

Chapter 1

Installing and using Python

1. Learning objectives

By the end of this chapter, students should be able to:

- Explain what Python is and what a programming language is;
- Identify some important characteristics and applications of Python;
- Install Python on a computer;
- Check whether Python has been installed correctly;
- Check the installed Python version;
- Distinguish between Python and the software used to write Python programs;
- Identify the main types of tools used to write Python code;
- Explain the basic role of the Python interpreter;
- Recognize the difference between interactive use and a saved Python script.
- Create a simple Python source file;
- Run a first Python program;

2. Introduction

Computers are widely used in engineering. They can be used to perform calculations, analyze experimental data, automate repetitive tasks, and solve complex engineering problems. However, a computer does not automatically know how to solve a problem. It needs special and precise instructions. **Programming** is the process of writing instructions that a computer can execute. A **programming language** provides a formal way to express these instructions. So, it allows a programmer to express instructions in a form that can be processed and executed by a computer. **Python** is one of these programming languages.

Throughout this course, Python will be used to develop programming skills and problem-solving abilities that can later be applied to engineering problems.

Figure 1.1 illustrate a useful way to think about programming using Python.

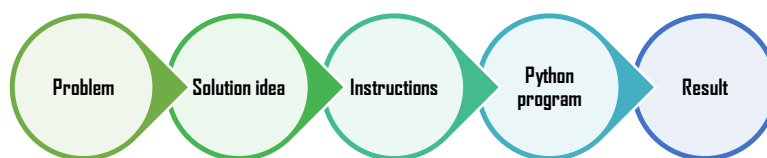


Figure 1.1 Problem-solving using Python

3. What is Python?

Python is a **high-level programming language** like C++, Perl, PHP and Java. Python is known for its relatively clear and readable syntax, less time to write and its portability. It is used in many areas including education, scientific computing, engineering, web development, artificial intelligence, machine learning, and many other applications.

For an engineering student, Python can eventually be used for tasks such as: numerical calculations, processing experimental measurements, analyzing sensor data, plotting and visualization, simulation, and data-driven engineering applications.

The objective of this course is not simply to memorize Python commands. The main objective is to learn how to **solve problems using programming**.

4. Why Python?

There are many programming languages, but Python is particularly suitable for beginners. One reason is that Python programs can often express an idea with relatively little code.

For example:

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

This instruction displays:

```
Hello, Python!
```

The syntax is relatively easy to read, which allows beginners to concentrate on programming concepts and problem solving. Python is also widely used outside education, including scientific and engineering applications. Therefore, learning Python provides two benefits:

1. It develops fundamental programming skills.
2. It provides a practical programming tool that can be used later in engineering work.

5. Installing Python

Before writing and executing Python programs, Python must be installed on the computer. Python is available for the major desktop operating systems: Windows, Linux and macOS.

The official Python website provides the current Python releases and installation resources (<https://www.python.org>). The safest approach is to obtain Python from the official Python website and avoid downloading the Python interpreter from unknown websites. The exact installation procedure depends on the operating system.

5.1. Installing Python on Windows

The Windows version of Python is available from the official Python website. Current releases provide Windows installers and the Python Install Manager.

A typical installation procedure is:

Step 1: Download Python

Open the official Python website and go to the Windows download section and choose an appropriate Python release for your computer. For most current Windows computers, the 64-bit version is appropriate.

Step 2: Start the installation

Run the downloaded installer and follow the installation instructions. When the installer provides an option related to making Python available from the command line or adding Python to the system path, beginners should pay attention to this option because it makes it easier to run Python from a terminal. *Figure 1.2* illustrates the installation progress of the Python 3 on Windows 10.

