# Lists and Tuples

MUKESH KUMAR

# Agenda

- Creating and manipulating lists
- List comprehensions
- Tuples and immutability

## Lists

- Properties of lists
- List methods
  - Add
  - Edit
  - Delete
- List indexing
- List slicing
- List of lists

## Characteristics of Lists:

**Ordered**: The order of elements is preserved.

**Mutable**: Elements can be added, removed, or modified.

**Heterogeneous**: Can store elements of different data types.

**Dynamic**: Size is not fixed; can grow or shrink.

**Duplicates Allowed**: Same value can appear multiple times.

# List Method Definitions:

| Method Name | Definition |
|---|---|
| `append(item)` | Adds an `item` to the end of the list |
| `insert(index, item)` | Inserts an `item` at a specified `index` |
| `remove(item)` | Removes the first occurrence of a specific `item` |
| `pop(index)` | Removes and returns the item at a specific `index`, or the last item if no index is provided |
| `sort()` | Sorts the list in ascending order (or descending, if `reverse=True` is used) |
| `reverse()` | Reverses the order of the list |
| `clear()` | Removes all items from the list, making it empty |
| `extend(iterable)` | Adds all items from an `iterable` to the end of the list |
| `index(item, start, end)` | Returns the index of the first occurrence of an `item` in a given range in the list |
| `count(item)` | Returns the number of times an `item` appears in the list |
| `copy()` | Returns a shallow copy of the list |

## Common Use Cases

- Storing and manipulating collections of data.
- Implementing stacks and queues.
- Representing a sequence of items that can change.

## PYTHON TUPLES

## What is a Tuple?

- A tuple is an ordered collection of items.
- Tuples are immutable (cannot be changed after creation).

- **Syntax:**
  - tuple_name = (item1, item2, item3, ...)
- **Example:**
  - my_tuple = (1, 2, 3)

## Characteristics of Tuples

- **Ordered:** Elements maintain the order in which they are defined.
- **Immutable:** Once created, elements cannot be modified.
- **Allow duplicates:** Tuples can have duplicate values.
- **Support nesting**: Tuples can contain other tuples, lists, etc.
- Can store mixed data types: (1, "Python", True).
- Accessed via indexing and slicing.

## Tuple Methods

| Method | Description |
|---|---|
| **count()** | Returns the number of times a specified value occurs in a tuple |
| **index()** | Searches the tuple for a specified value and returns the position of where it was found |

## When to use Tuples

- Representing coordinates (x, y, z).
- Returning multiple values from a function.
- Storing configuration settings.
- Using as keys in dictionaries.

## SETS

## What are Sets?

- An unordered collection of unique elements.
- Enclosed within curly braces {}.
- Does not allow duplicate elements.

## Key Characteristics

- **Unordered:** The order of elements is not guaranteed.
- **Unique:** Only one instance of each element is allowed.
- **Mutable:** Elements can be added or removed after creation.

# Set Methods

| Method | Description |
|---|---|
| add(element) | Adds an element to the set. |
| remove(element) | Removes an element from the set. Raises KeyError if the element is not found. |
| discard(element) | Removes an element from the set if it exists. Does **not** raise an error if the element is not found. |
| pop() | Removes and returns an **arbitrary** element from the set. |
| clear() | Removes **all** elements from the set. |

# Set Operations

- **union():** The union() method combines all elements from two sets, eliminating duplicates
- **intersection():** The intersection() method returns a set containing all elements that are common to both sets.
- **difference():** The difference() method returns a set containing all elements from the first set that are not in the second set.
- **symmetric_difference():** The symmetric_difference() method returns a set containing all elements from both sets except the common elements

## Set Operations (Using Operators)

- | for union: set1 | set2
- & for intersection: set1 & set2
- - for difference: set1 - set2
- ^ for symmetric difference: set1 ^ set2

## When to Use Sets

- Removing duplicates from a list.
- Checking for membership efficiently.
- Performing set operations like union, intersection, and difference.
- Representing collections of unique items (e.g., unique words in a text).