Chapter 10: Introduction to Python

🔰 Introduction

Python is a high-level, versatile, and beginner-friendly programming language widely used in artificial intelligence, data science, web development, automation, and many other fields. Its simple syntax and readability make it ideal for students starting their programming journey. This chapter introduces the basics of Python to help you understand how to write, execute, and debug simple Python programs.

🔍 10.1 What is Python?

- **High-level**: Abstracts many complex details of the computer.
- **Interpreted**: Code is executed line by line using an interpreter.
- **Dynamic typing**: No need to declare data types explicitly.
- **Open-source**: Free to use and has a large supportive community.

Applications of Python:

- Artificial Intelligence & Machine Learning
- Web development (using frameworks like Django and Flask)
- Automation and scripting
- Data analysis and visualization
- Game development

10.2 Setting up Python

1. **Download Python** from the official website: https://www.python.org
2. Install Python (ensure the “Add Python to PATH” checkbox is ticked).
3. Use **IDLE**, **VS Code**, or **Jupyter Notebook** to write and run Python programs.

✍ 10.3 Writing Your First Python Program

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

- **print()** is a built-in function that outputs text to the screen.

▯ 10.4 Python Syntax Basics

10.4.1 Variables and Data Types

```
name = "Alice"          # String
age = 14                # Integer
height = 5.3            # Float
is_student = True       # Boolean
```

- Python uses dynamic typing.
- Variable names are case-sensitive and should be descriptive.

10.4.2 Comments

- **Single-line comment**: starts with #
- **Multi-line comment**: enclosed in triple quotes (`''' '''` or `""" """`)

```
# This is a single-line comment
'''
This is a
multi-line comment
'''
```

🔄 10.5 Operators in Python

Operator Type	Example	Description
Arithmetic	`+ - * / % **`	Basic math operations
Assignment	`= += -=`	Assign or update values
Comparison	`== != > <`	Compare values (returns bool)
Logical	`and or not`	Combine conditional statements

Example:

```
a = 10
b = 5
print(a + b)        # Output: 15
print(a > b)        # Output: True
```

🔁 10.6 Control Structures

10.6.1 Conditional Statements

```
age = 18
if age >= 18:
    print("You can vote")
```

```
else:
    print("You cannot vote")
```

10.6.2 Loops

- **for loop** (used for iterating over sequences):

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

- **while loop** (runs as long as a condition is true):

```
x = 0
while x < 5:
    print(x)
    x += 1
```

10.7 Lists and Tuples

10.7.1 Lists (Mutable)

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)
```

10.7.2 Tuples (Immutable)

```
colors = ("red", "green", "blue")
print(colors[0])
```

10.8 Functions in Python

Functions help reuse code and make programs modular.

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

greet("Ravi")
```

- `def` is used to define a function.
- Functions may take arguments and return values.

10.9 Introduction to Modules

Modules are files containing Python code that can be imported.

Example:

```
import math
print(math.sqrt(16))
```

Common built-in modules:

- `math` (for mathematical operations)
- `random` (for generating random numbers)
- `datetime` (for working with dates and time)

! 10.10 Common Errors in Python

Error Type	Example	Reason
SyntaxError	`print("Hello"`	Missing parentheses/quotes
NameError	`print(x)` when x not defined	Using undefined variables
TypeError	`"2" + 2`	Invalid operations on data types

Always read error messages carefully—they help in debugging.

10.11 Practice Programs

1. Print your name and age.
2. Take user input for two numbers and display their sum.
3. Check if a number is even or odd.
4. Print numbers from 1 to 10 using a loop.
5. Create a list of 5 fruits and display the third one.

Summary

- Python is a versatile, easy-to-learn programming language.
- It supports various data types like integers, floats, strings, lists, tuples, and booleans.
- Control structures (if-else, loops) allow decision-making and repetition.
- Functions allow modular programming.
- Modules extend Python's capabilities with reusable code.
- Syntax errors, type errors, and name errors are common but easily fixable.
