Huawei HCCDA-AI certification

Trainer: Fawad Bahadur Marwat

2

Python

Python is a high-level, interpreted, general-purpose programming language known for its readability and simplicity.

CORVIT
Reaching end to end

Python

Key characteristics

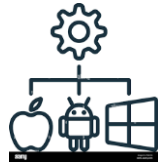
High-Level

Object-Oriented

Vast Ecosystem

Interpreted

Dynamically Typed



Cross-Platform

General-Purpose

Extensible

Large Community

Trainer: Fawad Bahadur Marwat

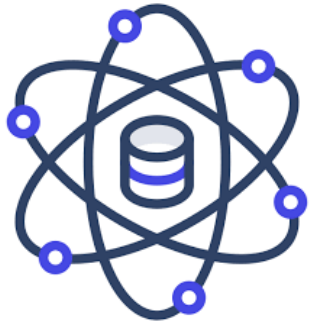
Python Programming

Applications

- Web Development
- Data Science
- Data Analytics
- Machine Learning & AI
- Automation and Scripting
- Software Development and Prototyping
- Desktop GUI applications
- Scientific and Numeric Computing
- Game Development
- Education and Research





CORVIT
Reaching end to end

Setting Up the Python Environment

1

Download Python

Setting Up the Python Environment

1

Download Python

2

Install Python

CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

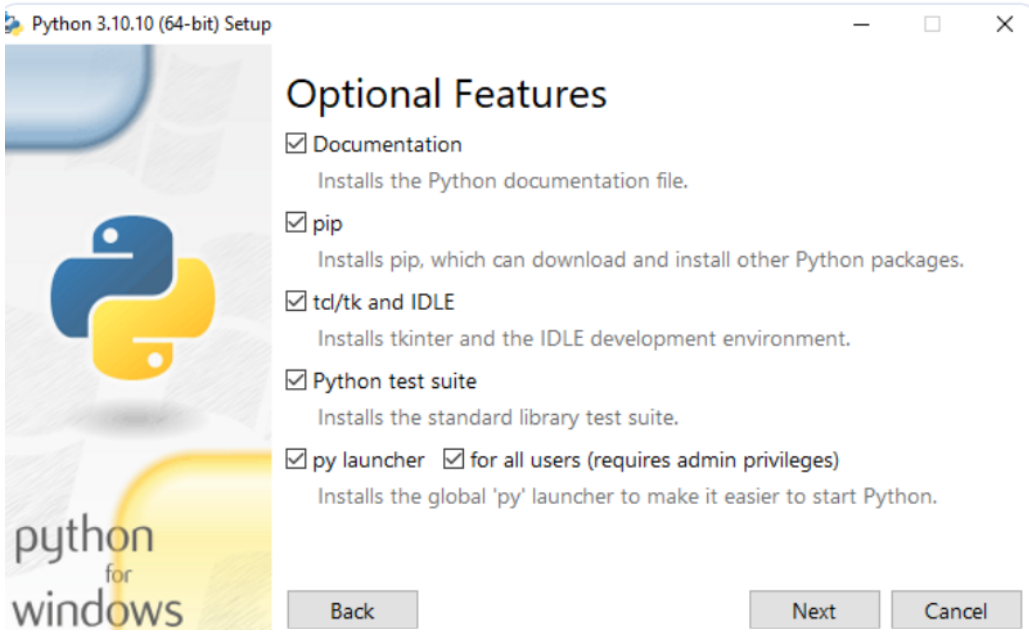
Setting Up the Python Environment

1

Download Python

2

Install Python



CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

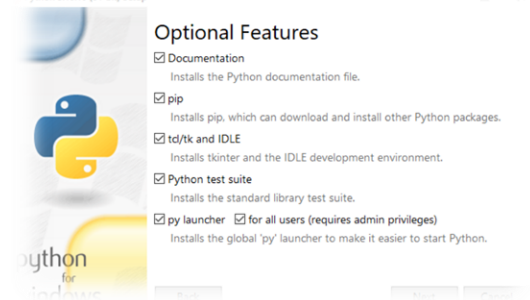
Setting Up the Python Environment

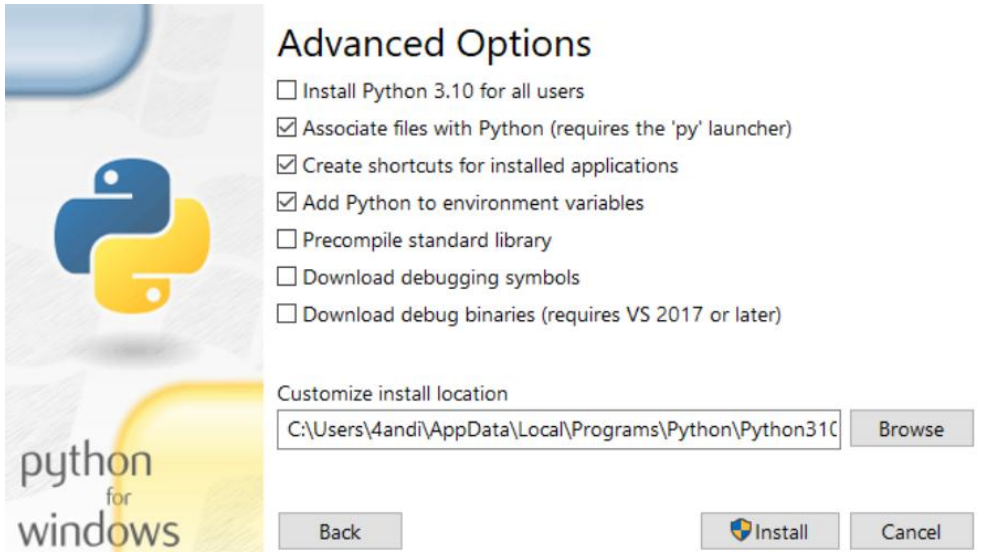
1

Download Python

2

Install Python





CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

