**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  
**Topic 15 - Lists**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**What is a List?**

A list in Python is a variable that can store multiple values, unlike standard variables that hold only one value. Lists are used to group related data together, which makes it easier to access and manipulate multiple values without creating individual variables for each item.

**Example of Individual Variables:**

*city\_0 = "Atlanta"*

*city\_1 = "Baltimore"*

*city\_2 = "Chicago"*

*city\_3 = "Denver"*

*city\_4 = "Los Angeles"*

*city\_5 = "Seattle"*

This works but can be cumbersome. A better way is to use a list:

*cities = ["Atlanta", "Baltimore", "Chicago", "Denver", "Los Angeles", "Seattle"]*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Why Use Lists?**

Lists help you manage groups of related items more effectively, especially when you need to work with a collection of data. Lists allow you to:

* Store multiple values in one variable.
* Access and update each value by its position, or **index**.
* Keep your code organized and make it easier to loop through multiple items.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_How to Create and Use Lists**

**1. Defining a List**

Create a list by enclosing the items within square brackets [] and separating each item with a comma.

*cities = ["Atlanta", "Baltimore", "Chicago", "Denver", "Los Angeles", "Seattle"]*

**2. Accessing Elements in a List**

You can access each element by its position, or **index**. Remember, list indices start at 0.

**Example:**

*print("Welcome to " + cities[3]) # Output: Welcome to Denver*

Here, cities[3] refers to "Denver" because it’s the fourth element (starting from index 0).

**3. Lists with Different Data Types**

Lists can hold different types of data (e.g., strings, integers, and floats) within the same list, though it's typically best to keep lists consistent.

**Example:**

*mixed\_things = [1, "Bob", "Now is"]*

*print(mixed\_things[0]) # Output: 1*

*print(mixed\_things[1]) # Output: Bob*

*print(mixed\_things[2]) # Output: Now is*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Important Points to Remember**

1. **Indexing starts at 0:**
   * The first item has an index of 0, the second item has an index of 1, and so on.
2. **Naming Rules for Lists:**
   * List names follow the same rules as variable names: only letters, numbers, and underscores are allowed, and no spaces.
   * Avoid starting list names with numbers.
3. **Use Plural Names for Lists:**
   * Using plural names (like cities instead of city) makes it clearer that the variable holds multiple items.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Summary**

Lists are a foundational feature in Python, providing a way to store and organize multiple values in a single variable. By using lists, you can keep your code clean and organized, accessing each value by its index as needed.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**