| GEN AI LEARNING  **Complete Generative AI Learning – New Year Challenge** | Written By: Aloy  Reference: UDEMY |
| --- | --- |

**Day -2**

**Agenda:**

* Data Structure using Python
* Function in Python

In Python, there are four fundamental data structures: lists, tuples, sets, and dictionaries.

A **\*\*list\*\*** is an ordered and mutable collection of items, allowing you to store and manipulate a sequence of elements. You can access elements by indexing, slice parts of the list, and modify elements in place. Lists support various methods like `append()`, `remove()`, `pop()`, and `sort()` to manage their contents.

A **\*\*tuple\*\*** is similar to a list in that it is an ordered collection of items, but it is immutable, meaning once a tuple is created, its elements cannot be changed. Tuples support indexing and slicing but have limited methods, mainly `count()` and `index()`.

A **\*\*set\*\*** is an unordered collection of unique elements, meaning that it does not allow duplicates. Sets are mutable, so you can add or remove elements, but since they are unordered, they do not support indexing or slicing. Sets are particularly useful for performing mathematical operations like union, intersection, and difference.

A **\*\*dictionary\*\*** is an unordered collection of key-value pairs. Keys in a dictionary must be unique and immutable, while values can be of any data type. Dictionaries allow you to access elements by their keys, rather than by an index. They are highly versatile, with methods like `keys()`, `values()`, `items()`, `get()`, and `pop()` available to manage key-value pairs efficiently.

These data structures are essential for various tasks in Python, and understanding their characteristics will enable you to choose the most appropriate one for your specific needs.

**Quick Study Material on Python Data Structures: Sets, Tuples, Lists, and Dictionaries**

**1. Lists**

* **Definition**: A list is an ordered collection of items that can be of any data type. Lists are mutable, meaning you can change their elements after creation.
* **Syntax**:

python

Copy code

my\_list = [1, 2, 3, 4, 5]

* **Key Features**:
  + **Indexing**: Access elements using indices, starting from 0.

python

Copy code

first\_item = my\_list[0] # 1

* + **Slicing**: Extract parts of the list.

python

Copy code

sub\_list = my\_list[1:3] # [2, 3]

* + **Mutability**: Modify elements in place.

python

Copy code

my\_list[2] = 10 # [1, 2, 10, 4, 5]

* + **Methods**: append(), remove(), pop(), sort(), reverse(), etc.

python

Copy code

my\_list.append(6) # [1, 2, 10, 4, 5, 6]

**2. Tuples**

* **Definition**: A tuple is an ordered collection of items similar to a list, but tuples are immutable, meaning you cannot change their elements after creation.
* **Syntax**:

python

Copy code

my\_tuple = (1, 2, 3, 4, 5)

* **Key Features**:
  + **Indexing**: Similar to lists.

python

Copy code

first\_item = my\_tuple[0] # 1

* + **Slicing**: Similar to lists.

python

Copy code

sub\_tuple = my\_tuple[1:3] # (2, 3)

* + **Immutability**: Cannot change elements.

python

Copy code

my\_tuple[2] = 10 # Error: 'tuple' object does not support item assignment

* + **Methods**: Limited to methods like count() and index().

python

Copy code

my\_tuple.count(3) # 1

**3. Sets**

* **Definition**: A set is an unordered collection of unique elements. Sets are mutable but do not allow duplicate elements.
* **Syntax**:

python

Copy code

my\_set = {1, 2, 3, 4, 5}

* **Key Features**:
  + **Unordered**: No indexing or slicing since sets are unordered.
  + **Unique Elements**: Automatically removes duplicates.

python

Copy code

my\_set = {1, 2, 2, 3} # {1, 2, 3}

* + **Mutability**: You can add or remove elements.

python

Copy code

my\_set.add(6) # {1, 2, 3, 4, 5, 6}

my\_set.remove(3) # {1, 2, 4, 5, 6}

* + **Set Operations**: Union, intersection, difference.

python

Copy code

another\_set = {4, 5, 6, 7}

union\_set = my\_set.union(another\_set) # {1, 2, 4, 5, 6, 7}

**4. Dictionaries**

* **Definition**: A dictionary is an unordered collection of key-value pairs. Keys must be unique and immutable, while values can be of any data type.
* **Syntax**:

python

Copy code

my\_dict = {'a': 1, 'b': 2, 'c': 3}

* **Key Features**:
  + **Key-Value Pairs**: Access elements by keys, not by index.

python

Copy code

value = my\_dict['a'] # 1

* + **Mutability**: You can add, remove, or modify key-value pairs.

python

Copy code

my\_dict['d'] = 4 # {'a': 1, 'b': 2, 'c': 3, 'd': 4}

my\_dict['a'] = 10 # {'a': 10, 'b': 2, 'c': 3, 'd': 4}

* + **Methods**: keys(), values(), items(), get(), pop(), etc.

python

Copy code

keys = my\_dict.keys() # dict\_keys(['a', 'b', 'c', 'd'])

These are the basic data structures in Python that you'll frequently use for various tasks. Understanding their properties and methods will help you choose the right one for your specific use case.

