Day 3: Data Structures

1. Lists

A list is an ordered, mutable collection of elements. Lists can store multiple data types such as integers, strings, floats, or even other lists.

[1] Creating Lists

Lists are created by placing elements inside square brackets [], separated by commas.

Characteristics:

• Ordered (elements have a fixed position)

• Mutable (elements can be modified)

• Allows duplicate values

[2] Indexing

Indexing is used to access individual elements in a list.

Concept:

• Indexing starts from 0 for the first element.

• Negative indexing accesses elements from the end of the list.

[3] Slicing

Slicing extracts a portion of a list.

Concept:

• Uses a start index, end index, and optional step value.

• End index is not included in the result.

[4] Common List Methods

1. append()

Adds a new element to the end of the list.

2. remove()

Removes the first occurrence of a specified value.

3. sort()

Arranges elements in ascending or descending order.

Importance of List Methods:

• Simplify data manipulation

• Improve code readability

• Reduce manual operations

2. Tuples

A tuple is an ordered collection similar to a list, but it is immutable, meaning its elements cannot be changed after creation.

[1] Immutable Sequences

• Once a tuple is created, elements cannot be added, removed, or modified.

• This makes tuples safer for fixed data.

Advantages:

• Faster than lists

• Useful for constant data

• Can be used as dictionary keys

[2] Packing and Unpacking

Tuple Packing

Combining multiple values into a single tuple.

Tuple Unpacking

Extracting values from a tuple into separate variables.

3. Dictionaries

A dictionary stores data in key–value pairs, allowing fast data retrieval.

[1] Key-Value Pairs

• Keys are unique and immutable

• Values can be any data type

• Data is accessed using keys rather than indexes

Characteristics:

- Unordered (in concept)
- Mutable
- Highly efficient lookup

[2] Dictionary Methods

• Adding new key-value pairs

• Accessing values

• Removing entries

• Retrieving keys and values collections

Dictionaries are widely used for structured data, configurations, and mappings.

[3] Nested Dictionaries

A nested dictionary contains dictionaries inside another dictionary.

• Allows representation of complex, hierarchical data

• Useful for real-world data such as user profiles or JSON data

4. Sets

A set is an unordered collection of unique elements.

[1] Unique Elements

• Duplicate values are automatically removed

• Elements are not stored in a fixed order

[2] Set Operations

- Union
- Combines elements from two sets, excluding duplicates.
- Intersection
- Returns elements common to both sets.

Set operations are useful in mathematical computations, data analysis, and comparisons.

5. String Methods

Strings are sequences of characters, and Python provides built-in methods to manipulate them efficiently.

[1] split()

Divides a string into a list of substrings based on a separator.

Processing user input or text data.

[2] join()

Combines elements of a sequence into a single string.

[3] strip()

Removes leading and trailing whitespace.

[4] replace()

Replaces occurrences of a substring with another substring.

[5] find()

Searches for a substring and returns its position.

6. List Comprehensions

List comprehensions provide a concise and readable way to create lists.

[1] Concept of List Comprehensions

• Combines looping and condition logic into a single expression

• Produces a new list without modifying the original

Advantages:

- Shorter syntax
- Improved readability
- Efficient list generation