# PYTHON ULTIMATE NOTES

# Contents


## 1. INTRODUCTION

- What is Python?
- Why Learn Python? (Popularity, Applications)
- Installation & Setup (Python, VS Code)
- Running Python Programs

## 2. PYTHON BASIC

- Variables & Data Types
- Input & Output
- Operators (Arithmetic, Logical, Comparison, Assignment)

## 3. CONTROL FLOW

- Conditional Statements (if, elif, else)
- Loops (for, while)
- Break, Continue, Pass

## 4. FUNCTION

- Defining & Calling Functions
- Arguments & Return Values
- Lambda Functions
- Scope of Variables (Local & Global)

## 5. Data Structures

- Lists (Methods, Slicing, Looping)
- Tuples (Immutable Data)
- Sets (Unique Values)
- Dictionaries (Key-Value Pairs, Methods)

## 6. Object-Oriented Programming (OOP)

- Classes & Objects
- Constructors (init method)
- Inheritance & Polymorphism
- Encapsulation & Abstraction

## 7. File Handling

- Reading & Writing Files
- Working with CSV & JSON

## 8. Exception Handling

- Try, Except, Finally
- Raising Custom Errors

## 9. Modules & Libraries

- Importing Modules
- Built-in vs. External Libraries
- Using math, random, datetime

## 10. Database Connectivity

- Connecting Python with MySQL or SQLite
- CRUD Operations (Create, Read, Update, Delete)

## 11. Web Scraping

- Using requests and BeautifulSoup
- Extracting Data from a Website

## 12. Data Science & Machine Learning (Intro)

- Using pandas, numpy, matplotlib
- Basic Data Visualization
- Introduction to scikit-learn

## 13. Two Mini Projects

- Project 1: To-Do List App (Console-Based)
- Users can add, remove, and mark tasks as completed
- Data stored in a text file or JSON

- Project 2: Weather App Using API
- Fetch weather data from an API (like OpenWeatherMap)
- Display temperature, humidity, and weather conditions
- Simple GUI using tkinter (optional)

# 1. INTRODUCTION

## 1. What is Python?

Python is a high-level, interpreted programming language known for its simplicity and readability. It is widely used for web development, data science, automation, AI/ML, game development, cybersecurity, and more.

## 2. Who Created Python & When?

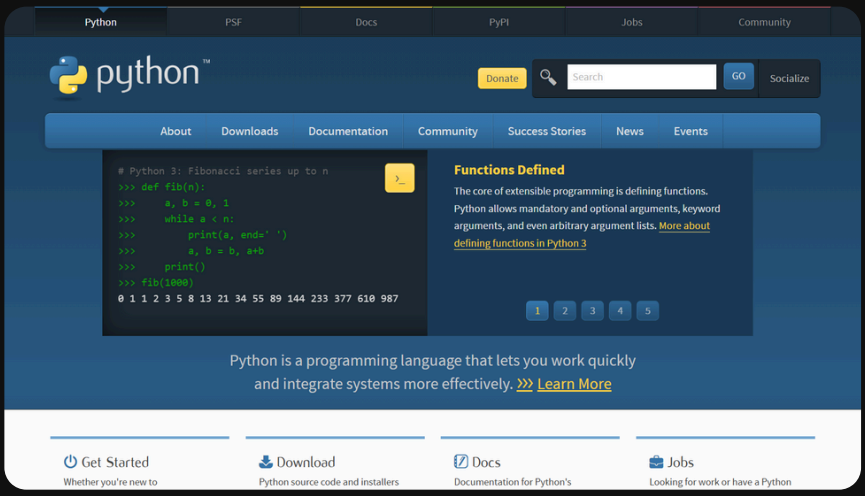
Python was created by Guido van Rossum in 1989 and officially released in 1991.

## 3. Why Guido van Rossum Create a Python?

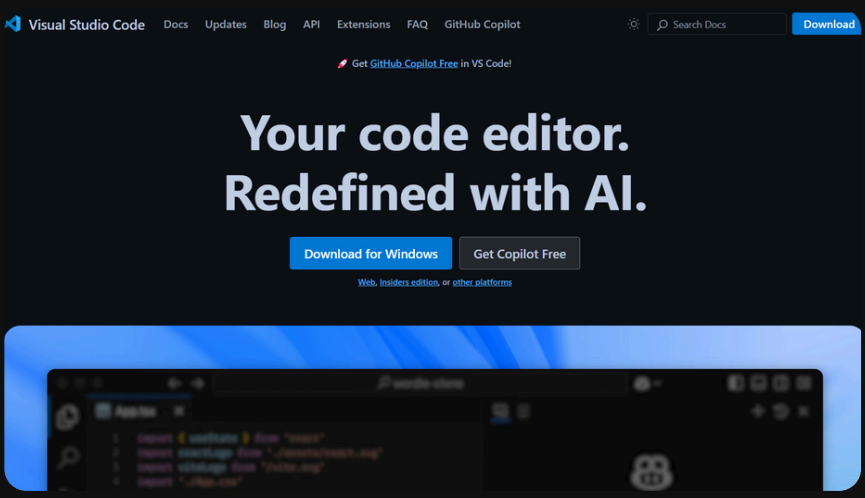
Guido wanted to create a simple, easy-to-learn programming language as an alternative to ABC language (which was complex). He named it "Python" after the British comedy show "Monty Python's Flying Circus", not the snake!