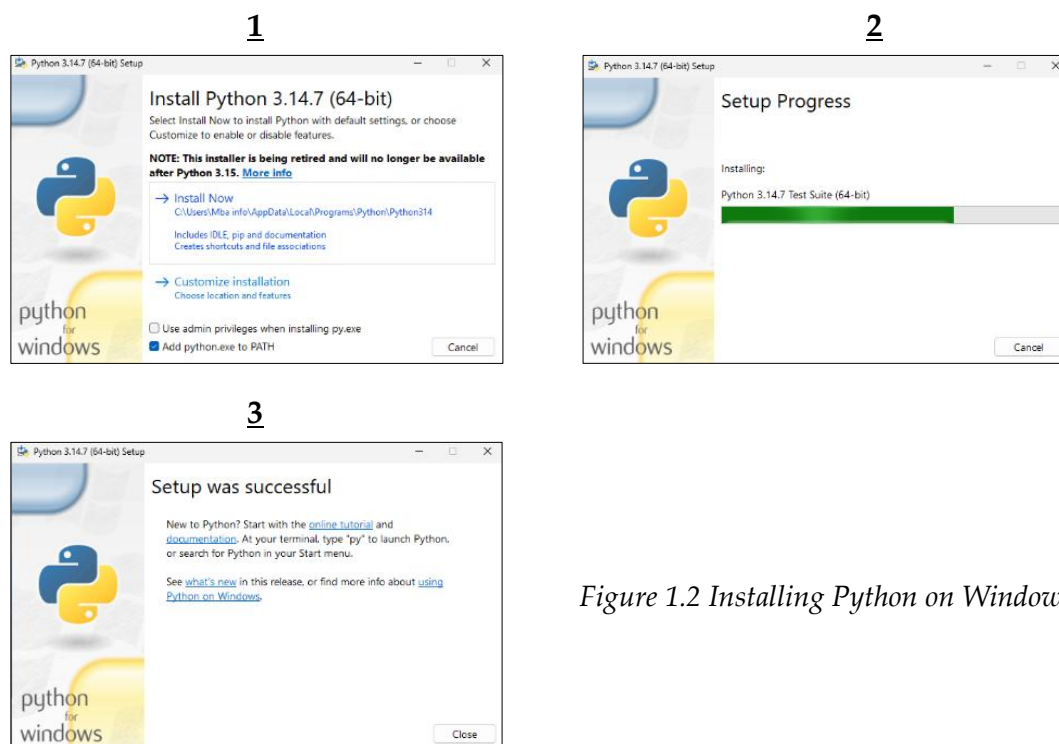


Figure 1.2 Installing Python on Windows 10

Step 3: Verify the installation

Open **Command Prompt** or **PowerShell**.

Enter:

```
python --version
```

If Python is available through the python command, the terminal should display something similar to:

```
Python 3.14.7
```

On some systems, the Python launcher can also be used:

```
py --version
```

The important point is to confirm that a Python 3 installation is available.

5.2. Installing Python on Linux

Many Linux distributions provide Python through their package management systems. Python may already be installed on a Linux system.

Open a terminal and enter:

```
python3 --version
```

If Python is installed, a result similar to the following may appear:

```
Python 3.14.7
```

The exact version depends on the Linux distribution and its installed packages. If Python is not installed, the student should follow the installation procedure recommended by the Linux distribution. The important principle is to use the package-management system of your Linux distribution when appropriate, rather than downloading random installation files from the Internet.

5.3. Installing Python on macOS

Python can be installed on macOS using the installers provided through the official Python website. The official Python downloads page provides current macOS releases and installers.

After installation, open the **Terminal** and check the installation with:

```
python3 --version
```

A result similar to:

```
Python 3.14.7
```

indicates that Python 3 is available through the python3 command.

5.4. Checking the Python installation

Regardless of the operating system, it is important to verify the installation before starting the use of Python.

The general procedure is:

1. Open a terminal or command-line interface.

2. Enter the appropriate Python version command.
3. Check that Python 3 is reported.

Typical commands are:

Windows

```
python --version
```

or:

```
py --version
```

Linux and macOS

```
python3 --version
```

The exact command can vary depending on the operating system and configuration.

5.5. The Python version

Python has different releases, and the version number may change over time.

For example:

```
Python 3.14.7
```

contains three important parts:

```
3 . 14 . 7
```

The first and the second numbers identify the major and the minor version respectively, where the third number indicates the patch release.

For this course, the important requirement is that students use a **modern Python 3 release** (3 . 14 . 7) compatible with the course examples.

6. Ways to write Python code

Python code is simply text written according to the rules of the Python language. There are several types of software that can be used to write this code. It is important to understand that:

Python is the programming language. An editor or IDE is a tool used to write and work with Python programs.

The same Python program can often be written using different editors.

6.1.Text editors

A **text editor** (Figure 1.3) is a program designed primarily for creating and editing text files. A simple text editor can be used to write Python code and the file can then be saved with a `.py` extension. A simple text editor has the advantage of being easy to understand, but it may provide few programming-specific features such as automatic code completion, debugging tools, syntax highlighting, project management or integrated execution. Therefore, although a simple text editor can be used to write Python, beginners often prefer tools designed specifically for programming.

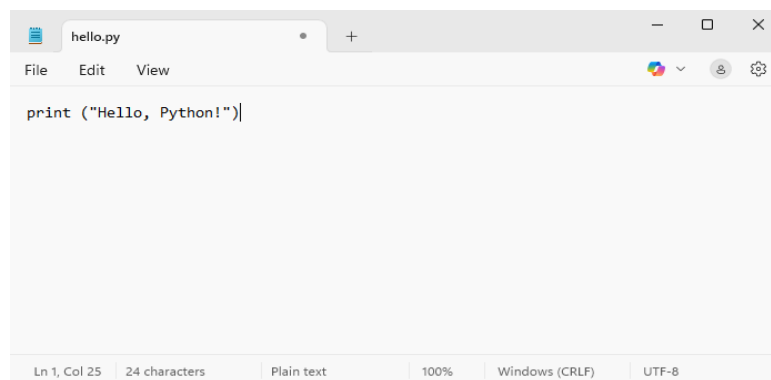


Figure 1.3 Simple text editor

6.2.Code editors

A **code editor** (Figure 1.4) is an editor designed specifically for writing computer programs. Unlike Text editors, code editor may provide features such as: syntax highlighting, indentation support, code completion, error indicators, search and replace, file management and extensions. One widely used example is **Visual Studio Code**.

