Module 1

# Course Overview

Welcome to the **Python for Data Science, AI, and Development** course. After completing this course, you will:

* Possess basic knowledge of Python.
* Understand different data types.
* Use lists, tuples, dictionaries, and sets.
* Apply conditions and branching.
* Implement loops and create functions.
* Perform exception handling.
* Create and use objects.
* Read and write files.
* Collect data using APIs and web scraping.

In addition to the module labs, you will demonstrate your skills in a peer-graded project and validate your knowledge with the final quiz.

## Course Content

This course is divided into five modules. Aim to complete at least one module per week.

### Module 1: Python Basics

* About the Course
* Types
* Expressions and Variables
* String Operations

### Module 2: Python Data Structures

* Lists and Tuples
* Dictionaries
* Sets

### Module 3: Python Programming Fundamentals

* Conditions and Branching
* Loops
* Functions
* Exception Handling
* Objects and Classes
* Practice with Python Programming Fundamentals

### Module 4: Working with Data in Python

* Reading and Writing Files with Open
* Pandas
* NumPy in Python

### Module 5: APIs and Data Collection

* Simple APIs
* REST APIs, Web Scraping, and Working with Files
* Final Exam

## Learning Assets

This course contains a variety of learning resources:

* **Videos & Readings**: Present core instruction.
* **Labs & Activities**: Provide hands-on learning experiences.
* **Discussions**: Enable peer interaction and knowledge sharing.
* **Peer-Reviewed Project**: Mimics real-world scenarios to showcase your skills.
* **Practice Quizzes**: Allow you to test your understanding.
* **Graded Quizzes**: Assess how well you have learned the concepts.

# Introduction to Python

## Users of Python

Python is widely used by professionals across different fields because of its clear and readable syntax.

### Experienced Programmers

* Can develop the same programs from other languages with less code.

### Beginners

* Python is an excellent starting language due to its large global community and wealth of documentation.

## Benefits of Using Python

Python is a powerful, high-level, general-purpose programming language. It offers:

* A large standard library with tools for:
  + Databases
  + Automation
  + Web scraping
  + Text processing
  + Image processing
  + Machine learning
  + Data analytics
* Applications across:
  + **Data Science**
  + **Artificial Intelligence and Machine Learning**
  + **Web Development**
  + **Internet of Things (IoT)**
* A strong global community and support from the Python Software Foundation.

## Popularity of Python

Python has become the most widely used and popular programming language in the data science industry:

* **2019 Kaggle Survey**: 75% of over 10,000 respondents reported using Python regularly.
* **Glassdoor (2019)**: More than 75% of data science job listings included Python.
* Over 80% of data professionals worldwide reported using Python in 2019 surveys.

### Organizations Using Python

Major organizations that rely on Python include:

* IBM
* Wikipedia
* Google
* Yahoo!
* CERN
* NASA
* Facebook
* Amazon
* Instagram
* Spotify
* Reddit

## Python in Data Science and AI

Python’s rich ecosystem of libraries supports advanced applications:

### Scientific Computing Libraries

* Pandas
* NumPy
* SciPy
* Matplotlib

### Artificial Intelligence and Machine Learning

* TensorFlow
* PyTorch
* Keras
* Scikit-learn

### Natural Language Processing

* Natural Language Toolkit (NLTK)

## Diversity and Inclusion in the Python Community

Python’s community is known for its commitment to diversity and inclusion:

### Python Software Foundation

* Implements a code of conduct to ensure safe and inclusive participation both online and offline.

### PyLadies

* An international mentorship group dedicated to supporting women in becoming active contributors and leaders in the Python open-source community.

## Summary

* Python has clear, readable syntax for both beginners and experienced developers.
* It is widely used in Data Science, AI, Web Development, and IoT.
* Python offers powerful libraries for scientific computing, machine learning, and NLP.
* The community is globally active and emphasizes diversity and inclusion.

# Getting Started with Jupyter

## Introduction

Welcome to **Getting Started with Jupyter**. After completing this module, you will be able to:

* Describe how to run, insert, and delete a cell in a notebook.
* Work with multiple notebooks.
* Present a notebook.
* Shut down a notebook session.

## Launching a Notebook

In the lab session of this module, you can launch a notebook using the Skills Network virtual environment.