# Installation & Setup

## First we are going to download python

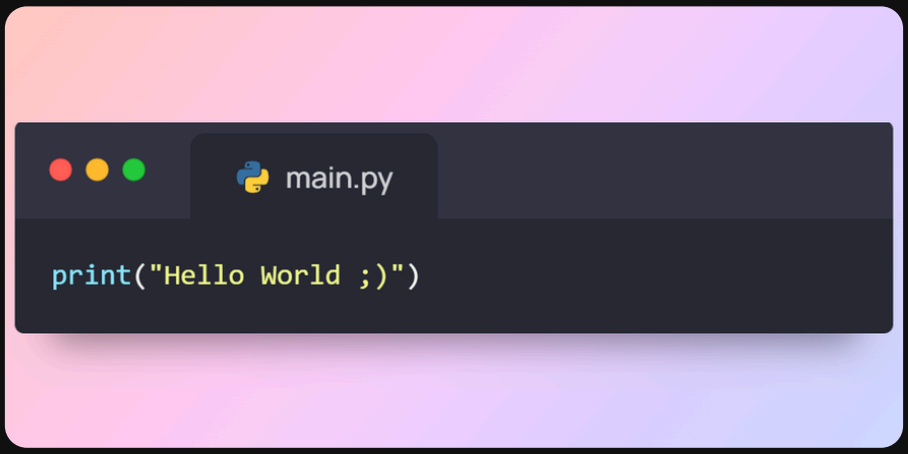



## Second we are going to download VS Code



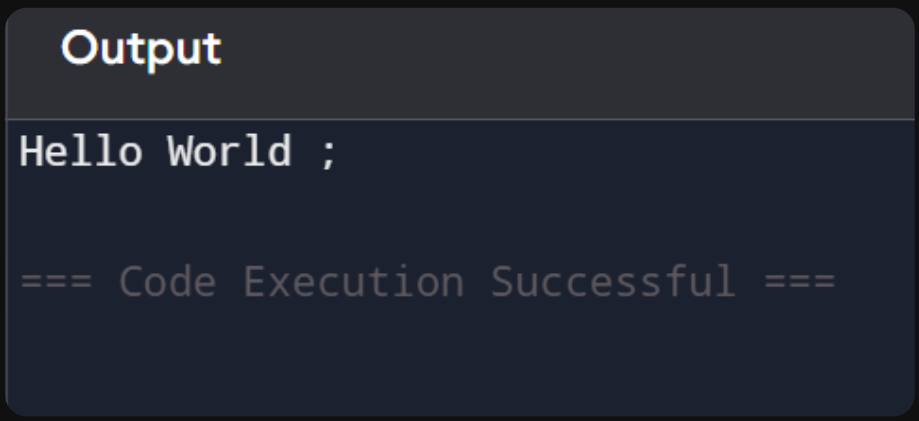
# Let's write your first program

**Input:-**




**In this program we are just simply printing this hello world program using python**

**Output:-**



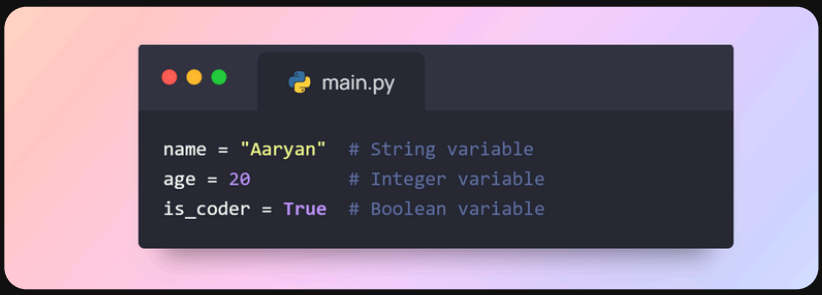

**So here you can see Hello World ; is printed successfully**

# 2. PYTHON BASIC

## Variables & Data Types

### 1. Variables

**A variable is a named storage location in memory that holds a value, which can change during program execution. Variables allow programmers to store, retrieve, and manipulate data efficiently.**