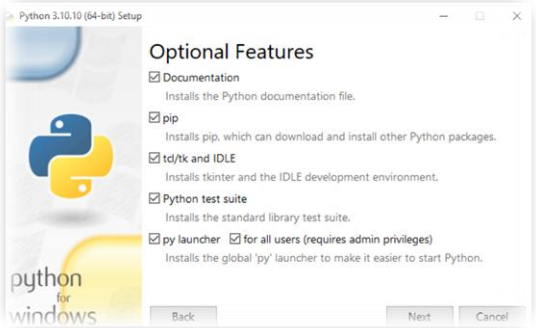
Setting Up the Python Environment

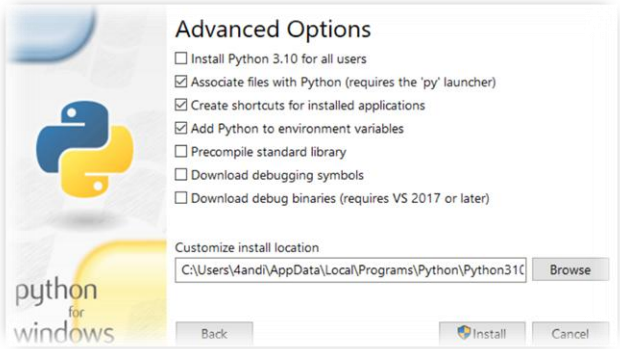
1

Download Python

2

Install Python





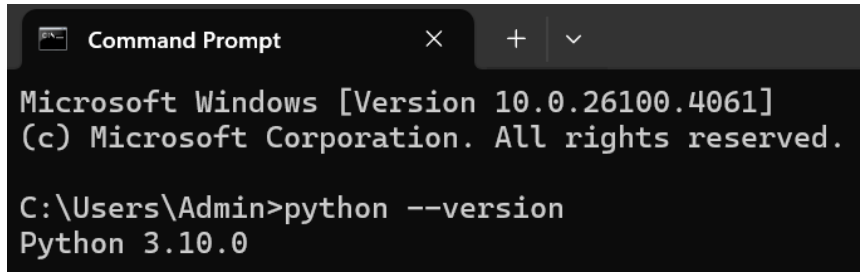
3

Verify the Python Installation

Go to **Start**

Enter **cmd** in the search bar

Click Command Prompt
Enter **python -version.**



CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

Setting Up the Python Environment

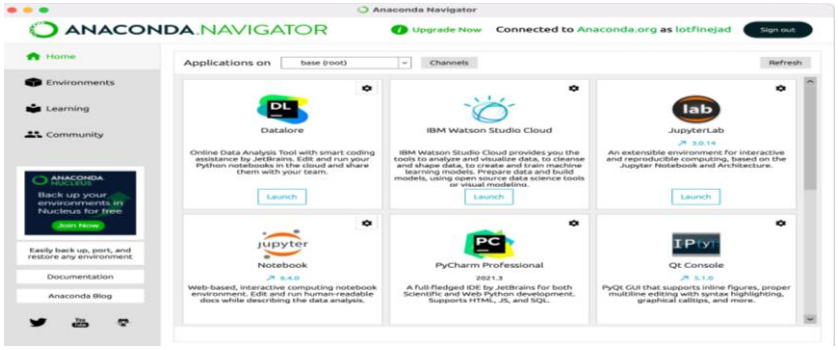
Step 1: Install Python via Anaconda Distribution

Download the Anaconda distribution from the official website.

Follow the installation instructions for your operating system.

Windows/Mac/Linux: Install the setup file and proceed through the default installation options.

Anaconda includes Python, Jupyter Notebook, and essential libraries, making it easier for you to start coding right away.



Step 2: Set Up Jupyter Notebook

Open Anaconda Navigator: Once installed, launch the Anaconda Navigator from your system.

Launch Jupyter Notebook:

In the Anaconda Navigator window, find and click on **Jupyter Notebook** to launch it.

Create a New Notebook:

Jupyter Notebook will open in your web browser.

Click on **New** in the top-right corner and select **Python 3**.

A new notebook will open where you can start writing and running Python code.

Start Coding:

In the new notebook, type your Python code in the cells and press **Shift + Enter** to execute.

CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

Basic Python Syntax

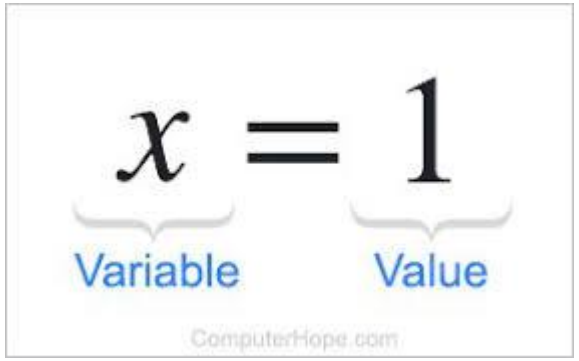
Python Syntax Overview:

Python code is executed line by line (interpreted language).

No need for explicit declaration of variables.

Indentation is crucial in Python (used for defining blocks of code).

Case-sensitive language (e.g., Variable and variable are different).



Basic Syntax Elements:

Variables: Names assigned to values (e.g., x = 10).

Operators: Symbols for performing operations (e.g., +, -, *, /).

Comments: Notes in the code for clarification (use # for single-line).

Functions: Defined blocks of reusable code (e.g., def my_function():).

CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

Writing Your First Python Program

"Hello, World!" Example:

Explanation of the print() function.

Writing and executing your first Python program.

Understanding Python Syntax:

Code structure in Python.

Code Cells vs. Markdown Cells in Jupyter Notebook:

Difference between code and markdown cells.

Adding notes and explanations in markdown.


```
print("Hello, World!")
```

CORVIT
Reaching end to end

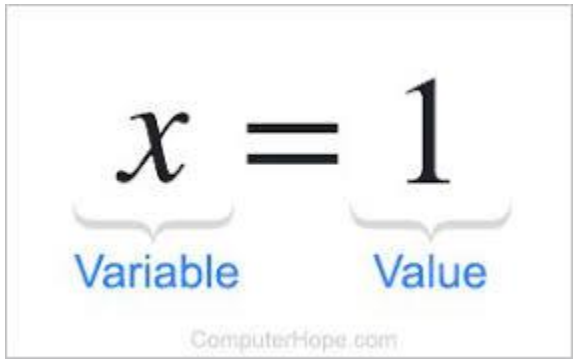
Trainer: Fawad Bahadur Marwat

Variable

A variable is a named container that stores data in memory.