* Select the check box, click **Open Tool**, and the environment will launch Jupyter Lab.
* Once the notebook opens, you can rename it by clicking **File > Rename Notebook** and entering a new name.

### Running Your First Cell

* Create a new notebook and type print("Hello World").
* Click the **Run** button to execute the cell.
* Alternatively:
  + From the main menu bar, click **Run > Run Selected Cells**.
  + Or press **Shift + Enter** as a shortcut.
* To run all cells in the notebook, select **Run All Cells**.

## Working with Cells

### Inserting a Cell

* Click the **plus (+)** symbol in the toolbar to insert a new cell.

### Deleting a Cell

* Highlight the cell, then click **Edit > Delete Cells**.
* Shortcut: Press **D** twice (**DD**) on the highlighted cell.

### Moving Cells

* You can move cells up or down as needed.

## Working with Multiple Notebooks

* Click the **plus (+)** button in the toolbar and select the file you want to open.
* Alternatively: **File > Open New Launcher** or **File > Open New Notebook**.
* Opened notebooks can be arranged side by side.

### Example

In one notebook, assign:

one = 1

two = 2

print(one + two)

This demonstrates working with multiple notebooks simultaneously.

## Presenting with Jupyter

Jupyter supports creating presentations directly from notebooks:

* Use **Markdown cells** to add titles, text, and descriptions.
* Create plots and combine them with code outputs.
* Convert cells into slides or sub-slides for presentations.

This feature allows seamless delivery of code, visualizations, and narrative as part of a project.

## Shutting Down Notebooks

When you finish working:

* Click the **stop icon** on the sidebar (second icon from the top).
* You can terminate all sessions at once or shut them down individually.
* Once shut down, you will see **“No Kernel”** in the top-right corner, confirming the notebook is inactive.
* Close the tabs after shutting down.

## Summary

In this module, you learned how to:

* Run, insert, and delete code cells.
* Work with multiple notebooks at the same time.
* Present results using Markdown and code cells.
* Shut down notebook sessions when finished.

# Module 1 Summary: Python Basics

## Data Types in Python

Python can distinguish among various data types:

* **Integers**: Whole numbers that can be positive or negative.
* **Floats**: Numbers with decimal points that represent whole or fractional values.
* **Strings**: Text data enclosed in single or double quotes, consisting of letters, digits, whitespace, or special characters.
* **Booleans**: Represent logical values, True or False.

### Typecasting

* Integers can be converted to floats and vice versa.
* Integers and floats can be converted to strings.
* Integers or floats can be converted to Booleans:
  + 0 → False
  + Non-zero values → True

## Expressions and Operations

Expressions in Python combine values and operations to produce results.

* Supports mathematical operations such as addition, subtraction, multiplication, and division.
* // performs integer division, discarding the fractional part.
* Python follows the **BODMAS** order of operations when evaluating expressions.

## Variables

Variables store and manipulate data in Python:

* The assignment operator (=) assigns values to variables.
* Reassigning a variable overrides its previous value.
* Mathematical operations can be performed on variables.
* Changing one variable affects others only if they reference the same **mutable object**.

## String Operations

Python provides powerful operations for manipulating strings:

* **String Basics**:
  + Strings are ordered sequences of characters.
  + Characters are indexed using positive and negative indices.
  + Strings support indexing, slicing, concatenation, and replication.
  + Strings are **immutable** and cannot be changed once created.
* **Escape Sequences**:
  + \n → New line
  + \t → Tab
  + \\ → Backslash
* **String Methods**:
  + Search, modify, and format strings.
  + Change case, replace characters, and find items.
  + Applying a method to a string creates and returns a new string.

## Key Takeaways

* Python supports multiple data types: integers, floats, strings, and Booleans.
* Typecasting allows conversion between data types.
* Expressions and operators enable mathematical calculations.
* Variables help store and manipulate data.
* Strings offer rich operations like slicing, formatting, and built-in methods, but remain immutable.

# Glossary: Python Basics

Welcome! This alphabetized glossary contains many of the terms you'll find within 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 engaging in other certificate programs.

