

CHAPTER 1

INTRODUCTION TO PROGRAMMING

LEARNING OBJECTIVES

After completing this chapter, the reader will be familiar with the following concepts.

- Become familiar with the terminologies used in programming languages and coding.
- Acquire a fundamental understanding of programming and software.

LEARNING OBJECTIVES

- Know how interpreters and compilers facilitate collaboration between a computer's hardware and software in the system's architecture.
- Overview of the Python Integrated and Learning Environment.
- Ability to use Python as a scripting language.
- Provide awareness about Python programming resources.
- Overview of Python documentation and Support.

1.1 Basics of Programming Language

- **Problem-solving** skill involves identifying issues, formulating problems, devising solutions, and communicating them
- **Computer program** is a comprehensive, step-by-step set of instructions that tells a computer exactly what it should be doing.

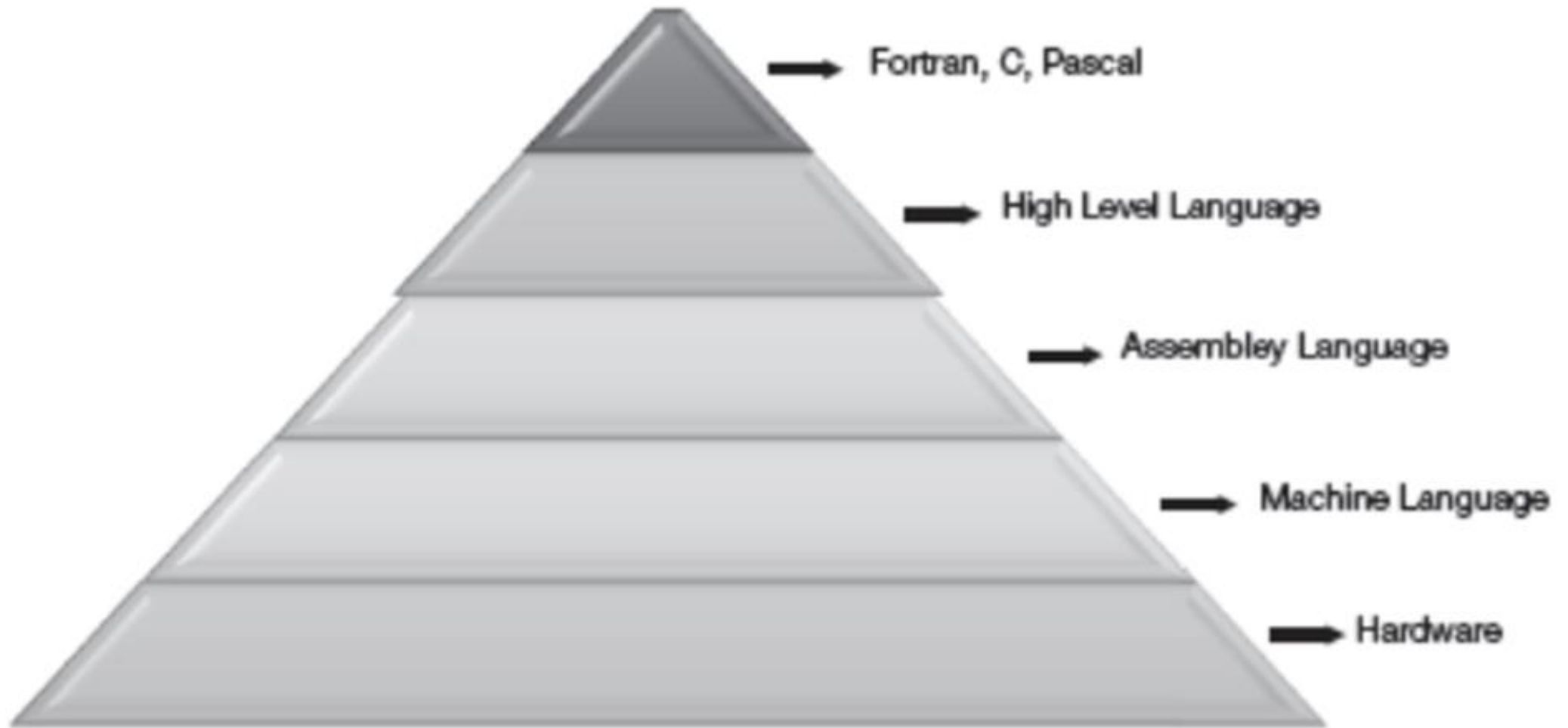


Fig. 1.1: Pyramid Showing the Level of the Translation of Languages

- *Coding Mechanism*-Coding guarantees that a computer's software and hardware can connect by compiling the code into assembly language.
- High-level and assembly-level languages are then converted into binary coded signals (1's and 0's) to allow computational hardware and software to communicate.