**Here, name, age, and is_coder are variables storing different types of data.**

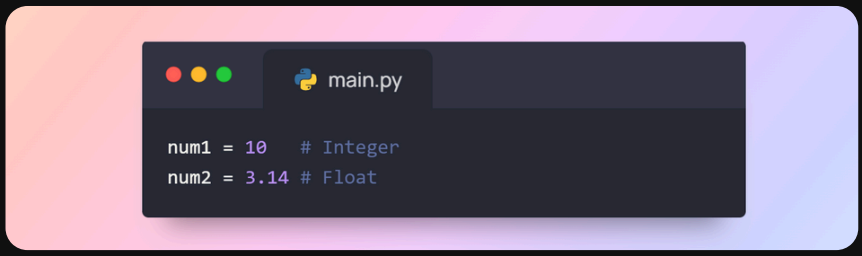
### 2. Data Types

**Data types define the type of data a variable can hold. Different programming languages have different data types, but the common ones include:**

- **Numeric Data Type**
- **String**
- **Boolean**
- **List**
- **Tuple**
- **Dictionary**
- **Set**

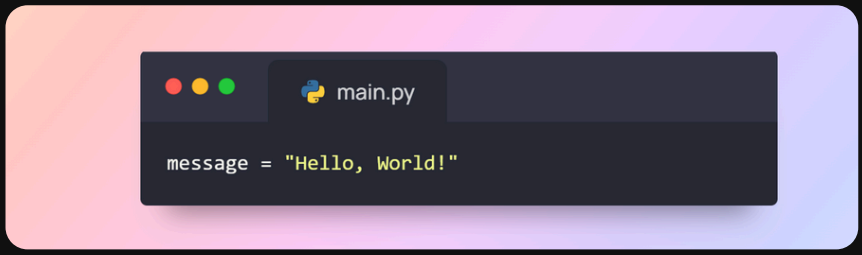
# Numeric Data Type

- **Integer (int): Whole numbers (e.g., 5, -10, 1000)**
- **Floating Point (float): Numbers with decimal points (e.g., 3.14, -0.99, 2.0)**
- **Complex (complex): Numbers with real and imaginary parts (e.g., 3 + 5j in Python)**



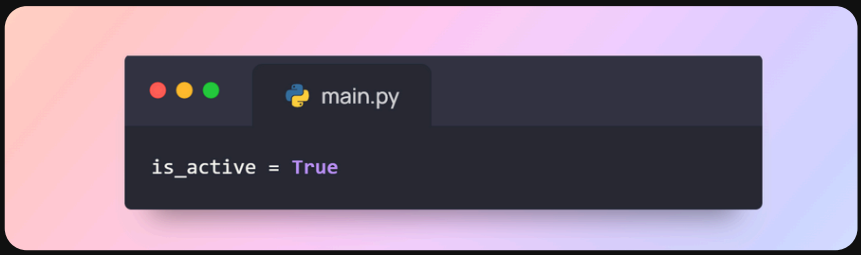

# String

- **A sequence of characters enclosed in quotes.**



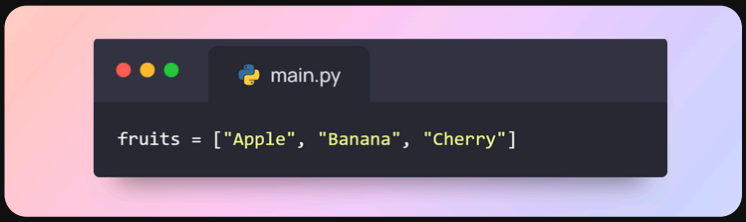

# Boolean (bool)

- **Represents True or False values.**



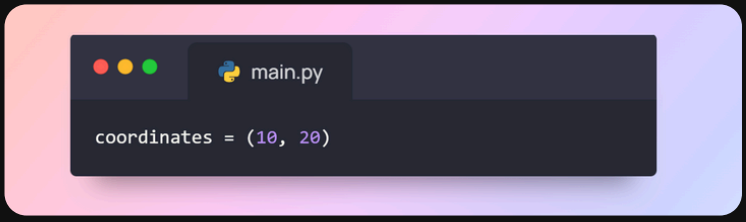
# List (Array in JavaScript)