Think of it like a labeled box where you can store values.

You can assign values using = and reassign them later.

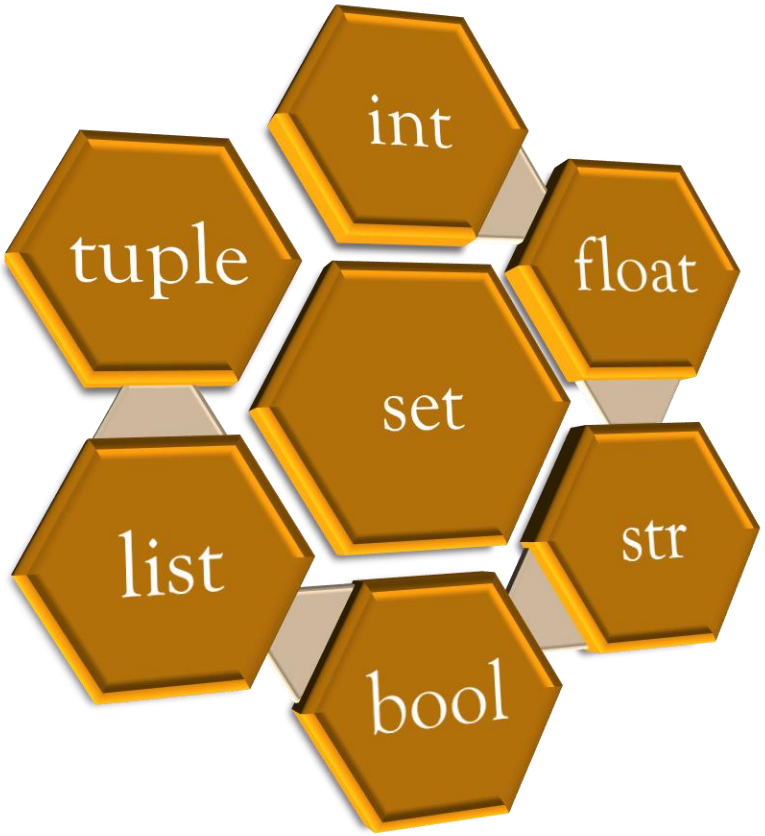



```
message = "Hello, Python!"  # Assign a string
print(message)              # Output: Hello, Python!

message = "New message"     # Reassign
print(message)              # Output: New message
```

CORVIT
Reaching end to end

Data Types



Integer

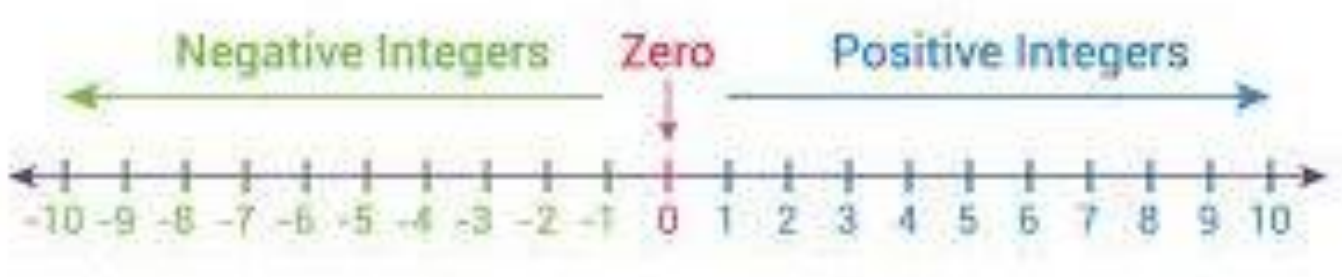
Integers are whole numbers, either positive, negative, or zero.