- *Learning to code* is crucial because it enhances problem-solving and logic abilities, leverages technology to power business operations, and facilitates fine-tuning.
- Learning to code leads to better job prospects.
- Programmers make apps, websites, and other digital products that change how people live worldwide.

1.2 Programming Versus Software

- “**Programming**” involves giving the computer the instructions and data it needs on time.
- *Programming Mechanism*-The process of programming is intricate. Programming is done in phases, such as composing problem statements, drawing flowcharts, designing coding algorithms, writing a computer program, analyzing and evaluating software, technical writing, keeping software up to date, and so on.

- **Computer software** is a set of program instructions.
- An **algorithm** is a set of instructions explaining how a particular computation should be performed in detail.
- A **programming language** is a language for writing down the instructions that a computer will follow. It has a structure of *syntax* (*precise form*) and *semantics* (*actual meaning*).

- There are two types of languages, namely **low-level languages** and **high-level languages**.
- Computer hardware's can only run programs written in **low-level languages/machine languages/ assembly languages**.
- A *computer program* is a series of instructions written in a high-level language to solve a problem.

1.3 Python Programming Language

Characteristics of Python Programming Language

- an open-source, high-level, interpreted, general-purpose, dynamic programming language
- relatively easy to pick up and use.
- Extensive libraries are available

- *flexibility as a programming language* means it can also be used as an add-on to make highly customizable programs.
- *Cross-platform language*
- *Software quality.*
- Python data types are strongly and dynamically typed.

- *embedded easily into any application to provide a programmable interface.*
- *Developer productivity.*
- Python's *rapid edit-test-debug cycle* and lack of a compilation step make development and debugging much more efficient.

Table 1.1: Object Oriented Programming Vs. Procedural Programming Vs. Functional Programming

Object Oriented Programming (OOP)	Procedural Programming	Functional Programming
The program is divided into small parts called objects.	The program is divided into small pieces called modules/procedures	Programs are divided into functions which consist of mathematical expressions
OOP follows the bottom-up approach	Procedural Programming follows the top-down approach	Top-down or Bottom-up is tracked based on programmers' desires. Bottom-up is preferred
Data or functions can be easily added to the system.	Adding data and functions is relatively not easy.	Adding data and functions is pretty not easy.

OOP has access specifiers as public, private, and protected	No access specifiers in procedural programming	No access specifiers in functional programming
OOP is more secure because it provides data hiding	Procedural Programming is less secure because it does not have any proper way of hiding the data	Functional Programming does not offer data hiding, so it is not secure
Examples: C++, Java, Python, C#	Examples: C, FORTRAN, Pascal, Basic	Examples: Lisp, Erlang, Haskell, Clojure

Limitations of Python

- *Speed is an issue.* As Python is an interpreted language, it is slower than compiled languages.
- *Design restrictions.* The dynamically typed language comes with errors only at runtime; Python's global interpreter lock and whitespace are a few design issues.

1.3.2 Implementations Using Python

- utilized for database access, Internet scripting, distributed programming, and extension-language works.
- used as Graphical User Interfaces (GUI), shell tools -system admin tools, and command line programs.
- language-based modules -instead of special-purpose parsers.

1.3.2 Implementations Using Python

- helpful for rapid prototyping and development.
- used in Web Application development,
 - building desktop GUIs,
 - Email Parsing,
 - Network programs,
 - Desktop applications,
 - Internet Protocol,
 - System administration, and
 - Video Games Development.