- **Ordered, mutable collection of elements.**




# Tuple

- **Similar to a list but immutable (cannot be changed).**




# Dictionary (dict)

- **Stores key-value pairs.**


# Set

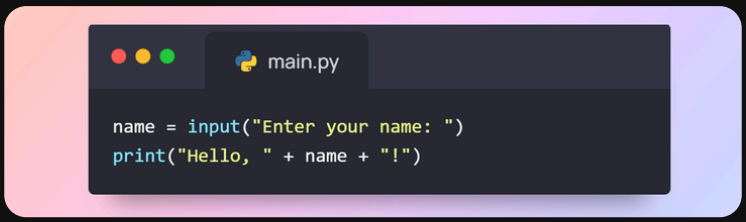
- **Unordered collection of unique elements.**

# Input & Output

**In Python, input and output operations are performed using built-in functions like input(), print(), and file handling methods.**

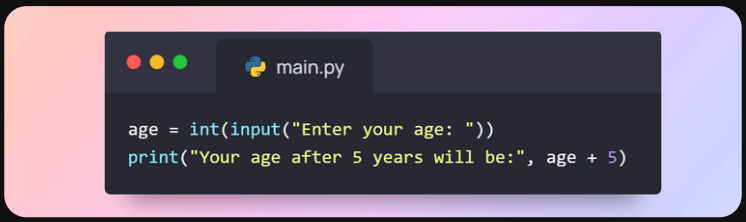
## 1. Input in Python

**Python provides the input() function to take input from the user.**

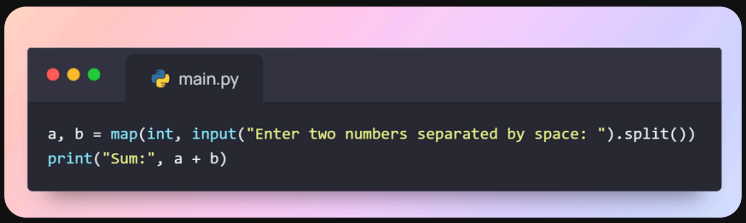



**By default, input() takes input as a string.**

### Example: Taking integer input




### Example: Taking multiple inputs



# 2. Output in Python

**Python uses the print() function to display output.**

## Example: Simple output

main.py

```
print("Hello World ;)")
```

## Example: Printing multiple values  pytho

main.py

```
name = "Aaryan"
age = 20
print("My name is", name, "and I am", age, "years old.")
```

## Using sep and end parameters

main.py

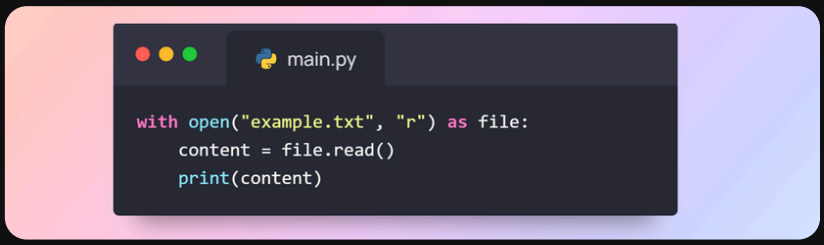
```
print("Python", "is", "fun", sep="-")  # Output: Python-is-fun
print("Hello", end=" ")
print("World!")  # Output: Hello World!
```

# 3. File Input & Output in Python

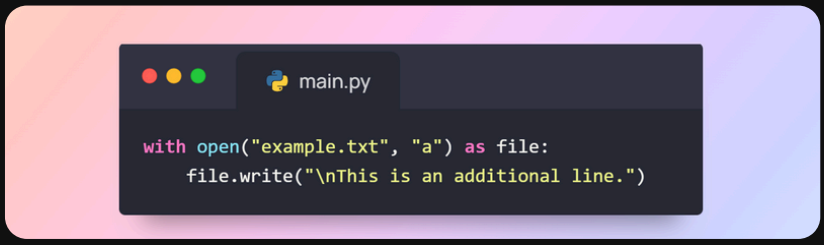
Python allows reading and writing files using open().

