# Python
for everybody

6. Data Structures

What is a Data Structure?

Definition

A **data structure** is a collection of items. In simple words, this is the way data is stored and organized in our system.

Choosing the **suitable** structure depends on our context, since each structure has its own advantages and limits.

Actually, we have already discussed some of them like **Lists** and **Strings**. In this session, we will focus on:

- Dictionaries
- Sets
- Tuples

6. Data Structures

6.1. Dictionaries

Dictionary: Definition & Syntax

Definition:

It is a collection that stores data in key-value pairs.

Syntax:

```python
my_dict = {
    "key1": "value1",
    "key2": "value2"
}
```

Characteristics of Dictionaries

1. Each item in a dictionary is a pair of **Key : Value**.
2. **Keys** must be unique and immutable (strings, numbers, tuples).
3. **Values** can be duplicated and of any type.
4. Since Python 3.7, dictionaries are **ordered** (insertion order is preserved).

Creating and Accessing

```python
student = {
    "name": "Ali",
    "age": 20,
    "city": "Rabat"
}

# Accessing a value by key
print(student["name"])    # Output: Ali

# Check if a key exists
print("city" in student)  # True

# Adding a new key-value pair
student["class"] = "CP2"

# Modifying a value
student["city"] = "Casablanca"
```

Dictionary Methods

```python
# Phone numbers stored as Strings to avoid syntax errors
TeleDirectory = {"ahmed": "0666666666", "mohamed": "0548129955"}

keys = TeleDirectory.keys()
values = TeleDirectory.values()
items = TeleDirectory.items()

# Safe Access
# Returns "Not found" if key doesn't exist (no error)
x = TeleDirectory.get("khalid", "Not found")

# Removing items
TeleDirectory.pop("ahmed") # Removes specific key
TeleDirectory.clear()      # Removes everything
```

Looping through Dictionaries

1. **Iterating through Items (Key & Value)**

```python
for key, value in TeleDirectory.items():
    print("Name:", key, "-> Tele:", value)
```

2. **Iterating through Keys (Default behavior)**

```python
for key in TeleDirectory:
    print("Name:", key, "-> Tele:", TeleDirectory[key])
```

6. Data Structures

6.2. Sets

Sets: Definition

Definition:

Set is an unordered collection of **unique** elements.

Syntax:

```python
myset = set() # Empty set

ids = set([10, 142, 83, 48, 48, 10, 50, 12])
print(ids)
# Output: {48, 50, 83, 10, 142, 12} (Order is random)
# Note: Duplicates are automatically removed
```

Characteristics of Sets

- **No duplicates:** Great for filtering repeated data.
- **Unordered:** You cannot access items by index (e.g., **`ids[0]`** will fail).
- **Hashing:** Internally it uses hashing for performance, so items must be *hashable* (immutable types like int, str, tuple).
- **Mutable:** The set itself can change (add/remove items), but the items inside must be immutable.
- Very useful for **mathematical operations**.

Set Methods

```python
ids = {"#10", "#120", "#53"}

# Adding items
ids.add("#1555")
ids.update(["#1230", "#8942"])

# Removing items
ids.remove("#120")   # Raises Error if not found
ids.discard("#999")  # No Error if not found

# Clear set
ids.clear()
```

Operations on Sets

```python
S1 = {1, 2, 3, 4, 5, 6, 7, 8, 9}
S2 = {5, 6, 7, 8, 9, 10, 11, 12}

# Union (All elements from both)
S3 = S1.union(S2)  # Result: {1, 2, ..., 12}
# OR: S3 = S1 | S2

# Intersection (Common elements)
S4 = S1.intersection(S2) # Result: {5, 6, 7, 8, 9}
# OR: S4 = S1 & S2

# Difference (In S1 but not in S2)
S5 = S1.difference(S2)   # Result: {1, 2, 3, 4}
# OR: S5 = S1 - S2
```

6. Data Structures

6.3. Tuples

Tuples: Definition

Definition:

A Tuple is an ordered collection similar to a list, but it is **immutable**.

Syntax:

- Lists use square brackets: `[1, 2]`
- Tuples use parentheses: `(1, 2)`

```python
geoLocation = ('10', '20', '30')
```

Characteristics of Tuples

- **Immutable:** Once created, elements cannot be changed.
- **Use Case:** Useful for data that should not change during execution (e.g., weekdays, GPS coordinates, configuration constants).
- **Dictionary Keys:** Unlike lists, Tuples can be used as keys for dictionaries (because they are hashable).

Manipulating Tuples

```python
# Casting a list to a tuple
mytuple = tuple([4, 7, 8, 9])

# Comparing two tuples (Element by element)
# Checks 1 vs 1, then 2 vs 7...
print((1, 2, 3) < (1, 7, 9)) # Output: True

# Tuple Assignment (Unpacking)
x, y = (15, 30)
# Result: x = 15, y = 30
```

Tuples as Keys

```python
TeleDirectory = {}


# form a unique identifier
person1 = ("Ahmed", "Alami")
person2 = ("Ahmed", "Khalidi")
person3 = ("Ali", "Alami")

TeleDirectory[person1] = "0610459784"
TeleDirectory[person2] = "0610515454"
TeleDirectory[person3] = "0698894474"

print(TeleDirectory[("Ahmed", "Alami")])
# Output: 0610459784
```

Data Structures Summary

Type	**Syntax**	**Ordered?**	**Mutable?**
List	`[1, 2]`	Yes	Yes
Dictionary	`{"k":"v"}`	No (mostly)	Yes
Tuple	`(1, 2)`	Yes	No
Set	`{1, 2}`	No	Yes