Day 3: Lists and Tuples

LISTS

Definition**:** Ordered, mutable collection of items.

Syntax:

Square Brackets [ ]

```
my_list = [1, 'apple', 3.5]
```

Key Operations:

REMEMBER: Index starts at zero (0)

INDEXING:

Get the first item on the list:

```
my_list = [1, 'apple', 3.5]
my_list[0]
```

Result: 1

SLICING: **list_name[start:stop:step]**

Select elements from index 1 up to, but not including, index 3:

```
my_list = [1, 'apple', 3.5]
my_list[1:3]
```

Result: ['apple', 3.5]

ADDING:

Add ‘banana’ to the list:

```
my_list = [1, 'apple', 3.5]
my_list.append('banana')
```

Result: [1, ‘apple’, 3.5, ‘banana’]

REMOVING:

Remove ‘apple’ from the list:

```
my_list = [1, 'apple', 3.5]
my_list.remove('apple')
```

Result: [1,3.5]

SORTING:

Sort the list:

```
my_list = [5, 9, 3, 11]
my_list.sort()
```

Result: [3, 5, 9, 11]

OTHER METHODS:

`.extend([new_list])`

Add elements from list_b to list_a:

```
list_a = [1, 'apple', 3.5]
list_b = ['banana', 'tomato']
```

Result: [1, ‘apple’, 3.5, ‘banana’, ‘tomato’]

`.insert(index, item)`:

Insert 'banana' at index 1 in my_list:

```
my_list = [1, 'apple', 3.5]
my_list.insert(1, 'banana')
```

Result: [1, 'banana', 'apple', 3.5]

TUPLES

Definition: Ordered, immutable collection of items.

Syntax:

Parentheses ( )

```
my_tuple = (10, 20, 'orange')
```

Key Operations:

REMEMBER: Once created, tuples cannot be changed.

INDEXING:

Get the first item in my_tuple:

```
my_tuple = (10, 20, 'orange')
my_tuple[0]
```

Result: 10

SLICING:

Get a slice from my_tuple from index 0 to 2 (up to but not including 2):

```
my_tuple = (10, 20, 'orange')
my_tuple[0:2]
```

Result: (10, 20)

LENGTH: (This applies to lists as well)

Find the number of items in my_tuple:

```
my_tuple = (10, 20, 'orange')
len(my_tuple)
```

Result: 3

CONCATENATION:

Combine my_tuple with another tuple:

```
my_tuple = (10, 20, 'orange')
my_tuple + (30, 40)
```

Result: (10, 20, 'orange', 30, 40)

Use Case Comparison

LISTS: Use when you need flexibility (e.g., adding/removing items).

TUPLES: Use when you need a fixed, unchanging collection (e.g., coordinates).

Lists: Practice Exercises

1. LIST MANIPULATION:
   - Create a list of your top five favorite movies. Then, add a new movie to the end of the list using **.append()**, and remove the second movie from the list using **.remove()**.
   - Example:

```
movies = ["Inception", "Avatar", "Matrix", "Toy Story", "The Godfather"]
```

2. INDEXING AND SLICING:
   - Using a list of numbers, retrieve the last two items without using negative indices.
   - Example: `numbers = [10, 20, 30, 40, 50]`
   - Expected Output: `[40, 50]`

3. INSERTING ITEMS:
   - Start with a list of colors `['red', 'blue', 'green']`. Insert the color `'yellow'` at the second position and `'purple'` at the end of the list.
   - Expected Output: `['red', 'yellow', 'blue', 'green', 'purple']`

Tuples: Practice Exercises

1. CREATING AND INDEXING A TUPLE:

- Create a tuple named `dimensions` with three values representing length, width, and height. Access the width (the second item) and print it.
- Example: `dimensions = (10, 5, 15)`
- Expected Output: 5

2. SLICING A TUPLE:
   - Given the tuple `numbers = (0, 1, 2, 3, 4, 5, 6, 7, 8)`, retrieve a slice from index 2 to 6 (up to but not including index 6).
   - Expected Output: `(2, 3, 4, 5)`

3. CONCATENATING TUPLES:
   - Create two tuples: `fruits = ('apple', 'banana')` and `vegetables = ('carrot', 'lettuce')`. Then combine them into a new tuple called `groceries`.
   - Expected Output: `('apple', 'banana', 'carrot', 'lettuce')`