## A

* **AI**: Artificial intelligence is the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings.
* **Application Development**: The process of planning, designing, creating, testing, and deploying a software application to perform various business operations.
* **Arithmetic Operations**: Basic calculations like addition, subtraction, multiplication, and division, also known as algebraic or mathematical operations.
* **Array of Numbers**: A set of numbers or objects that follow a pattern, presented as rows and columns to explain multiplication.
* **Assignment Operator in Python**: A binary operator (=) that assigns a value to a variable.
* **Asterisk**: Symbol (\*) used to perform various operations in Python.

## B

* **Backslash**: An escape character (\) used in Python strings to treat the following character in a special way.
* **Boolean**: A system of algebraic notation using binary digits 0 (False) and 1 (True).

## C

* **Colon**: Used to represent indented blocks, and to fetch data or index ranges in arrays.
* **Concatenate**: To link things together in a chain or series.

## D

* **Data Engineering**: Field focused on turning raw data into usable information through blending, testing, and optimizing data.
* **Data Science**: Interdisciplinary field focused on extracting knowledge from large datasets, preparing data, and presenting findings for decision-making.
* **Data Type**: Refers to the type of value a variable has, defining what operations can be applied without errors.
* **Double Quote**: Symbol (" ") used to represent strings in Python.

## E

* **Escape Sequence**: A sequence of characters starting with an escape character to perform special functions.
* **Expression**: A combination of operators and operands that evaluates to a value.

## F

* **Float**: A data type representing floating-point numbers.
* **Forward Slash**: Symbol (/) used for division and other operations in Python.
* **Foundational**: Denoting an underlying basis or principle; fundamental.

## I

* **Immutable**: Objects (like int, float, bool, string, tuple) that cannot be changed once created.
* **Integer**: Whole numbers, positive, negative, or zero.

## M

* **Manipulate**: Modifying or creating new strings by altering existing ones.
* **Mathematical Conventions**: Generally agreed-upon facts, notations, or usage in mathematics.
* **Mathematical Expressions**: Statements involving numbers or variables connected by operators.
* **Mathematical Operations**: Calculating values using operands and operators.

## N

* **Negative Indexing**: Accessing elements from the end of a sequence using negative numbers.

## O

* **Operands**: Quantities on which operations are performed.
* **Operators in Python**: Symbols used to perform operations on variables and values.

## P

* **Parentheses**: Used to call an object.

## R

* **Replicate**: To make an exact copy.

## S

* **Sequence**: Formally defined as a function whose domain is an interval of integers.
* **Single Quote**: Symbol (' ') used to represent strings in Python.
* **Slicing in Python**: Technique to extract portions of a list or string.
* **Special Characters**: Characters not considered letters or digits, such as symbols or punctuation.
* **Stride Value**: The number of bytes from one row of pixels to the next in memory.
* **Strings**: Arrays of Unicode characters.
* **Substring**: A sequence of characters within a larger string.

## T

* **Type Casting**: Converting one data type to another (also called Type Conversion or Coercion).
* **Types in Python**: Categories of data items, such as integers, floats, strings, and Booleans.

## V

* **Variables**: Containers for storing data values.

Module 2

# Lists and Tuples

## Compound Data Types

Lists and tuples are examples of compound data types in Python. They are key data structures used to organize and manage collections of data.

## Tuples

### Definition

* Tuples are **ordered sequences**.
* Represented as comma-separated elements enclosed in parentheses ( ).
* They can contain multiple data types (strings, integers, floats, etc.), but the variable type remains a **tuple**.

### Accessing Elements

* Elements are accessed using **indexing** with square brackets [].
* Indexing starts from 0.
* Negative indexes can be used to access elements from the end of the tuple.

### Operations on Tuples

* **Concatenation**: Tuples can be combined using the + operator.
* **Slicing**: A range of elements can be accessed using [start:end]. The end index is exclusive.
* **Length**: The len() function returns the number of elements in a tuple.

### Immutability

* Tuples are **immutable** (cannot be modified after creation).
* Assigning one tuple variable to another references the same object.
* To modify a tuple, a **new tuple** must be created.

### Functions with Tuples

* The sorted() function creates a **new sorted list** from a tuple.
* Tuples can also contain other tuples or complex objects, known as **nesting**.

### Nesting and Indexing

* Tuples can be nested (tuples within tuples).
* Standard indexing rules apply to access nested elements.
* This nesting can be visualized like a **tree structure**.
* Example: tup[2][3] -> means the fourth element of the sub-tuple at index 2 inside a tuple.

