# Tasks Accomplished today

**17-02-2025**
**Suneetha.V**

## 1. What is a List

A **list** in Python is an **ordered** collection of items that can be of any data type (integer, string, float, etc.). Lists are **mutable**, meaning their contents can be changed after they are created.

### Key Features:

- **Ordered**: Items in a list are stored in a specific order, and this order is maintained.
- **Mutable**: You can change the contents (add, remove, or modify elements).
- **Indexed**: You can access items by their index, starting from `0`.
- **Can contain duplicates**: A list can have multiple occurrences of the same item.
- **Allows mixed data types**: Lists can store elements of different types.

- **Syntax for creating a list:**

  my_list = [1, 2, 3, 4, "hello", 3.14]

  Example:

  #list----------

  my_list = [1, 2, 3, 4, "hello", 3.14]

  print(my_list)

  output:[1, 2, 3, 4, 'hello', 3.14]

## 2. What is a Tuple

A **tuple** in Python is similar to a list but **immutable**. Once created, the contents of a tuple cannot be changed. Tuples are often used to store heterogeneous data or to ensure that the data cannot be modified.

# Tasks Accomplished today

**Key Features:**

- **Ordered**: Like lists, tuples maintain the order of elements.
- **Immutable**: Once a tuple is created, you cannot modify, add, or remove elements.
- **Indexed**: Elements can be accessed using indices.
- **Allows mixed data types**: You can store elements of different types in a tuple.
- **Can contain duplicates**: Tuples can store multiple identical values.

```
# Creating a tuple:
my_tuple = (1, 2, 3, "hello", 3
```

Example:

```
# example:
my_tuple = (1, 2, 3, "hello", 3.14)
print(my_tuple)
```

Output: (1, 2, 3, 'hello', 3.14)

## 3. What is a Set

A **set** is an **unordered** collection of unique elements. It does not maintain any specific order, and it automatically removes duplicate values. Sets are **mutable**, but their elements must be **immutable** (e.g., numbers, strings, tuples).

**Key Features:**

- **Unordered**: The elements do not have a specific order, and you cannot access them by index.
- **Unique elements**: Sets automatically eliminate duplicates.
- **Mutable**: You can add or remove elements, but you cannot modify individual elements.
- **Fast membership testing**: Checking if an element exists in a set is generally faster than in a list or tuple.

# Tasks Accomplished today

```
# Creating a set:

 my_set = {1, 2, 3, 4, 5}

# example:

 my_set = {1, 2, 3, 4, 5}

 print(my_set)

output:{1, 2, 3, 4, 5}
```

## 4.what is a  Dictionary

A **dictionary** (often referred to as a **dict**) is an **unordered** collection of **key-value pairs**. Each key is unique, and each key maps to a value. Dictionaries are **mutable** and allow for fast lookups by key.

### Key Features:

- **Unordered**: The key-value pairs are not stored in any specific order (though this behavior changed in Python 3.7+ where dictionaries maintain insertion order).
- **Mutable**: You can modify, add, and remove key-value pairs.
- **Keys must be immutable**: The keys in a dictionary must be of a hashable (immutable) type (e.g., strings, numbers, tuples).
- **Fast lookups**: Values can be accessed very efficiently using their keys.

```
# Creating a dictionary

my_dict = {"name": "Suneetha", "age": 25, "city": "hyderabad"}

Example:

my_dict = {"name": "Alice", "age": 25, "city": "New York"}

print(my_dict)

Output: {'name': 'Alice', 'age': 25, 'city': 'New York'}
```

----------**-Tasks:**----------

# Tasks Accomplished today

—----------1. #creating 2 different lists:

```
# First list (of integers)
list_1 = [1, 2, 3, 4, 5]

# Second list (of mixed data types)
list_2 = ["apple", 3.14, True, 42]

# Display lists
print("List 1:", list_1)
print("List 2:", list_2)
```

output:List 1: [1, 2, 3, 4, 5]

List 2: ['apple', 3.14, True, 42] ----------

**—----------2.#creating 2 different tuples:----------**

```
# First tuple (of strings)
tuple_1 = ("cat", "dog", "bird")

# Second tuple (of mixed data types)
tuple_2 = (100, "hello", 3.14, False)

# Display tuples
print("Tuple 1:", tuple_1)
```