Used for counting, indexing, and arithmetic operations

Cannot have decimal or fractional parts

Commonly used in loops, indexing arrays, and mathematical computations

Examples of integers: -10, 0, 15



CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

Float

Floats are numbers with decimal points, representing fractional values.

Useful for precise calculations (e.g., financial transactions, measurements)

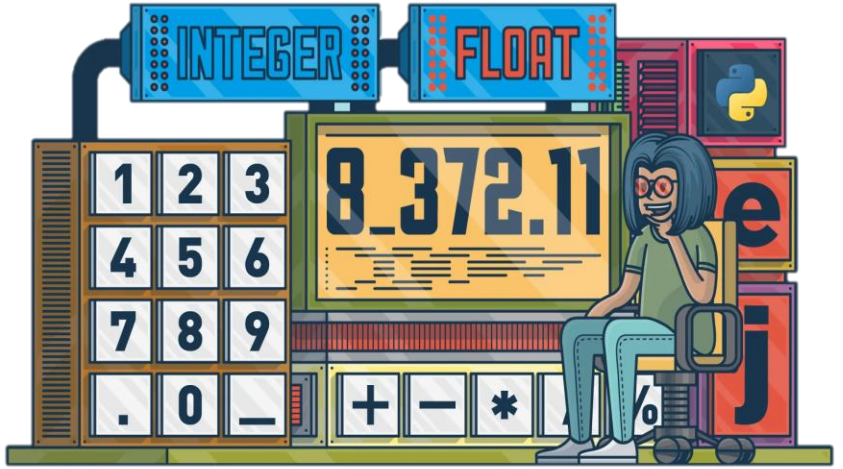
Can represent large or small numbers using scientific notation (e.g., 1.23e4)

Can be positive or negative

Floats take up more memory than integers because of the decimal precision

Example:

price = 19.99 Represents the price of an item



NumericVariables.py

C: > ProCodeGuide > Python > sources > Numer

```
1  variable1 = 1.98
2  variable2 = 454.567
3  variable3 = -284902.8277
4  variable4 = 8e2
5  print(type(variable1))
```

CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

String

Strings are sequences of characters used to store and manipulate text.

Can contain letters, numbers, symbols, and spaces

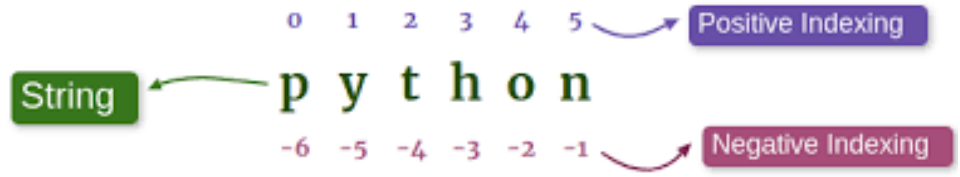
Enclosed in either single (') or double (") quotes

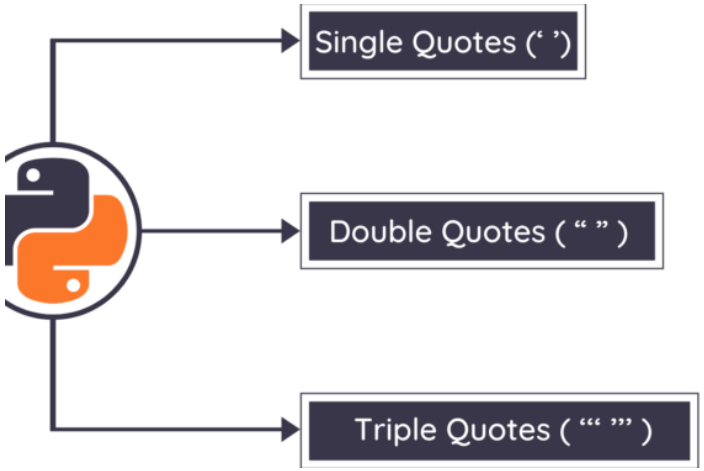
Strings are immutable: once created, their content cannot be changed

Commonly used for names, messages, filenames, etc.