1.4 Compiler Versus Interpreter

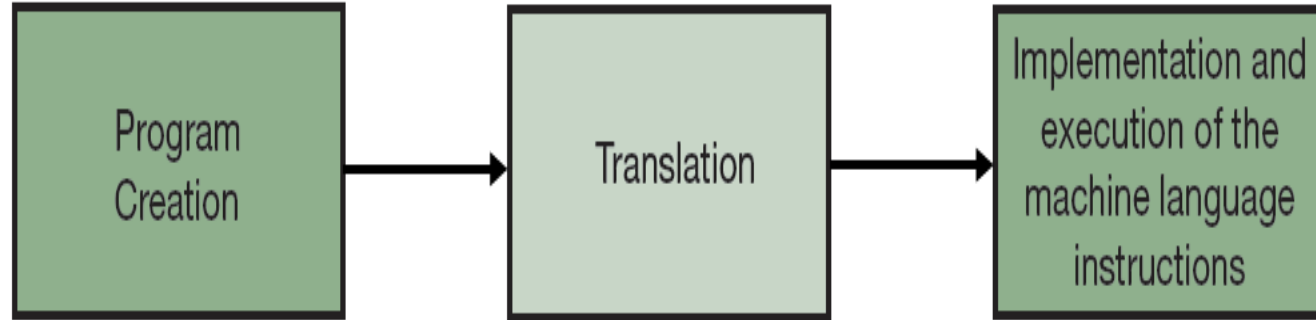


Fig. 1.2: The Process of Creating a Computer Program

- **Compiler** (Fig 1.3) meticulously scans and converts the source code before launching the program. In this perspective, the high-level program is called the *source code*, and the translated program is called the *object code* or the *executable*. Compiling a program allows users to run it repeatedly without retranslating it each time. A hardware executor can then run the compiled object code.

