way** to create new lists.

```
squares = [x**2 for x in range(5)]
print(squares)
```

Output:

```
[0, 1, 4, 9, 16]
```

---

3. Tuples

Definition:

A **tuple** is an **ordered**, **immutable** collection, enclosed in **parentheses ( )**.

Example:

```
numbers = (10, 20, 30)
print(numbers)
```

Creating Tuples:

```
t1 = (1, 2, 3)
t2 = ("a", "b", "c")
t3 = (1, "apple", 3.14)
```

Accessing Tuple Elements:

```
print(t1[0])
print(t1[-1])
```

Why Tuples?

- Faster than lists
- Data safety (cannot be changed)
- Can be used as keys in dictionaries

Converting Between List and Tuple:

```
list1 = [1, 2, 3]
tuple1 = tuple(list1)
list2 = list(tuple1)
```

Tuple Unpacking:

```
colors = ("red", "green", "blue")
a, b, c = colors
print(a)
```

Output:

```
red
```

---

4. Strings

Definition:

A **string** is a **sequence of characters** enclosed in quotes (`' '`, `" "`, or `''' '''`).

Example:

```
name = "Python"
print(name)
```

Accessing Characters:

```
name = "Python"
print(name[0])    # P
print(name[-1])   # n
```

String Slicing:

```
text = "HelloWorld"
print(text[0:5])     # Hello
print(text[:5])      # Hello
print(text[5:])      # World
```

Common String Methods:

Method	Description	Example
`upper()`	Convert to uppercase	`"hello".upper()`
`lower()`	Convert to lowercase	`"HELLO".lower()`

`title()`	First letter capital	`"hello world".title()`
`replace()`	Replace substring	`"apple".replace("a", "A")`
`find()`	Find substring index	`"hello".find("e")`
`split()`	Split string	`"a,b,c".split(",")`
`join()`	Join list into string	`",".join(["a","b","c"])`

Checking Membership:

```
word = "Python"
print("th" in word)    # True
print("z" not in word) # True
```

Looping Through a String:

```
for ch in "Hello":
    print(ch)
```

---

5. Dictionaries

Definition:

A **dictionary** stores data in **key–value pairs**, enclosed in **curly braces { }**.

Example:

```
student = {"name": "Ravi", "age": 21, "marks": 85}
print(student)
```

Accessing Values:

```
print(student["name"])
print(student.get("marks"))
```

Adding / Updating Values:

```
student["age"] = 22
student["city"] = "Delhi"
```

Removing Elements:

```
student.pop("marks")      # remove key
del student["city"]       # delete key
student.clear()           # clear dictionary
```

Looping Through Dictionary:

```
for key, value in student.items():
    print(key, ":", value)
```

Useful Dictionary Methods:

Method	Description	Example
`keys()`	Returns all keys	`student.keys()`
`values()`	Returns all values	`student.values()`
`items()`	Returns key-value pairs	`student.items()`
`update()`	Updates with another dict	`student.update({"age": 23})`

Example: Dictionary of Students

```
marks = {"Aman": 85, "Ravi": 90, "Sneha": 78}
for name in marks:
    print(name, "=>", marks[name])
```

Output:

```
Aman => 85
Ravi => 90
Sneha => 78
```

---

6. Quick Summary Table

Data Type	Ordered	Mutable	Syntax	Example
List	✅	✅	`[ ]`	`[1, 2, 3]`
Tuple	✅	❌	`( )`	`(1, 2, 3)`
String	✅	❌	`" "`	`"Python"`
Dictionary	❌	✅	`{ }`	`{"a":1,"b":2}`