## Writing to a file


## Reading from a file

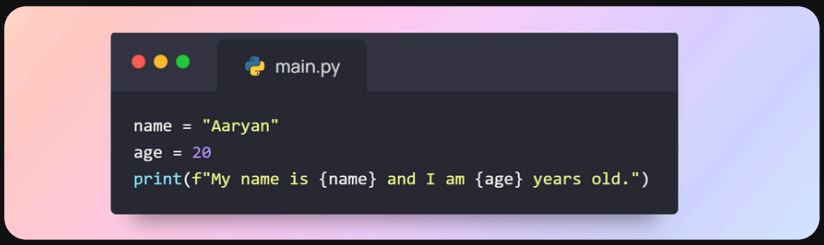



## Appending to a file python

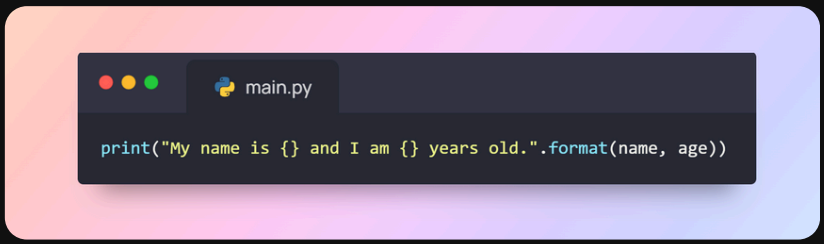


# 4. Formatted Output

**Python provides f-strings for formatted output.**




## Using format() method




## Conclusion

✅ Use input() to get user input.
✅ Use print() to display output.
✅ Use open() for file handling.
✅ Use f-strings for formatted output.

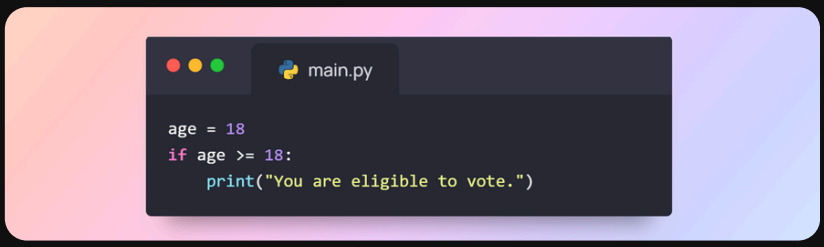
# 3. Control Flow

## 1. Conditional Statements (if, elif, else)

**Conditional statements allow a program to make decisions based on conditions.**

### if Statement

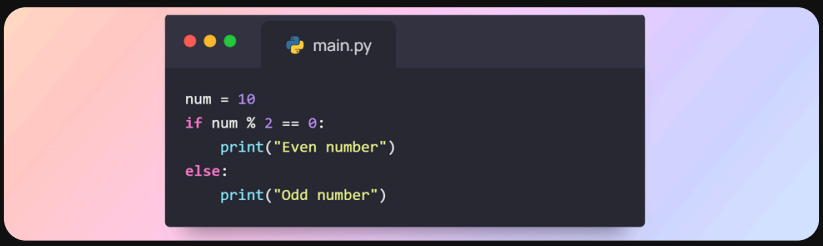
**Executes a block of code if a condition is True.**




➡ **If age is 18 or more, the message is printed.**

### if-else Statement

**Executes one block of code if True, another if False.**



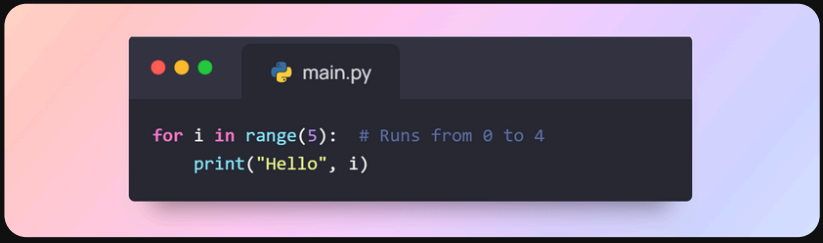

➡ **If num is divisible by 2, it prints "Even number"; otherwise, "Odd number".**

# Loops in Python

**Loops help execute a block of code multiple times.**

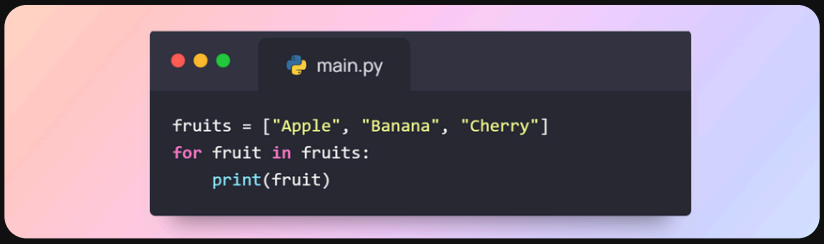
## 👉 for Loop

**Used to iterate over a sequence (list, string, range, etc.)**




➡ **range(5) generates numbers from 0 to 4.**

## Looping through a list:




## 👉 while Loop

**Repeats as long as the condition is True.**


➡ **Runs while count is ≤ 5.**

# Break, Continue, Pass

## 👉 break Statement

**Exits the loop immediately.**

main.py

```python
for i in range(10):
    if i == 5:
        break  # Stops loop at 5
    print(i)
```