## Lists

### Definition

* Lists are **ordered sequences** represented with square brackets [ ].
* Lists are **mutable**, meaning their contents can be changed.

### Accessing Elements

* Elements are accessed using indexing, similar to tuples.
* Negative indexes are supported.
* Slicing works the same as in tuples.

### Operations on Lists

* **Concatenation**: Lists can be combined using the + operator.
* **Extend**: The .extend() method adds multiple elements to the end of a list.
* **Append**: The .append() method adds a single element.
* **Deletion**: Elements can be removed using the del command.

### Mutability

* Lists can be modified directly:
  + Changing an element by index.
  + Adding new elements using methods.
  + Deleting elements.
* **Aliasing**: Assigning one list variable to another references the same object. Changes in one affect the other. e.g. A = [1,2,3] and B = A.
* **Cloning**: To avoid aliasing, lists can be cloned so each variable references a separate object.



### Conversion

* Strings can be converted into lists using the .split() method.
* A delimiter can be passed to .split() to control how the string is divided.

## Summary

* **Tuples**: Ordered, immutable, represented by parentheses ( ).
* **Lists**: Ordered, mutable, represented by square brackets [ ].
* Both support indexing, slicing, concatenation, and nesting.
* Tuples are often used when data should remain constant, while lists are preferred when modifications are needed.

# Dictionaries

## Definition

* Dictionaries are **collections of key-value pairs** in Python.
* Represented using curly brackets { }.
* Keys act like indexes but do not have to be integers. They are usually strings and must be **immutable** and **unique**.
* Values can be immutable, mutable, and can include duplicates.

## Structure

* Each **key** is followed by a **value**, separated by a colon :.
* Each key-value pair is separated by a comma.

## Example

albums = {

"Back in Black": 1980,

"The Dark Side of the Moon": 1973,

"The Bodyguard": 1992,

"Thriller": 1982

}

* Keys: Album titles (e.g., "Back in Black").
* Values: Release years (e.g., 1980).

## Accessing Values

* Values are accessed using **square brackets** with the key.

albums["Back in Black"] # Output: 1980

albums["The Dark Side of the Moon"] # Output: 1973

## Modifying Dictionaries

* **Add Entry**: Assign a new key with a value.

albums["Graduation"] = 2007

* **Delete Entry**: Use the del command.

del albums["Thriller"]

## Checking Membership

* Use the in command to check if a key exists.

"Back in Black" in albums # Output: True

"Random Album" in albums # Output: False

## Dictionary Methods

* **Keys**: albums.keys() returns all keys.
* **Values**: albums.values() returns all values.

## Visualization

* Think of a dictionary like a **table**:
  + The first column represents **keys** (e.g., album titles).
  + The second column represents **values** (e.g., release years).

## Summary

* Dictionaries store data as key-value pairs.
* Keys must be unique and immutable.
* Values can be duplicates and may be mutable or immutable.
* They are powerful for lookups, modifications, and managing structured data.

# Sets

## Definition

* Sets are a **collection type** in Python.
* Like lists and tuples, they can contain different data types.
* **Key Characteristics**:
  + **Unordered**: They do not record element positions.
  + **Unique Elements**: Duplicate values are not allowed.

## Creating Sets

* Defined using **curly brackets { }**.
* Duplicate items are automatically removed.
* You can convert a list to a set using the set() function (**typecasting**).

my\_list = ["ACDC", "BackInBlack", "ACDC"]

my\_set = set(my\_list)

print(my\_set) # Output: {"ACDC", "BackInBlack"}

## Basic Set Operations

### Adding Elements

* Use the add() method.

A = {"Thriller", "ACDC"}

A.add("inSync")

* Adding the same item again has no effect (no duplicates).

### Removing Elements

* Use the remove() method.

A.remove("inSync")

### Membership Test

* Use the in keyword.

"ACDC" in A # Output: True

"Who" in A # Output: False

## Mathematical Set Operations

### Intersection

* The intersection of two sets contains only the elements present in **both sets**.

AlbumSet1 = {"ACDC", "BackInBlack", "Thriller"}

AlbumSet2 = {"ACDC", "BackInBlack", "The Bodyguard"}

AlbumSet3 = AlbumSet1 & AlbumSet2

print(AlbumSet3) # Output: {"ACDC", "BackInBlack"}

### Union

* The union of two sets contains **all unique elements** from both sets.

