ISBN: XXX-X-XXXX-XXXX-X

First Edition

To everyone who learns by breaking things.

Preface

This book is written to help students build a **strong and clear foundation** in **Python programming**.

The content is explained in **simple and easy English** so that beginners can understand concepts without confusion. Each topic is explained step by step with clear examples.

This book focuses on:

- Understanding basic to core Python concepts.
- Learning how Python works internally.
- Writing clean and readable code.
- Preparing a strong base for advanced topics.

Special care is taken to explain:

- Important terms in an easy way.
- Differences between similar concepts.
- Real-world usage of Python.

This book is suitable for:

- Beginners who are new to programming.
- Students preparing for exams or interviews.
- Anyone who wants to learn Python from basics.

Note: Readers are encouraged to practice the examples and experiment with the code to gain confidence and deeper understanding.

Python Programming

by Amresh Maurya

Jan 2026

Contents

Chapter 1

Definition of Python

1.1 Definition of Python

Python is a high-level, interpreted programming language that is widely used for general-purpose programming. It emphasizes code readability and allows programmers to write clear and logical programs using a simple and structured syntax.

1.2 Features of Python

- **Simple and Readable Syntax** Python uses an English-like syntax, which makes programs easy to read and write.
- **High-Level Language** Python hides complex details of the system, allowing programmers to focus on problem solving.
- **Interpreted Language** Python programs are executed line by line, which makes debugging easier.
- **Object-Oriented** Python supports object-oriented concepts such as classes and objects.
- **Portable (Platform Independent)** Python programs can run on different operating systems without any modification.
- **Extensive Library Support** Python provides a large standard library that supports various programming tasks.
- **Open Source** Python is free to use and distribute.

1.3 Compiler

A compiler is a program that translates the entire source code written in a high-level programming language into machine code at once. The generated machine code is saved as an executable file, which can be run later.

Examples: C, C++

- Translates the whole program at one time
- Produces an executable file
- Errors are reported after compilation

1.4 Interpreter

An interpreter is a program that translates and executes source code line by line. It does not create a separate executable file. Each line is analyzed and executed immediately.

Examples: Python, JavaScript

- Translates and executes code line by line
- No separate executable file is generated
- Easier to debug compared to compiled languages

Chapter 2

Types of Codes and Program Execution

2.1 Types of Codes and Program Execution

2.1.1 Source Code

Source Code is the code that a programmer writes in a high-level programming language like Python, C, or Java. It is human-readable and easy to understand.

Example:

```
1 print("Hello, Python!")
```

Listing 2.1: Hello Python Example

2.1.2 Machine Code

Machine Code is the code that the computer's processor can directly understand. It is in binary (0s and 1s) and not readable by humans.

Example:

```
10101010 11001100  (just an illustration)
```

2.1.3 Binary Code

Binary Code is the language of computers consisting of 0s and 1s. Machine code is a type of binary code. All programs eventually get converted into binary code so the CPU can execute them.

2.1.4 Executable File

Executable File is a file that contains machine code ready to be run on a computer. For example, in Windows, programs ending with `.exe` are executable files.

2.1.5 Bytecode

Bytecode is an intermediate code between source code and machine code. It is generated when a program is compiled but not yet executed by the computer. Python converts source code into bytecode first.

Example: `.pyc` files in Python are Bytecode.

2.1.6 Intermediate Code

Intermediate Code is another name for bytecode. It is not specific to any computer hardware and can be executed by a virtual machine, like Python Virtual Machine or Java Virtual Machine.

Chapter 3

Virtual Machine (VM) and PVM

3.1 Virtual Machine (VM)

A **Virtual Machine (VM)** is a software program that emulates a physical computer. It provides an environment where programs can run as if they were executed on a real computer.

- Acts as a bridge between the program and the hardware.
- Makes programs platform-independent.
- Examples include: Java Virtual Machine (JVM) and Python Virtual Machine (PVM).

3.2 Python Virtual Machine (PVM)

The **Python Virtual Machine (PVM)** is a component of the Python system that executes Python bytecode. When you run a Python program:

1. Python source code (`.py`) is compiled into bytecode (`.pyc`).
2. The PVM interprets the bytecode and executes it on the computer.

Key Points:

- Makes Python platform-independent.
- Allows Python programs to run on Windows, Linux, or Mac without modification.

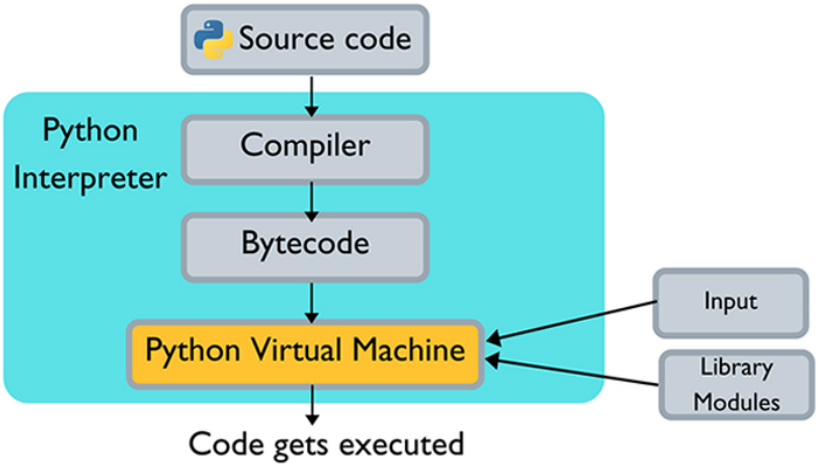

Figure 3.1: Output of Python Program

- It is a software interpreter, not hardware.