➡ **The loop stops when i reaches 5.**

## 👉 continue Statement

**Skips the current iteration and moves to the next.**

main.py

```python
for i in range(5):
    if i == 3:
        pass  # Placeholder, does nothing
    print(i)
```

➡ **The number 2 is skipped in the output.**

## 👉 pass Statement

**A placeholder that does nothing (used when a block is required but not written yet).**

main.py

```python
for i in range(5):
    if i == 2:
        continue  # Skips 2
    print(i)
```

➡ **pass is used when a statement is required but not implemented.**

# Conclusion

✅ if-elif-else → Decision making
✅ for loop → Iterates over a sequence
✅ while loop → Runs while the condition is True
✅ break → Exits loop
✅ continue → Skips current iteration
✅ pass → Does nothing (placeholder)

# 4. Functions

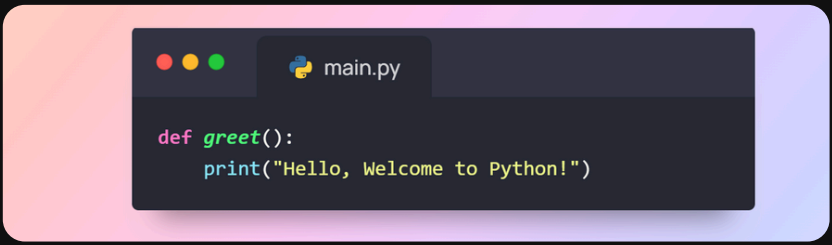
## Functions in Python

**Functions are reusable blocks of code that perform a specific task. They help make programs modular and reduce repetition.**

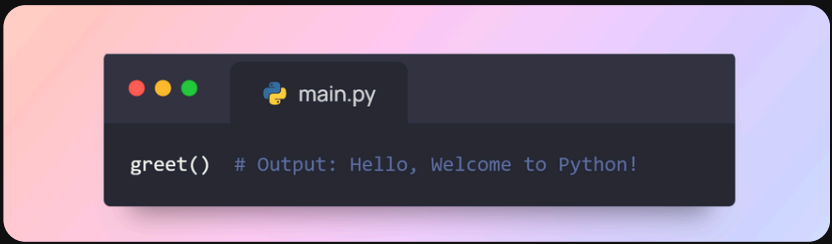
### 1️⃣ Defining & Calling Functions

**A function is defined using the def keyword.**

### 👉 Defining a Function




### 👉 Calling a Function



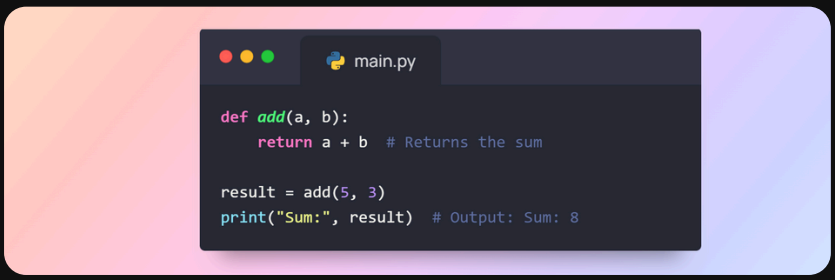

➡ **The function is executed when called.**

# 2 Arguments & Return Values

## 👉 Function with Parameters

**A function can take inputs called parameters.**

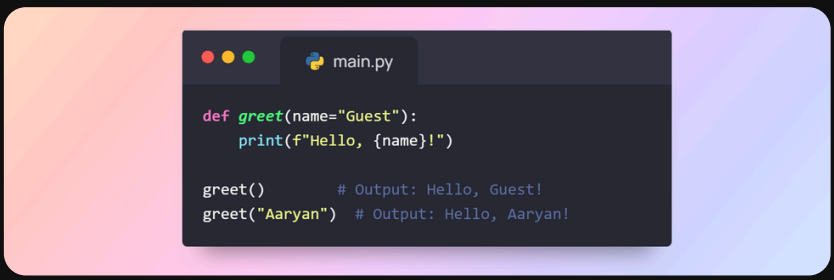

## 👉 Function with Multiple Parameters




➡ **return sends a value back to the caller.**

## 👉 Default Parameter Value

**If no argument is passed, it uses the default value python**



## 👉 Keyword Arguments (Named Parameters)

**You can pass arguments by name, making them more readable.**

main.py

```python
def introduce(name, age):
    print(f"My name is {name} and I am {age} years old.")

introduce(age=20, name="Aaryan")
```

