**Python Data Structures**

This module begins a journey into Python data structures by explaining the use of lists and tuples and how they are able to store collections of data in a single variable. Next learn about dictionaries and how they function by storing data in pairs of keys and values, and end with Python sets to learn how this type of collection can appear in any order and will only contain unique elements.

**Learning Objectives**

* Describe and manipulate tuple combinations and list data structures.
* Execute basic tuple operations in Python.
* Perform list operations in Python.
* Write structures with correct keys and values to demonstrate understanding of dictionaries.
* Work with and perform operations on dictionaries in Python.
* Create sets to demonstrate understanding of the differences between sets, tuples, and lists.
* Work with sets in Python, including operations and logic operations.

# **Lists and Tuples**

## **Lists and Tuples**

**lists and tuples**, which are important data structures in Python. Here’s a simplified explanation and summary:

Explanation:

* **Tuples**:
  + A tuple is a collection of items that are ordered and unchangeable (immutable).
  + They are written with parentheses, e.g., (1, 2, 3).
  + You can access items in a tuple using their index (starting from 0).
  + You can combine tuples or get parts of them (slicing), but you cannot change the items inside a tuple.
* **Lists**:
  + A list is similar to a tuple but is changeable (mutable).
  + Lists are written with square brackets, e.g., [1, 2, 3].
  + You can also access items using their index.
  + You can add, remove, or change items in a list.
  + Lists can be combined and sliced just like tuples.

Summary:

* The video explains the differences between tuples and lists in Python.
* It highlights that tuples are immutable and use parentheses, while lists are mutable and use square brackets.
* Both can be accessed using indexes and can be combined or sliced.
* The video also discusses how to manipulate lists, such as adding or removing elements.

# **Dictionaries**

Key Points:

* **Dictionaries vs. Lists**:
  + Lists use **integer indexes** (like addresses) to access elements.
  + Dictionaries use **keys** (which can be characters) to access **values** (the information stored).
* **Creating a Dictionary**:
  + Use **curly brackets** {}.
  + Each key is followed by a value, separated by a **colon** :.
  + Key-value pairs are separated by **commas**.
* **Example**:
  + An album title can be a key, and the release date can be the value.
* **Accessing Values**:
  + Use square brackets [] with the key to get the value.
* **Adding/Deleting Entries**:
  + You can add new key-value pairs or delete existing ones.
* **Checking Keys**:
  + Use the **"in" command** to check if a key exists in the dictionary.
* **Getting Keys and Values**:
  + Use the **keys()** method to get all keys and the **values()** method to get all values.

Summary:

The video explains how to use dictionaries in Python, highlighting their structure, how to create them, access values, and manage entries. It emphasizes the difference between dictionaries and lists, making it clear how to work with this important data structure in Python programming.

# **Sets**

Key Points:

* **Definition**: Sets are collections of unique elements and are unordered, meaning the order of elements doesn't matter.
* **Creating Sets**: You define a set using curly brackets {}. For example, {1, 2, 3} is a set.
* **Removing Duplicates**: If you create a set from a list with duplicates, the duplicates will be removed.
* **Adding and Removing Elements**:
  + Use the add() method to add an element to a set.
  + Use the remove() method to remove an element from a set.
* **Checking Membership**: You can check if an element is in a set using the in keyword.
* **Set Operations**:
  + **Intersection**: The common elements between two sets can be found using the & operator.
  + **Union**: All unique elements from both sets can be combined using the | operator.
  + **Subset**: You can check if one set is a subset of another using the issubset() method.

Summary:

The video explains how to work with sets in Python, including how to create them, add or remove elements, and perform operations like intersection and union. Sets are useful for storing unique items and performing mathematical set operations.

# **Summary**

**Module 2 Summary: Python Data Structures**

Congratulations! You have completed this module. At this point, you know that:

* In Python, we often use tuples to group related data together.Tuples refer to ordered and immutable collections of elements.
* Tuples are usually written as comma-separated elements in parentheses “()".
* You can include strings, integers, and floats in tuples and access them using both positive and negative indices.
* You can perform operations such as combining, concatenating, and slicing on tuples.
* Tuples are immutable, so you need to create a new tuple to manipulate it.
* Tuples, termed nesting, can include other tuples of complex data types.
* You can access elements in a nested tuple through indexing.
* Lists in Python contain ordered collections of items that can hold elements of different types and are mutable, allowing for versatile data storage and manipulation.
* A list is an ordered sequence, represented with square brackets "[]".
* Lists possess mutability, rendering them akin to tuples.
* A list can contain strings, integers, and floats; you can nest lists within it.
* You can access each element in a list using both positive and negative indexing.
* Concatenating or appending a list will result in the modification of the same list.
* You can perform operations such as adding, deleting, splitting, and so forth on a list.
* You can separate elements in a list using delimiters.
* Aliasing occurs when multiple names refer to the same object.
* You can also clone a list to create another list.
* Dictionaries in Python are key-value pairs that provide a flexible way to store and retrieve data based on unique keys.
* Dictionaries consist of keys and values, both composed of string elements.
* You denote dictionaries using curly brackets.
* The keys necessitate immutability and uniqueness.
* The values may be either immutable or mutable, and they allow duplicates.
* You separate each key-value pair with a comma, and you can use color highlighting to make the key more visible.
* You can assign dictionaries to a variable.
* You use the key as an argument to retrieve the corresponding value.
* You can make additions and deletions to dictionaries.
* You can perform an operation on a dictionary to check the key, which results in a true or false output.
* You can apply methods to obtain a list of keys and values in a dictionary.
* Sets in Python are collections of unique elements, useful for tasks such as removing duplicates and performing set operations like union and intersection. Sets lack order.
* Curly brackets "{}" are helpful for defining elements of a set.
* Sets do not contain duplicate items.
* A list passed through the set function generates a set containing unique elements.
* You use “Set Operations” to perform actions such as adding, removing, and verifying elements in a set.
* You can combine sets using the ampersand "&" operator to obtain the common elements from both sets.
* You can use the Union function to combine two sets, including both the common and unique elements from both sets.
* The sub-set method is used to determine if two or more sets are subsets.

# **Glossary: Python Data Structure**

**Glossary: Python Data Structures**

Welcome! This alphabetized glossary contains many of the terms in this course. This comprehensive glossary also includes additional industry-recognized terms not used in course videos. These terms are important for you to recognize when working in the industry, participating in user groups, and participating in other certificate programs.

| **Term** | **Definition** |
| --- | --- |
| Aliasing | Aliasing refers to giving another name to a function or a variable. |
| Ampersand | A character typically "&" standing for the word "and." |
| Compound elements | Compound statements contain (groups of) other statements; they affect or control the execution of those other statements in some way. |
| Delimiter | A delimiter in Python is a character or sequence of characters used to separate or mark the boundaries between elements or fields within a larger data structure, such as a string or a file. |
| Dictionaries | A dictionary in Python is a data structure that stores a collection of key-value pairs, where each key is unique and associated with a specific value. |
| Function | A function is a block of code, defining a set procedure, which is executed only when it is called. |
| Immutable | Immutable Objects are of in-built datatypes like int, float, bool, string, Unicode, and tuple. In simple words, an immutable object can't be changed after it is created. |
| Intersection | The intersection of two given sets is the largest set, which contains all the elements that are common to both sets. |
| Keys | The keys () method in Python Dictionary, returns a view object that displays a list of all the keys in the dictionary in order of insertion using Python. |
| Lists | A list is any list of data items, separated by commas, inside square brackets. |
| Logic operations | In Python, logic operations refer to the use of logical operators such as "and," "or," and "not" to perform logical operations on Boolean values (True or False). |
| Mutable | Immutable objects are of in-built datatypes like int, float, bool, string, Unicode, and tuple. A mutable object can be changed after it is created. |
| Nesting | A nested function is simply a function within another function and is sometimes called an "inner function". |
| Ratings in python | Ratings in Python typically refer to a numerical or qualitative measure assigned to something to indicate its quality, performance, or value. |
| Set operations | Set operations in Python refer to mathematical operations performed on sets, which are unordered collections of unique elements. |
| Sets in python | A set is an unordered collection of unique elements. |
| Syntax | The rules that define the structure of the language for python is called its syntax. |
| Tuples | These are used store multiple items in a single variable. |
| Type casting | In python, this is converting one data type to another. |
| Variables | In python, a variable is a symbolic name or identifier used to store and manipulate data. Variables serve as containers for values, and these values can be of various data types, including numbers, strings, lists, and more. |
| Venn diagram | A Venn diagram is a graphical representation that uses overlapping circles to illustrate the relationships and commonalities between sets or groups of items. |
| Versatile data | Versatile data, in a general context, refers to data that can be used in multiple ways, is adaptable to different applications or purposes, and is not restricted to a specific use case. |