Supports various operations like concatenation (+), slicing ([]), and formatting (f-strings)

Example: greeting = "Hello, world!" # A basic greeting message





CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

Boolean (bool)

Booleans represent one of two possible values: True or False.

Used in conditional statements (e.g., if, while) to control program flow

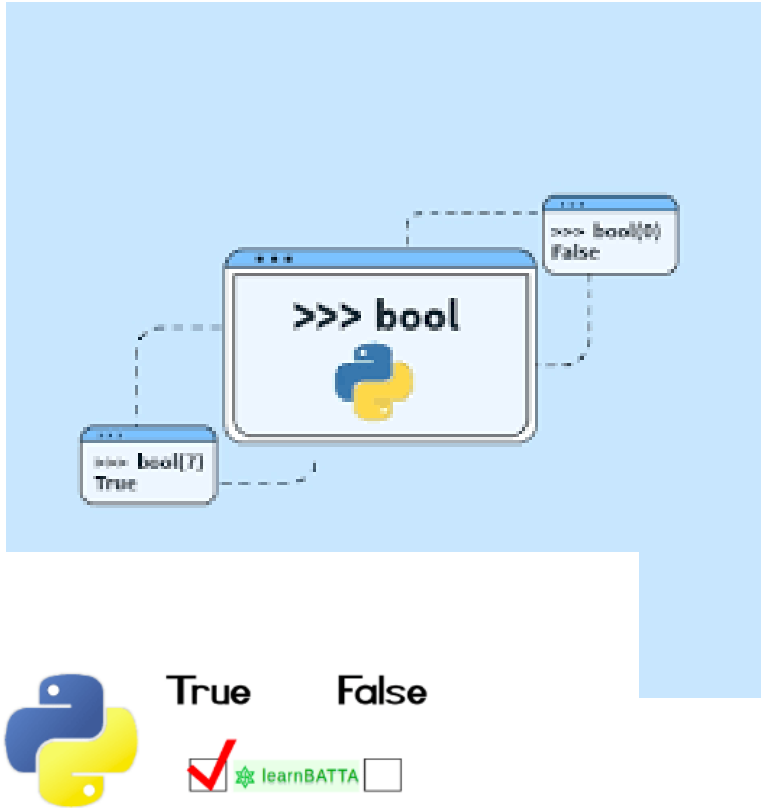
Result of logical operations (e.g., 5 > 3 is True)

Can be converted from other data types: bool(0) is False, bool(1) is True

Often used to represent states, such as on/off, open/closed

Example:

```
is_logged_in = True  # Indicates if a user is logged in
```



Working with boolean data type in python

CORVIT
Reaching end to end

List

Lists are ordered collections of items that can store different data types.

Mutable: You can add, remove, or modify elements in a list

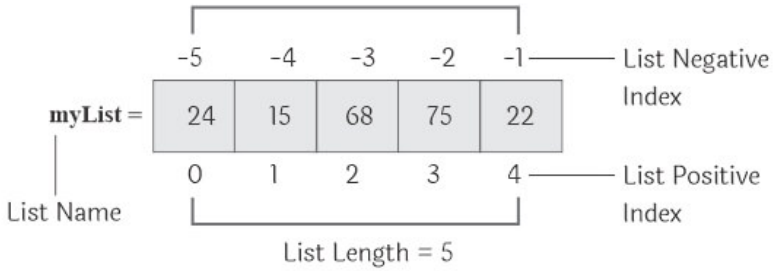
Lists are indexed, starting from 0

Supports various operations such as appending (.append()), removing (.remove()), and sorting (.sort())

Can store different data types: integers, strings, even other lists

Commonly used to store sequences of items like names, numbers, or objects

Example: fruits = ["apple", "banana", "cherry"] # A list of fruits



CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

Tuple

Tuples are ordered, immutable collections of items, meaning their values cannot be changed once assigned.