➡ **Order doesn't matter when using keyword arguments.**

## 👉 Variable-Length Arguments (*args and **kwargs)

***args → Allows multiple positional arguments.**

****kwargs → Allows multiple keyword arguments.**

main.py

```python
def sum_all(*numbers):
    return sum(numbers)

print(sum_all(1, 2, 3, 4))  # Output: 10
```

main.py

```python
def print_details(**info):
    for key, value in info.items():
        print(f"{key}: {value}")

print_details(name="Aaryan", age=20, country="India")
```

➡ ***args collects multiple values as a tuple.**

➡ ****kwargs collects key-value pairs as a dictionary.**

## 3 Lambda Functions (Anonymous Functions)

**A lambda function is a small anonymous function with a single expression.**

👉 **Example: Lambda Function**


👉 **Example: Lambda with Multiple Arguments**


➡ **Lambda functions are useful for short, single-use operations.**

# 4 Scope of Variables (Local & Global)

**Scope determines where a variable can be accessed**

## 👉 Local Variable (Exists only inside a function)

main.py

```python
def my_function():
    x = 10  # Local variable
    print(x)

my_function()
# print(x)  # ❌ This will cause an error (x is not defined outside the function)
```

➡ x is inside the function and can't be accessed outside.

## 👉 Global Variable (Accessible everywhere)

main.py

```python
x = 50  # Global variable

def my_function():
    print(x)  # Can access global variable

my_function()
print(x)  # Output: 50
```

## 👉 Modifying Global Variables inside Functions

**Use the global keyword if you need to modify a global variable inside a function.**


**➡ Without global, Python will create a new local variable instead of modifying the global one.**

# Conclusion

- ✅ **Functions → Reusable code blocks**
- ✅ **Arguments → Inputs to functions**
- ✅ **return → Sends a value back**
- ✅ **Lambda → Short anonymous functions**
- ✅ **Scope → Defines where a variable is accessible**

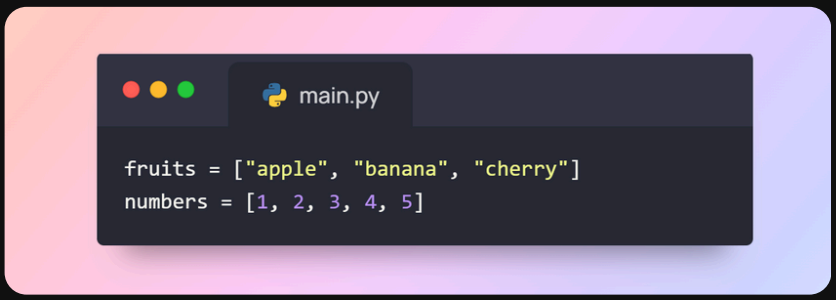
# 5. Data Structures

**Python has four main built-in data structures: Lists, Tuples, Sets, and Dictionaries. These help in storing and managing data efficiently.**

## 1. Lists (Ordered, Mutable)

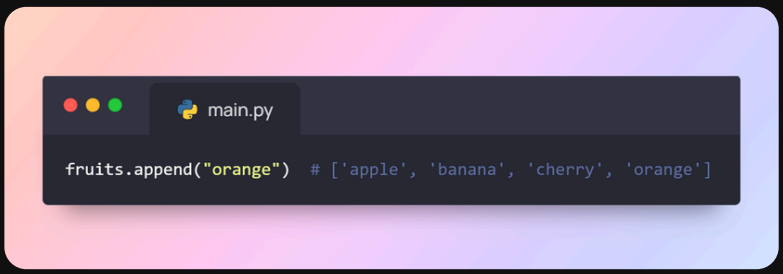
**A list is a collection of items that are ordered and changeable (mutable). Lists allow duplicate values.**