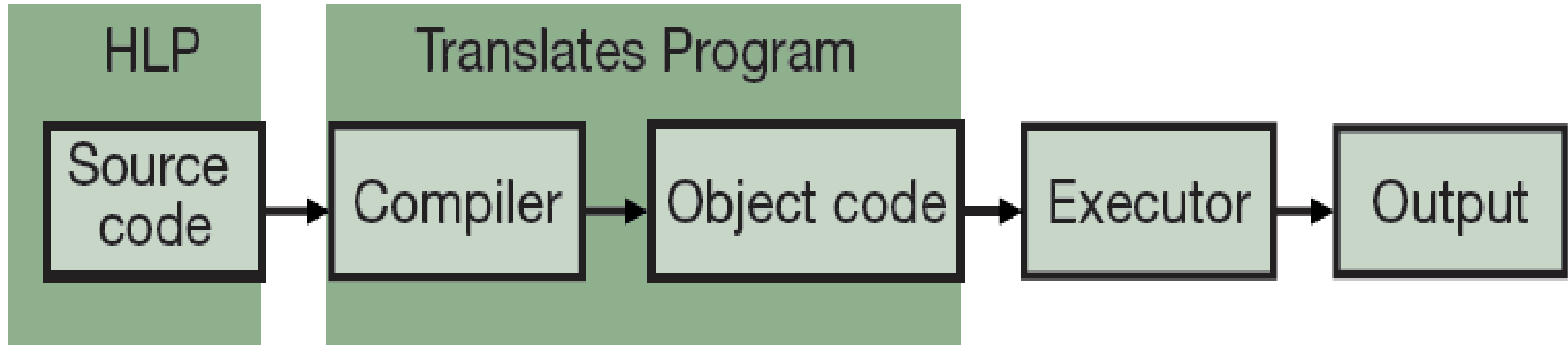


Fig. 1.3: Structure of a Compiler

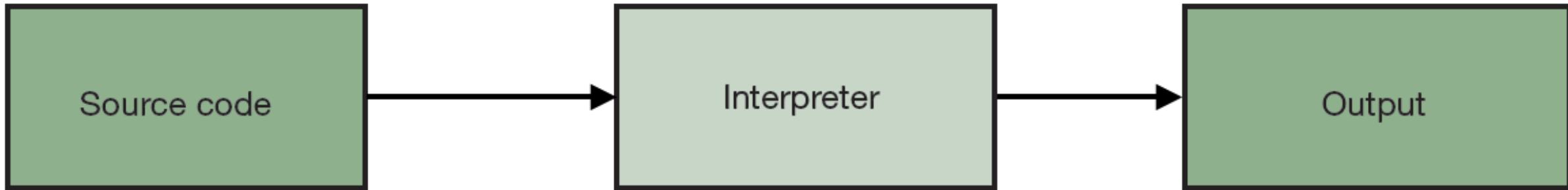


Fig. 1.4: High-Level Block diagram of an Interpreter

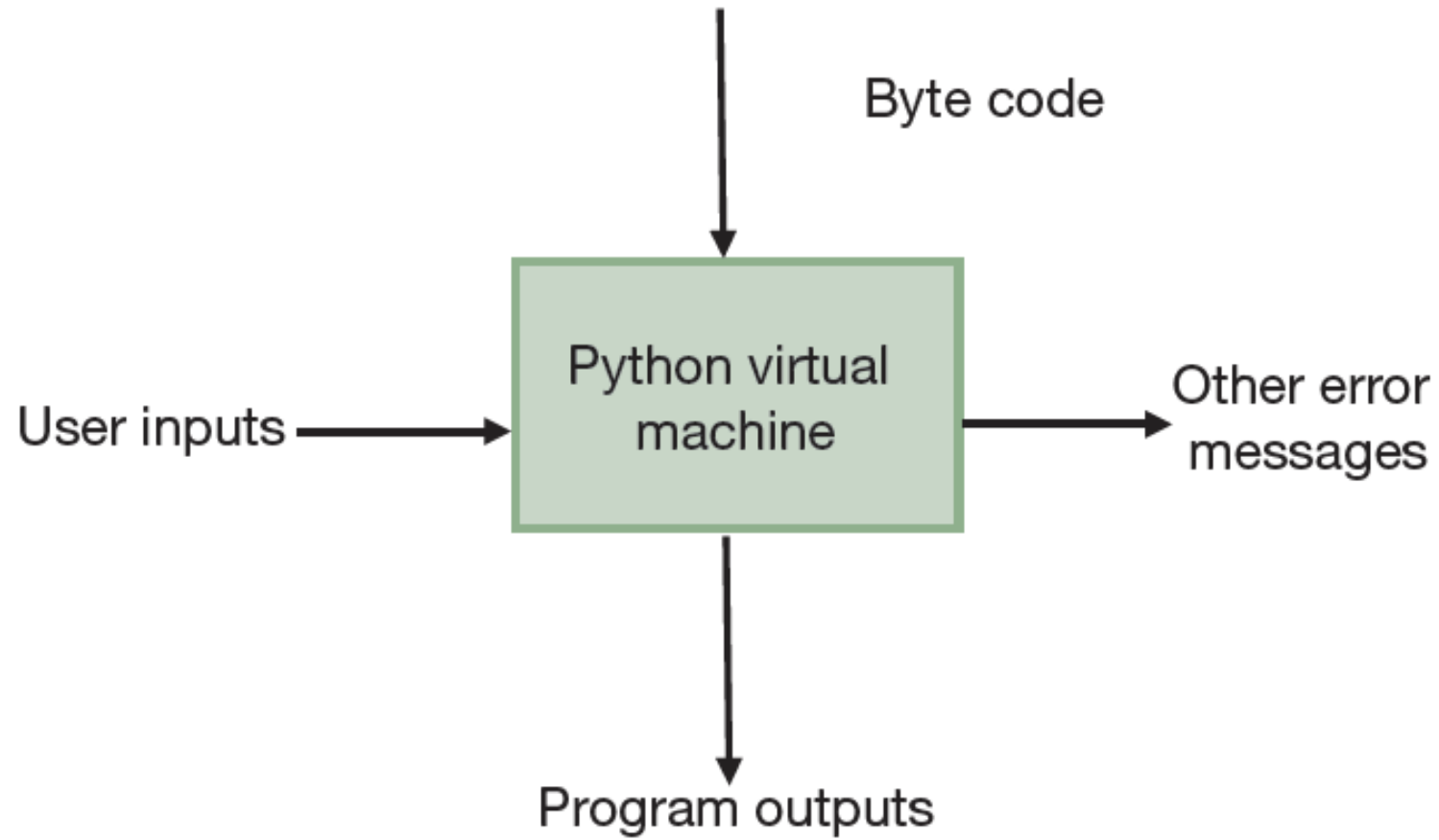


Fig. 1.5: Python Interpreter

- **Table 1.2: Comparison of Compiler and Interpreter**

Basis for Comparison	Compiler	Interpreter
Input	Scans and takes an entire program once at a time and translates it into a machine code.	It takes a single line of code or instruction once at a time and translates it.
Saving	It saves code as it compiles it.	It does not keep the interpreted code.
Output	It creates intermediate object code.	It does not create any intermediate object code.
Working Mechanism	The compilation is done before execution.	Compilation and execution take place concurrently.

