MODULE 02 - Python, Data Structures:

LIST's Theory

Introduction to Lists in Python

A **list** in Python is an **ordered, mutable collection** that can store multiple data types. Lists allow **modifications**, meaning we can add, remove, and modify elements dynamically.

Key Features of Lists

Ordered – Elements maintain their insertion order.
Mutable – Lists can be changed after creation.
Heterogeneous – Can store different data types (integers, strings, floats, etc.).
Indexed – Elements can be accessed via index values.

Python Documentation: Lists

Creating Lists

Lists are defined using square brackets `[]`.

```python
# Empty list
empty_list = []

# List with elements
numbers = [1, 2, 3, 4, 5]

# Mixed data types
mixed_list = ["Python", 3.14, True, 42]

print(numbers)  # Output: [1, 2, 3, 4, 5]
```

Python Documentation: List Literals

Accessing Elements

Python lists use **zero-based indexing**, meaning the first element is at index `0`.

```python
languages = ["Python", "Java", "C++"]

print(languages[0])  # Output: Python
print(languages[-1])  # Output: C++ (Negative indexing: last element)
```

Negative indices allow access from the end of the list (`-1` is the last element, `-2` the second-last, etc.).

Python Documentation: Indexing

Modifying Lists

Lists allow modification after creation.

```python
fruits = ["Apple", "Banana", "Cherry"]
fruits[1] = "Blueberry"  # Modifying an element
print(fruits)  # Output: ['Apple', 'Blueberry', 'Cherry']
```

Python Documentation: Mutable Sequences

Adding Elements

Python provides multiple ways to add elements to a list:

```
fruits = ["Apple", "Banana"]

# Append: Adds element at the end
fruits.append("Cherry")
print(fruits)  # ['Apple', 'Banana', 'Cherry']

# Insert: Adds element at a specific index
fruits.insert(1, "Blueberry")
print(fruits)  # ['Apple', 'Blueberry', 'Banana', 'Cherry']

# Extend: Merge another list
fruits.extend(["Mango", "Grapes"])
print(fruits)  # ['Apple', 'Blueberry', 'Banana', 'Cherry', 'Mango', 'Grapes']
```

Python Documentation: list.append(), list.insert(), list.extend()

Removing Elements

Elements can be removed using different methods:

```
numbers = [1, 2, 3, 4, 5]

# Remove by value
numbers.remove(3)
print(numbers)  # [1, 2, 4, 5]

# Remove by index (pop returns the removed element)
numbers.pop(1)
print(numbers)  # [1, 4, 5]

# Clear: Removes all elements
numbers.clear()
print(numbers)  # []
```

Python Documentation: list.remove(), list.pop(), list.clear()

Common List Methods

Method	Description
`append(x)`	Adds `x` to the end of the list
`insert(i, x)`	Inserts `x` at index `i`
`remove(x)`	Removes first occurrence of `x`
`pop(i)`	Removes and returns element at `i`
`clear()`	Removes all elements from the list
`index(x)`	Returns index of `x`
`count(x)`	Counts occurrences of `x`
`sort()`	Sorts the list in ascending order
`reverse()`	Reverses the list
`copy()`	Returns a copy of the list

Python Documentation: List Methods

Looping Through Lists

You can iterate over lists using loops.

```
colors = ["red", "green", "blue"]

for color in colors:
    print(color)
```

Using `enumerate()` to get index and value:

```
for index, color in enumerate(colors):
    print(f"Index {index}: {color}")
```

Python Documentation: Looping Techniques

Summary & Key Takeaways

Lists store multiple items in an ordered, mutable structure.
They support indexing, slicing, and iteration.
Common operations include adding, modifying, and removing elements.
List comprehensions provide an efficient way to create lists.

Python Documentation: Python Lists

Mastering lists is essential in Python, as they are one of the most widely used data structures!