Note: The Python Virtual Machine (PVM) is a software component that interprets Python bytecode and executes it on any platform, making Python programs platform-independent.

Chapter 4

IDLE in Python

4.1 IDLE in Python

IDLE stands for **Integrated Development and Learning Environment**.

IDLE is the **default tool** that comes with Python. It is mainly used by beginners to write and run Python programs.

- It comes free with Python installation.
- It has a Python shell to run code line by line.
- It has a simple editor to write Python files.
- It shows errors clearly.

Note: IDLE is good for learning Python, not for big projects.

4.2 IDE (Integrated Development Environment)

An **IDE** is a complete software used for coding, testing, and debugging.

- It supports many programming languages.
- It has code editor, debugger, terminal, and tools.
- It is used for professional and large projects.

Examples:

- PyCharm
- VS Code
- Eclipse

4.3 Text Editor

A **Text Editor** is a simple tool to write plain text or code.

- It cannot run programs by itself.
- It does not show detailed errors.
- It is very lightweight.

Examples:

- Notepad
- Notepad++
- Sublime Text

4.4 Difference Between IDE, IDLE, and Text Editor

Feature	**IDE**	**IDLE**	**Text Editor**
Purpose	Full development	Learning Python	Writing text/code
Run Code	Yes	Yes	No
Debugger	Yes	Basic	No
Complex Projects	Yes	No	No
Ease of Use	Medium	Very Easy	Very Easy

Summary

- **IDE** = Professional tool for big projects.
- **IDLE** = Simple Python tool for learning.
- **Text Editor** = Only for writing text.

Chapter 5

Keywords in Python

5.1 Python Keywords

Keywords are reserved words in Python. They have a special meaning and **cannot be used as variable names**.

Python keywords are **case-sensitive** and always written in **lowercase**.

List of All Python Keywords

Keyword	**Simple Meaning**
False	Boolean value for false
None	Represents no value
True	Boolean value for true
and	Logical AND operator
as	Create an alias
assert	Check a condition
async	Used for asynchronous programming
await	Wait for async result
break	Exit a loop
class	Define a class
continue	Skip current loop iteration
def	Define a function
del	Delete an object
elif	Else-if condition
else	Execute if condition fails
except	Handle errors
finally	Always executed block
for	Loop through items
from	Import specific items
global	Declare global variable
if	Conditional statement
import	Import a module
in	Check membership
is	Check object identity
lambda	Create anonymous function
nonlocal	Declare nonlocal variable
not	Logical NOT operator
or	Logical OR operator
pass	Do nothing (placeholder)
raise	Raise an exception
return	Return value from function
try	Test code for errors
while	Loop while condition is true
with	Used for resource management
yield	Return value from generator

Note:

- Keywords cannot be used as variable names.

Chapter 6

Variables in Python

6.1 Variables in Python

A **variable** is a name used to store data in a program. In Python, a variable is created when you assign a value to it.

- Variables store numbers, text, or other data.
- Python does not need a data type while declaring a variable.
- The data type is decided automatically.

Rules for Naming Variables

- A variable name must start with a **letter** or **underscore** (_).
- It cannot start with a number.
- It can contain letters, numbers, and underscores.
- Spaces are not allowed.
- Python keywords cannot be used as variable names.
- Variable names are **case-sensitive**.

Correct Variable Examples

```
age = 25
_name = "Python"
total_marks = 450
price2 = 99.5
is_active = True
```

Listing 6.1: Correct Variable Names

Wrong Variable Examples

```
2age = 30          # Cannot start with number
total marks = 50 # Space not allowed
class = "A"        # 'class' is a Python keyword
price$ = 100       # Special characters not allowed
```

Listing 6.2: Wrong Variable Names

Case Sensitivity Example

```
value = 10
Value = 20

print(value)  # Output: 10
print(Value)  # Output: 20
```

Listing 6.3: Case Sensitivity in Variables

Appendix

6.2 Python Installation

This section provides instructions to install Python on **Windows, Linux, and macOS**.

Windows

1. Visit the official Python website: `https://www.python.org/downloads/`.
2. Download the latest stable release executable installer (e.g., `python-3.x.x-amd64.exe`).
3. Run the installer and make sure to **check the box "Add Python to PATH"**. 4. Verify installation in Command Prompt:

```
python --version
```

Linux (Ubuntu/Debian)

Open a terminal and run:

```
sudo apt update
sudo apt install python3 python3-pip
python3 --version
```

macOS

1. macOS may have Python pre-installed, but it is recommended to install the latest version via [Homebrew](https://brew.sh/):

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent
   .com/Homebrew/install/HEAD/install.sh)"
brew install python
python3 --version
```

Bibliography

Bibliography

[1] Python Documentation. `https://docs.python.org/3/`

[2] Leslie Lamport, *LaTeX: A Document Preparation System.*

Bibliography