Basis for Comparison	Compiler	Interpreter
Pertaining Programming Languages	C, C++, C#, Scala, and typescript use a compiler.	PHP, Perl, Python, and Ruby use an interpreter.
Speed	Comparatively faster. It takes more time to examine the source code, but the execution time is reasonably quick.	It is slower. It takes less time to examine the source code, but the execution time is slower.
Memory	Requires more memory due to the creation of object code.	It requires less memory as it does not generate intermediate object code.
Errors	Exhibits all errors after compilation, all at the same time.	Exhibits errors of each line one by one.
Error Detection	Difficult	Easier comparatively

1.5 Integrated Development Environment (IDE)

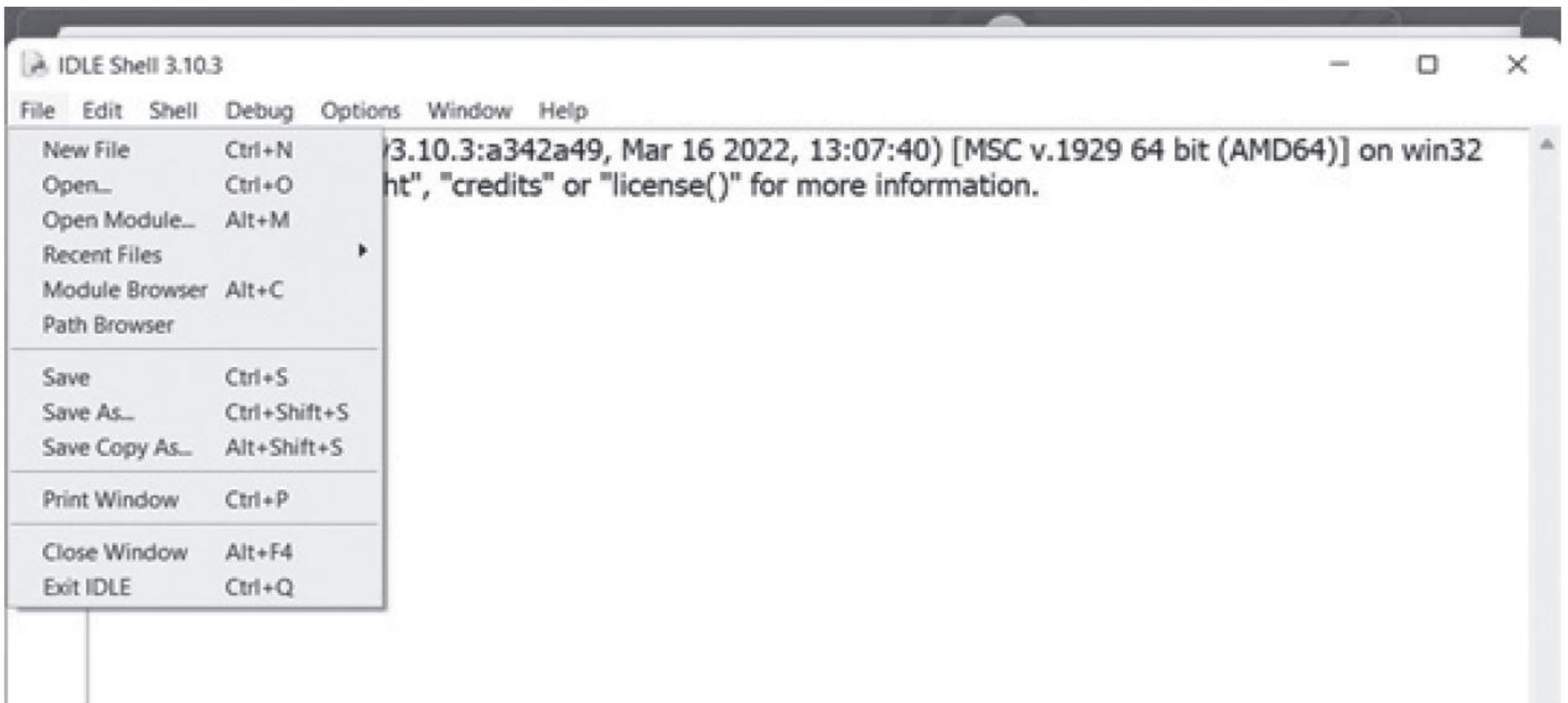
- IDLE stands for Interactive DeveLopment Environment. IDLE, as an *interpreter*, translates the code into a language the computer can understand.
- Python's default implementation, IDLE (“Integrated Development and Learning Environment”), includes the complete set of IDE packages and fundamentals.

