Programming Module Notes

Structured Learning Guide & Textbook

Author: D Charan Jeet

Senior Curriculum Designer

Module Focus: Python Basics (Level I)
Target Audience: Beginners & Academic Students

February 6, 2026

Contents

Chapter 1

Level I: BASICS

Establishing the Foundation of Python Syntax

1.1 Introduction to Python

A. Concept Overview

Python is a high-level, interpreted, general-purpose programming language known for its simplicity and readability. It emphasizes code readability with its notable use of significant indentation.

B. Why It Matters

Python is the language of choice for Data Science, Artificial Intelligence, Web Development, and Automation. Its "English-like" syntax makes it the perfect starting point for new programmers, removing the barrier of complex syntax found in languages like C++ or Java.

C. Detailed Explanation

- **Interpreted:** Python code is executed line-by-line by the interpreter (not compiled all at once). This makes debugging easier but execution slightly slower than compiled languages.
- **Dynamic Typing:** You do not need to declare the data type of a variable. Python figures it out at runtime.
- **Execution Modes:**
  1. **Interactive Mode:** Typing commands directly into the Python shell (REPL) for instant results.
  2. **Script Mode:** Writing code in a file (e.g., `main.py`) and executing the file.

D. Installation Setup

To start coding, you need:

- **Python Interpreter:** Download from python.org.
- **IDE (Integrated Development Environment):**
  - *VS Code:* Lightweight, industry standard.

  - *PyCharm:* Feature-rich, specific for Python.
  - *Jupyter:* Great for data science and notes.

1.2 Variables & Data Types

A. Concept Overview

Variables are named containers used to store data values. Data types define what kind of data can be stored (numbers, text, true/false logic).

B. Syntax & Rules

Syntax Rule

```
variable_name = value
```

Naming Rules (Identifiers):

- Must start with a letter (a-z, A-Z) or underscore (_).
- Cannot start with a number.
- Case-sensitive (`Age` is different from `age`).
- No keywords (e.g., `if`, `else`, `class`).

C. Code Examples: Primitives

```python
# Integer (Whole number)
age = 25

# Float (Decimal number)
price = 19.99

# String (Text)
name = "D Charan Jeet"

# Boolean (Logic)
is_student = True

# Checking types
print(type(age))     # Output: <class ’int’>
print(type(price))   # Output: <class ’float’>
```

Listing 1.1: Data Types in Python

D. Type Conversion (Casting)

Sometimes you need to treat a number as text, or text as a number.

- `int()`: Converts to integer.
- `float()`: Converts to float.
- `str()`: Converts to string.

```python
x = "100"        # This is a string
y = int(x) + 50  # Convert ’x’ to int, then add
print(y)         # Output: 150
```

Listing 1.2: Type Casting

Common Errors

TypeError: You cannot add a string directly to a number.

```
print("Age: " + 25)  # ERROR
print("Age: " + str(25)) # CORRECT
```

1.3 Operators & Input/Output

A. Concept Overview

I/O allows the program to interact with the user. **Operators** allow the program to perform calculations and logic.

B. Input & Output Functions

1. The `print()` **function** displays output to the console.

```
print("Hello", "World", sep="-")
# Output: Hello-World
```

2. The `input()` **function** takes input from the user. *Note:* `input()` *always returns a STRING.*

```
name = input("Enter name: ")
age = int(input("Enter age: ")) # Wrapping in int() to do math later
```

C. Operators Classification

1. **Arithmetic Operators:**
   - + (Add), - (Subtract), * (Multiply)
   - / (Division - returns float)
   - // (Floor Division - returns int, removes decimal)
   - % (Modulus - returns remainder)
   - ** (Exponentiation - Power)
2. **Relational (Comparison) Operators:**
   - == (Equal to), != (Not equal)
   - >, <, >=, <=
3. **Logical Operators:**
   - `and`: True if both operands are true.
   - `or`: True if at least one operand is true.
   - `not`: Reverses the state.

D. Real-World Use Case: Simple Calculator

```
# Take two numbers
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

# Perform operations
sum_val = num1 + num2
diff_val = num1 - num2
prod_val = num1 * num2
div_val = num1 / num2

print(f"Sum: {sum_val}")
print(f"Difference: {diff_val}")
```

Listing 1.3: Basic Arithmetic Calculator

1.4 Module Summary

- Python is an interpreted, high-level language ideal for beginners.
- Variables are dynamic; you don't declare types explicitly.
- `input()` captures data as strings; use type casting for math.
- Arithmetic operators like // (Floor Div) and `**` (Power) are unique and powerful.
- Proper indentation is mandatory in Python syntax.

1.5 Practical Labs Exercises

Lab 1: The "Hello World" Script

Write a script that prints "Hello, [Your Name]" to the console using a variable.

Lab 2: Area & Perimeter Calculator

Write a program that:

- Asks the user for the `length` and `width` of a rectangle.
- Calculates Area (`L * W`) and Perimeter (`2 * (L+W)`).
- Prints the results clearly.

Lab 3: Simple Interest Calculator

Create a program to calculate Simple Interest:

$$SI = \frac{P \times R \times T}{100}$$

Take Principal (P), Rate (R), and Time (T) as user inputs.

1.6 Interview-Style Questions

1. **Q:** What is the difference between / and // operators in Python?
   A: / performs standard division resulting in a float (e.g., 5/2 = 2.5), whereas // performs floor division resulting in an integer (e.g., 5//2 = 2).
2. **Q:** Why is Python called a "dynamically typed" language?
   A: Because type checking happens at runtime, and variables do not require explicit type declaration before use.
3. **Q:** What data type does the `input()` function return by default?
   A: It always returns a string (`str`).

Notes Prepared by D Charan Jeet for Academic Use