### Creating a List

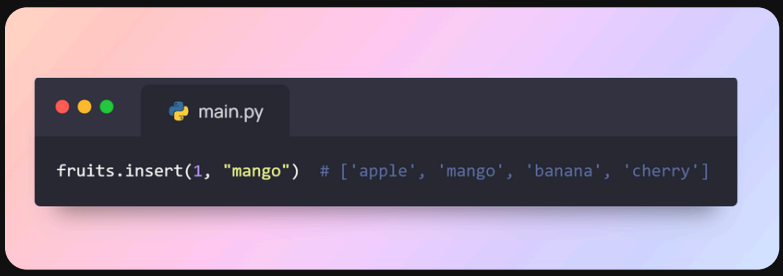



### List Methods

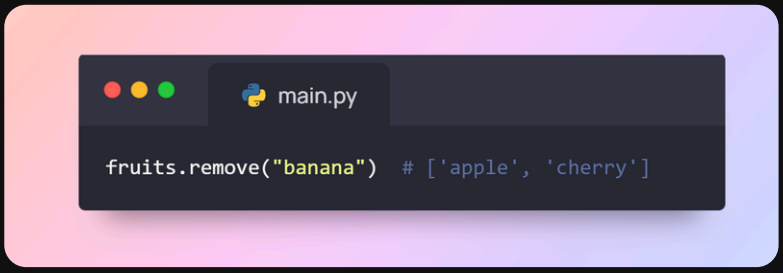
**Append: Adds an item to the end**



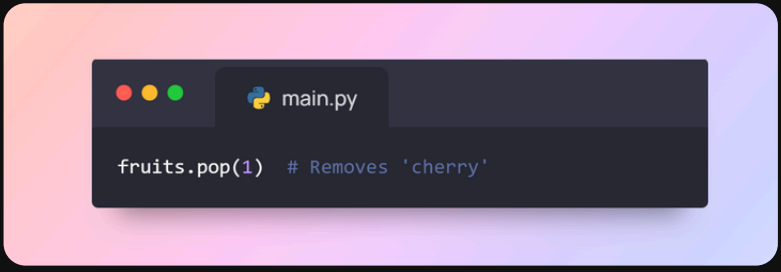
## Insert: Adds an item at a specific index



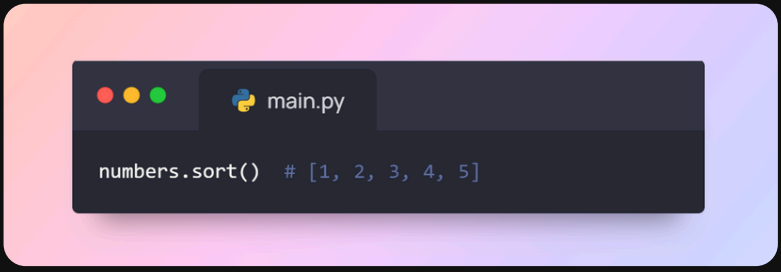

## Remove: Removes a specific item




## Pop: Removes an item by index




## Sort: Sorts the list in ascending order

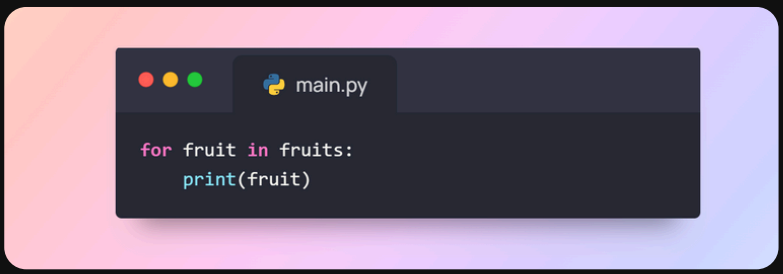


### Reverse: Reverses the list


# Slicing Lists

## Extracting a part of a list using start:end:step

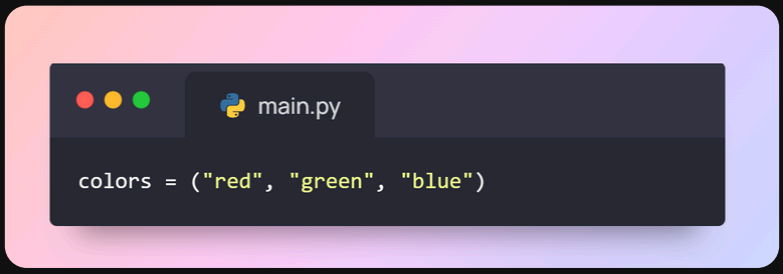

## Looping Through a List



# 2. Tuples (Ordered, Immutable)

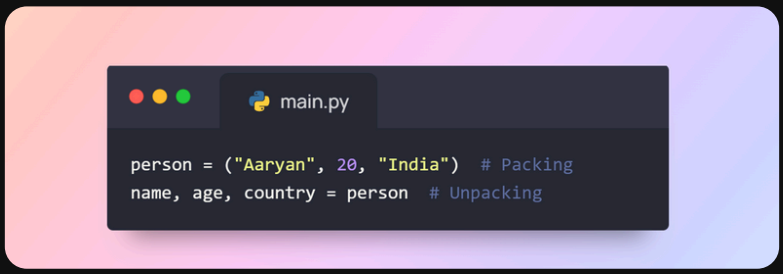
**A tuple is like a list, but it cannot be changed (immutable). Tuples are used for fixed data.**

## Creating a Tuple




## Accessing Elements python


## Tuple Packing & Unpacking




## Why Use Tuples?

- **Faster than lists**
- **Used for fixed values (e.g., coordinates, database records)**