Chapter 11: Python Programming

Introduction

Python is a high-level, interpreted programming language known for its simplicity and readability. In the field of Artificial Intelligence (AI), Python is widely used because of its clear syntax and the vast number of libraries available for machine learning, data processing, and automation.

In this chapter, you will learn the fundamentals of Python programming that form the building blocks of developing AI applications. This includes understanding variables, data types, operators, control structures, functions, and working with simple input-output operations.

11.1 Introduction to Python

- Developed by Guido van Rossum in the late 1980s.
- Python is open-source and supported by a vast developer community.
- Key features:
    - Easy to read and write
    - Interpreted and dynamic
    - Extensive libraries for AI, data science, and automation (like NumPy, Pandas, Scikit-learn)

11.2 Setting up Python Environment

- **Installation**: Use python.org or install Anaconda (comes with pre-installed libraries).
- **IDEs**:
    - IDLE (comes with Python)
    - Jupyter Notebook
    - VS Code / PyCharm

11.3 Python Syntax Basics

- Python uses **indentation** instead of braces `{}`.
- Case-sensitive language.

Example:

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

11.4 Variables and Data Types

Variables

- Containers to store data values.
- Declared directly (no need to mention data type):

  ```
  x = 5
  name = "AI"
  ```

Data Types

- **Numeric**: int, float
- **String**: str
- **Boolean**: bool (True/False)
- **List, Tuple, Dictionary** (introduced briefly here)

11.5 Operators in Python

- **Arithmetic Operators**: `+, -, *, /, //, %, **`
- **Relational Operators**: `==, !=, >, <, >=, <=`
- **Logical Operators**: `and, or, not`
- **Assignment Operators**: `=, +=, -=`, etc.

Example:

```
x = 10
y = 5
print(x + y)  # 15
```

11.6 Input and Output

Input

- Takes input from the user using `input()` function.

```
name = input("Enter your name: ")
```

Output

- Prints data to the screen using `print()` function.

```
print("Welcome", name)
```

11.7 Conditional Statements

Used to perform different actions based on different conditions.

if Statement

```
if age > 18:
    print("Eligible to vote")
```

if-else Statement

```
if age >= 18:
    print("Adult")
else:
    print("Minor")
```

if-elif-else Ladder

```
if marks >= 90:
    print("A Grade")
elif marks >= 75:
    print("B Grade")
else:
    print("Needs Improvement")
```

11.8 Loops in Python

for Loop

Used to iterate over a sequence (like list, string, range):

```
for i in range(5):
    print(i)
```

while Loop

Runs as long as a condition is true.

```
count = 1
while count <= 5:
    print(count)
    count += 1
```

11.9 Functions

Functions are reusable blocks of code.

Defining a Function

```
def greet(name):
    print("Hello", name)
```

Calling a Function

```
greet("AI Learner")
```

11.10 Lists (Introduction)

Lists are ordered, mutable collections of items.

```
fruits = ["apple", "banana", "cherry"]
print(fruits[0])  # apple
```

11.11 Error Handling (Basic)

- Common errors: SyntaxError, NameError, TypeError
- Use `try-except` block to handle exceptions.

```
try:
    print(10 / 0)
except ZeroDivisionError:
    print("Cannot divide by zero!")
```

11.12 Python Libraries for AI (Brief Overview)

Though covered in detail in later chapters, here's a sneak peek:

- **NumPy** – for numerical operations
- **Pandas** – for data analysis
- **Matplotlib** – for data visualization
- **Scikit-learn** – for machine learning

Summary

In this chapter, you learned the basics of Python programming, an essential skill for working in AI. You explored how to write Python code using variables, data types, operators, control structures (like if-else and loops), functions, and basic error handling. Python's simplicity and wide support make it ideal for building AI projects. Mastery of these foundational topics is essential before progressing to more advanced AI applications.