1.5.1 Python in interactive mode (windows OS)

- *Shell or Interactive shell or Interactive Interpreter-* The Python IDLE instantly launches a Python shell, saving time.
- It is a Read-Eval-Print Loop (REPL). It reads every statement, evaluates, and displays a Python statement.
- Code snippets can be run in Python's Interpreter shell

1.5.2 Python in script mode

- A Python script or program is read line by line and executed when run in script mode.
- In script mode, code can be written in a standalone file (ending in.py) and run independently.
- These lines of code are easily modifiable and reusable.



**Fig. 1.11: Python shell -File Drop
Down Menu**

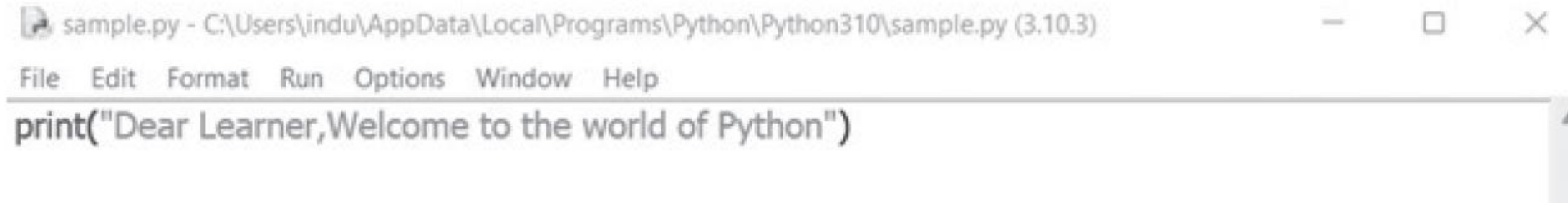


Fig. 1.12: Screenshot of the Python editor

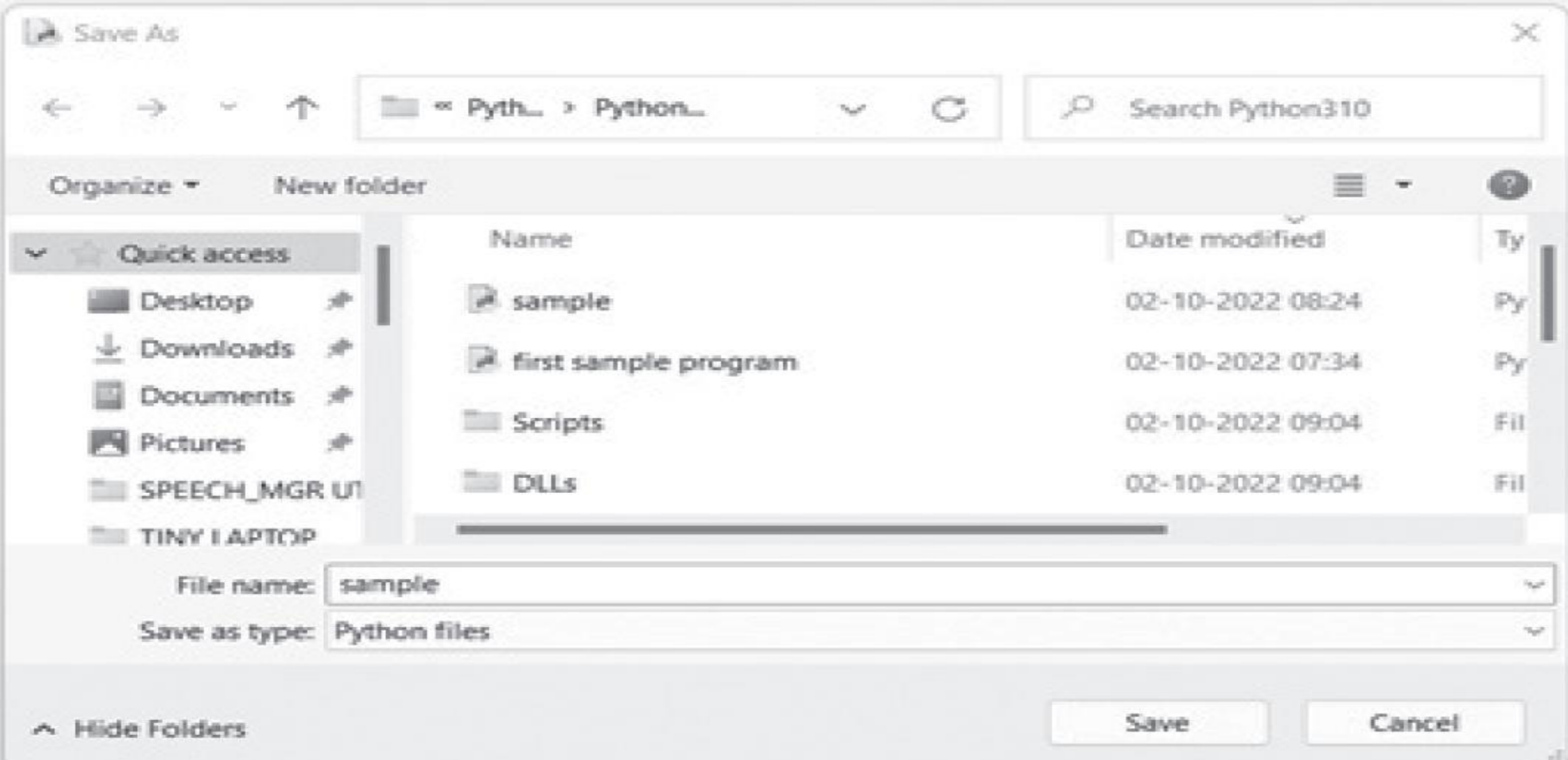


Fig. 1.13: Screenshot of saving the file sample.py

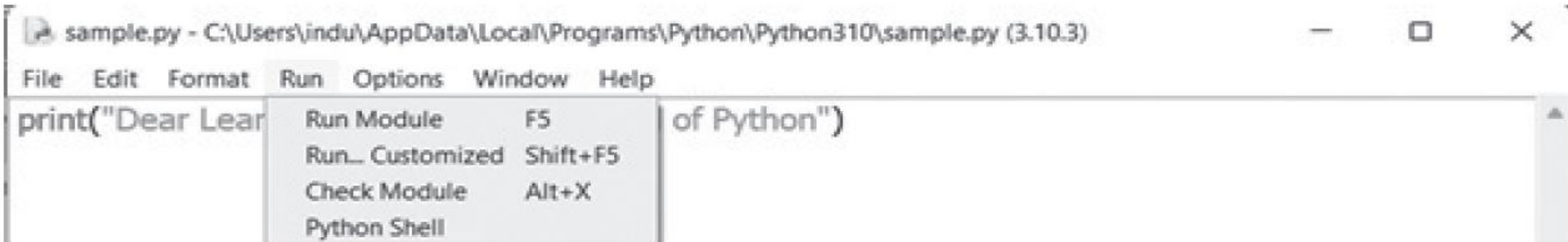
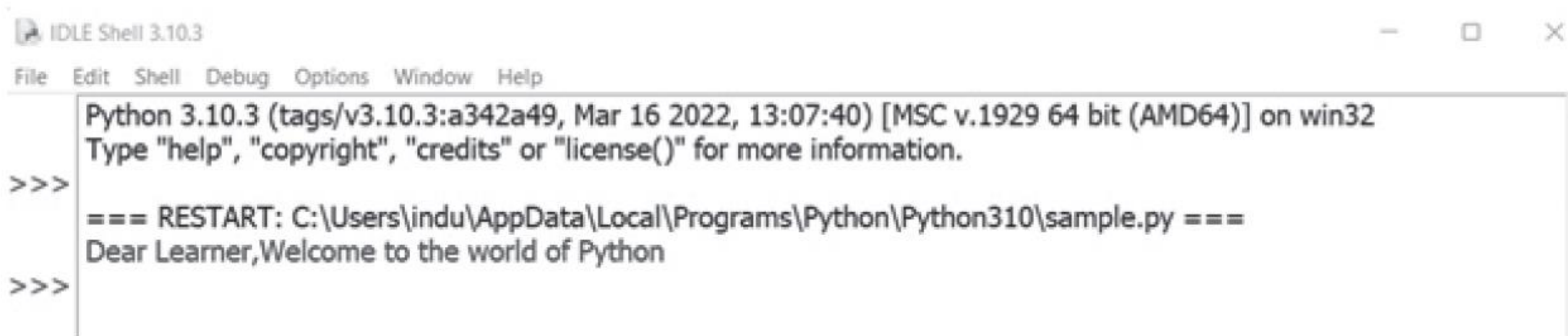