# Tasks Accomplished today

print("Tuple 2:", tuple_2)

output:Tuple 1: ('cat', 'dog', 'bird')

Tuple 2: (100, 'hello', 3.14, False)----------

——————3.#creating 2 different sets—————

# First set (unique integers)

set_1 = {1, 2, 3, 4, 5}

# Second set (unique mixed data types)

set_2 = {"apple", 3.14, True, 5}

# Display sets

print("Set 1:", set_1)

print("Set 2:", set_2)

output:Set 1: {1, 2, 3, 4, 5}

Set 2: {3.14, 'apple', 5, True}----------

**----------4.#creating 2 different dictionaries----------**

# First dictionary (key-value pairs)dict_1 = {

"name": "Alice",

"age": 30,

"city": "New York"

}

# Tasks Accomplished today

```
# Second dictionary (key-value pairs)
dict_2 = {
    "product": "Laptop",
    "price": 1200,
    "stock": 50
}

# Display dictionaries
print("Dictionary 1:", dict_1)
print("Dictionary 2:", dict_2)
output:Dictionary 1: {'name': 'Alice', 'age': 30, 'city': 'New York'}
        Dictionary 2: {'product': 'Laptop', 'price': 1200, 'stock':
50}--------------------------------
```

## 1. Tuple vs. List:

Both **tuples** and **lists** are ordered collections of elements in Python, but they differ primarily in terms of **mutability**, **performance**, and their typical use cases.

**Key Differences:**

| feature | tuple | list |
|---|---|---|
| mutability | Immutable (cannot be modified) | Mutable (can be modified) |
| syntax | Defined using parentheses ( ) | Defined using square brackets [ ] |
| performance | Faster in performance (due to immutability) | slightly slower (due to mutability) |
| methods | Fewer built-in methods | More built-in methods |

| | | |
|---|---|---|
| | (no `append`, `remove`, etc.) | (e.g., `append`, `insert`, `remove`, etc.) |
| duplicates | Allows duplicates | Allows duplicates |

## 2. List vs. Set:

**Lists** and **sets** are both collections, but they differ in terms of **ordering**, **duplicate handling**, and **operations**.

**Key Differences:**

| feature | list | set |
|---|---|---|
| ordering | Ordered (elements have an index) | Unordered (no indexing) |
| duplicates | Allows duplicates | Does **not allow duplicates** |
| mutability | Mutable (can be changed) | Mutable (can be changed) |
| methods | More methods (e.g., `append`, `pop`) | Fewer methods (e.g., `add`, `remove`) |

## 3. Set vs. Dictionary:

Both **sets** and **dictionaries** are unordered collections, but they differ in terms of their structure and use cases. A **set** only contains values, while a **dictionary** contains **key-value pairs**.

**Key Differences:**

| feature | set | dictionary |
|---|---|---|
| Data structure | Unordered collection of unique items | Unordered collection of key-value pairs |
| duplicates | Does **not allow duplicates** | **Keys must be unique**, values can be duplicated |
| mutability | Mutable (can add/remove | Mutable (can add/remove |

# Tasks Accomplished today

| | elements) | key-value pairs) |
|---|---|---|
| indexing | No indexing (cannot access by index) | Can access values by key (indexing by key) |

# Step 1: Create a tuple:

```
my_tuple = (1, 2, 3, "hello", 3.14)

print("Original Tuple:", my_tuple)
```

Output : Original Tuple: (1, 2, 3, 'hello', 3.14)

# Step 2: Convert the tuple to a list:

```
my_list = list(my_tuple)

print("Converted List:", my_list)
```

Output: Converted List: [1, 2, 3, 'hello', 3.14]

# Step 3: Update the list (change an element):

```
my_list[2] = 100 # Changing the third element (index 2)

print("Updated List:", my_list)
```

Output: Updated List: [1, 2, 100, 'hello', 3.14]

# Step 4: Convert the list back to a tuple:

```
my_tuple = tuple(my_list)

print("Converted Back to Tuple:", my_tuple)
```

# Tasks Accomplished today

Output:  Converted Back to Tuple: (1, 2, 100, 'hello', 3.14)

# Step 5: Delete the tuple:

```
del my_tuple
```

output:  Error: name 'my_tuple' is not defined