AlbumSet1 | AlbumSet2

### Subset

* Check if a set is a subset of another using issubset().

AlbumSet3.issubset(AlbumSet1) # Output: True

## Visualization with Venn Diagrams

* **Sets** can be represented as circles.
* **Intersection**: Overlapping area.
* **Union**: A Combination of both circles.
* **Subset**: One circle fully inside another.

## Summary

* Sets are unordered collections of unique elements.
* They support operations like **add**, **remove**, and **membership checks**.
* Mathematical operations include **intersection**, **union**, and **subset checking**.
* Useful for managing collections with no duplicates and for mathematical computations.

# Module 2 Summary: Python Data Structures

## Tuples

* Ordered and immutable collections of elements.
* Defined using **parentheses ()** with comma-separated values.
* Can include strings, integers, floats, and even nested tuples.
* Access elements using positive and negative indexing.
* Operations include combining, concatenating, and slicing.
* Since tuples are immutable, new tuples must be created for modifications.

## Lists

* Ordered and mutable collections of items.
* Defined using **square brackets []**.
* Can contain mixed data types such as strings, integers, floats, and nested lists.
* Elements are accessed using positive and negative indexing.
* Support operations like adding, deleting, splitting, concatenating, and appending.
* Aliasing occurs when multiple names refer to the same list object.
* Lists can be cloned to create independent copies.

## Dictionaries

* Store **key-value pairs** for flexible data retrieval.
* Defined using **curly brackets {}**.
* Keys must be immutable and unique, while values can be mutable and allow duplicates.
* Key-value pairs are separated by commas.
* Access values using keys.
* Support operations like adding, deleting, and checking for the existence of keys (returns True/False).
* Provide methods to retrieve lists of keys and values.

## Sets

* Unordered collections of **unique elements**.
* Defined using **curly brackets {}**.
* Automatically remove duplicate items.
* A list passed through the set() function generates a set of unique elements.
* Support set operations such as **adding, removing, and membership checks**.
* Mathematical operations include:
  + **Intersection (&)** → Common elements between sets.
  + **Union (|)** → All unique elements from both sets.
  + **Subset check (issubset())** → Determine if one set is contained within another.

## Summary

* **Tuples**: Ordered, immutable, suitable for fixed collections.
* **Lists**: Ordered, mutable, versatile for dynamic collections.
* **Dictionaries**: Key-value storage with unique keys and flexible values.
* **Sets**: Unordered, unique elements, ideal for duplicate removal and set operations.

# Glossary: Python Data Structures

Welcome! This alphabetized glossary contains many of the terms in this course. It also includes additional industry-recognized terms important for working in the industry, participating in user groups, and other certificate programs.

## A

### Aliasing

Aliasing refers to giving another name to a function or a variable.

### Ampersand

A character typically "&" standing for the word "and."

## C

### Compound Elements

Compound statements contain (groups of) other statements; they affect or control the execution of those other statements in some way.

## D

### 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.

## F

### Function

A function is a block of code, defining a set of procedures, which is executed only when it is called.

## I

### Immutable

Immutable objects are of built-in datatypes like int, float, bool, string, Unicode, and tuple. These objects cannot be changed after creation.

### Intersection

The intersection of two sets is a new set containing only the elements that are present in both sets.

## K

### 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.

## L

### 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).

## M

### Mutable

Mutable objects in Python are objects whose values can be changed after they are created. These objects allow modifications such as adding, removing, or altering elements without creating a new object.

## N

### Nesting

A nested function is simply a function within another function and is sometimes called an "inner function."

## R

### Ratings in Python

Ratings in Python typically refer to a numerical or qualitative measure assigned to something to indicate its quality, performance, or value.

## S

### 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 Python language.

## T

### Tuples

Used to store multiple items in a single variable.

### Type Casting

In Python, type casting means converting one data type to another.

## V

### Variables

A variable is a symbolic name or identifier used to store and manipulate data. Variables serve as containers for values of various data types, including numbers, strings, lists, and more.

### Venn Diagram

A graphical representation that uses overlapping circles to illustrate the relationships and commonalities between sets or groups of items.

### Versatile Data

Versatile data refers to data that can be used in multiple ways, adaptable to different applications or purposes, and not restricted to a specific use case.