Fig. 1.14: Screenshot showing the run module or F5 button



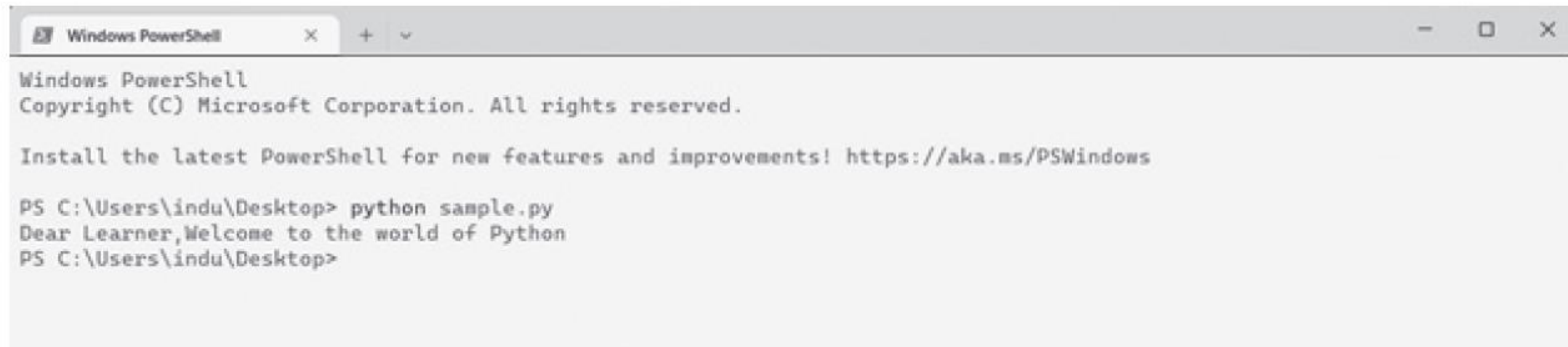
The screenshot shows the IDLE Shell 3.10.3 window. The title bar reads "IDLE Shell 3.10.3". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following output:

```
Python 3.10.3 (tags/v3.10.3:a342a49, Mar 16 2022, 13:07:40) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
=== RESTART: C:\Users\indu\AppData\Local\Programs\Python\Python310\sample.py ===
Dear Learner,Welcome to the world of Python

>>>
```

Fig. 1.15: Screenshot showing the output display of the program(sample.py)



The screenshot shows a Windows PowerShell window with a title bar that includes the application icon, the text 'Windows PowerShell', and standard window controls (close, maximize, and a dropdown arrow). The main content area displays the following text:

```
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows  
  
PS C:\Users\indu\Desktop> python sample.py  
Dear Learner,Welcome to the world of Python  
PS C:\Users\indu\Desktop>
```

Fig.1.16: Screenshot showing the output of the program in the Windows PowerShell

- **Common Python syntax colours:** Colour coding in IDLE aids in understanding the meaning so that one can easily visualize the mistakes made.
- **Keywords in orange colour**
- **Strings in green colour**
- **Comments in red colour**
- **Definitions in blue colour**
- **Misc. Words in black colour**

1.5.3 Command Line Interpreter

- *Command Line Interpreter (CLI), Console User Interface (CUI), command processor, shell, command line shell, and command interpreter.*
- A command line interpreter, also known as a console or shell, takes user input (typically in the form of a single command) and applies it to the system.

- In the absence of errors, it executes the command and returns the expected data; otherwise, an error message is displayed.

```
PS C:\Users\indu\Desktop> python sample.py  
Dear Learner,Welcome to the world of Python  
PS C:\Users\indu\Desktop>
```

Fig. 1.17: Python command prompt

1.6 Various Resources for Python Programming

- Sublime Text 3
- Notepad++
- Atom
- Visual Studio Code
- Vim

Python IDLE Editor

- Spyder
- Thonny
- PyCharm

Getting Help in Python

- **Python Official Website:** <https://www.Python.org/>
- **Python Documentation Website:** <https://www.Python.org/doc/>