Tuples are useful for fixed data sets that should not be altered

Like lists, tuples are indexed starting from 0

Can contain mixed data types, including integers, floats, strings, or other tuples

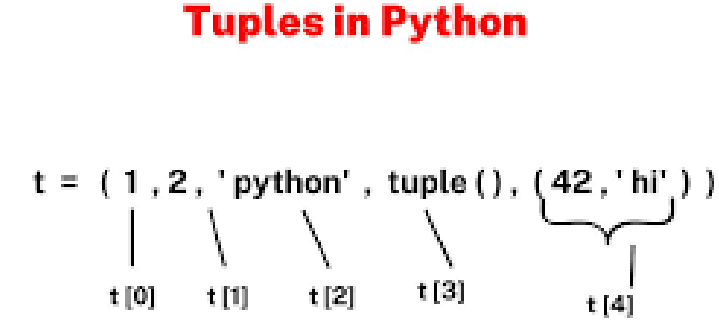
Tuples can be unpacked into separate variables: x, y = (10, 20)

Commonly used for storing related pieces of data, like coordinates or color values

Example: coordinates = (34.0522, -118.2437) # Latitude and longitude of a location

```python
my_tuple = ("dog", 4.5, True, 7, "apple")
print(my_tuple)
# ('dog', 4.5, True, 7, 'apple')

my_second_tuple = tuple(["dog", 4.5, True, 7, "apple"])
print(my_second_tuple)
# ('dog', 4.5, True, 7, 'apple')
```



CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

Dictionary

Dictionaries store data in key-value pairs, allowing fast lookups by unique keys.

Keys must be unique and immutable (e.g., strings, numbers, tuples)

Values can be of any data type and can be changed (mutable)

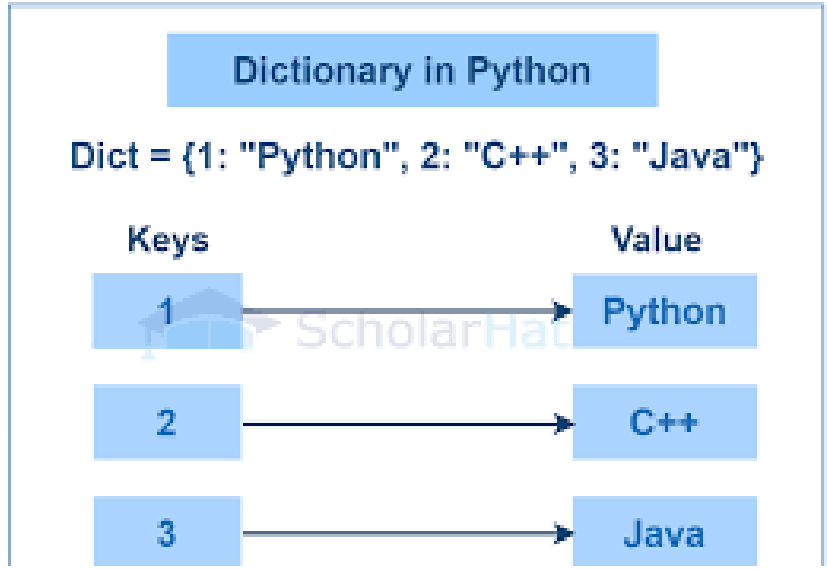
Commonly used for structured data like user profiles, settings, or JSON data

Supports operations like adding new key-value pairs (dict['key'] = value) and removing keys (del dict['key'])

Keys are accessed quickly compared to lists due to the dictionary's hashing mechanism

Example:

student = {"name": "Alice", "age": 22} # A dictionary representing a student




```
# Create a dictionary

my_dict = {'Alex': 5,
           'Ben' : 10,
           'Carly': 12,
           'Danielle': 7,
           'Evan' : 6}
my_dict
```

```
{'Alex': 5, 'Ben': 10, 'Carly': 12, 'Danielle': 7, 'Evan': 6}
```

CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

Comparison of Mutability and Use Case

Data Type	Example	Mutable	Use Case
Int	x = 5	-	Whole numbers
Float	y = 3.24	-	Decimals
Str	s = ‘text’	❌	Text
Bool	Flag = True	-	True/False
List	[1,2,3]	☑	Modifiable collection
Tuple	(1,2)	❌	Fixed Collection
Dict	{“key” : “value”	☑	Key-Value Pairs
Set	{1,2,3}	☑	Unique elements