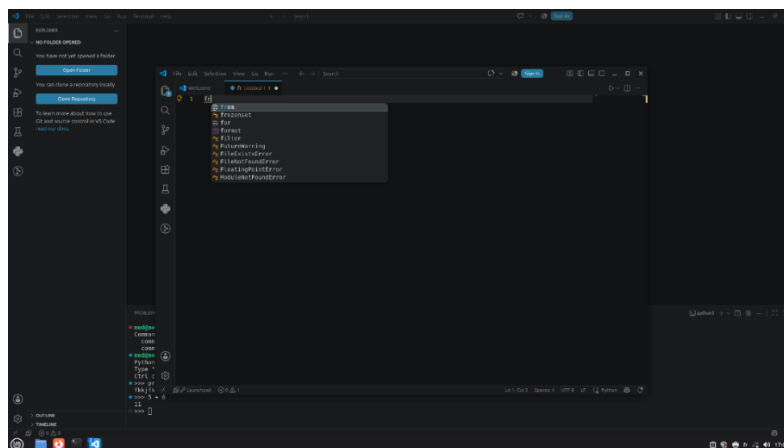


Figure 1.4 Visual Studio Code editor

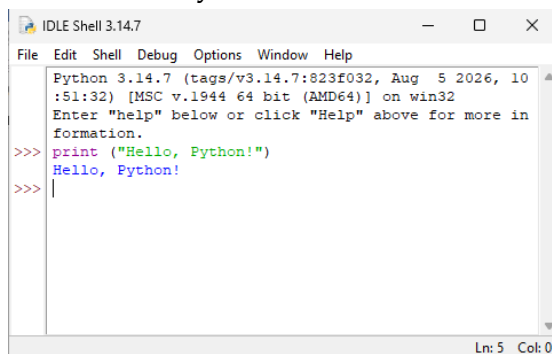
6.3.Integrated Development Environments (IDEs)

An **Integrated Development Environment** (*Figure 1.5*), usually called an **IDE**, combines several programming tools in one application. An IDE may provide: a code editor, a Python interpreter or interpreter integration, debugging tools, project management, syntax highlighting, code completion and tools for running programs.

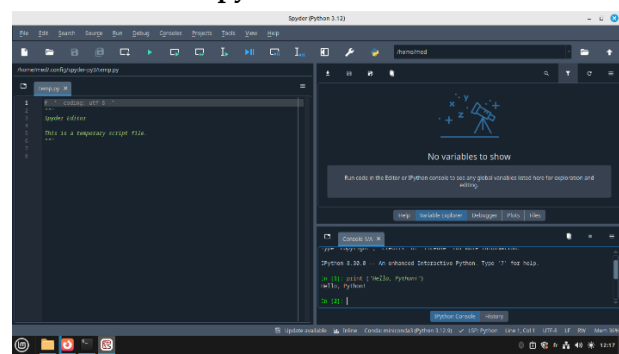
Examples of Python IDEs include **IDLE**, **Spyder** and **PyCharm**.

Python's own documentation identifies **IDLE** as Python's Integrated Development and Learning Environment and notes that it generally accompanies Python installations, although availability can vary by operating system.

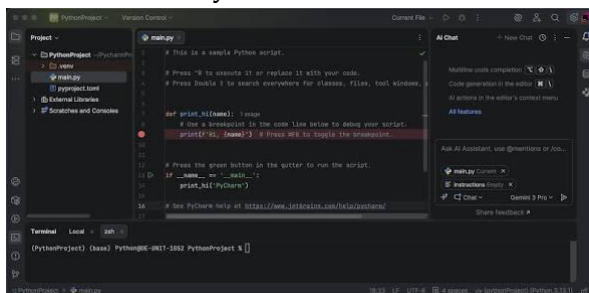
1. Python IDLE



2. Spyder



3. PyCharm



4.

Figure 1.5 Integrated Development Environments (IDEs)

6.4.Notebook environments

Another way of working with Python is through a **notebook environment** (*Figure 1.5*). A notebook divides code into separate sections called **cells**. A student can write some code, execute it, observe the result, modify the code, and execute it again. **Jupyter Notebook** is a well-known example.

Notebook environments are particularly useful for experimentation, numerical calculations, data analysis, visualization, and scientific work. They will become more relevant when students work with data and engineering applications.

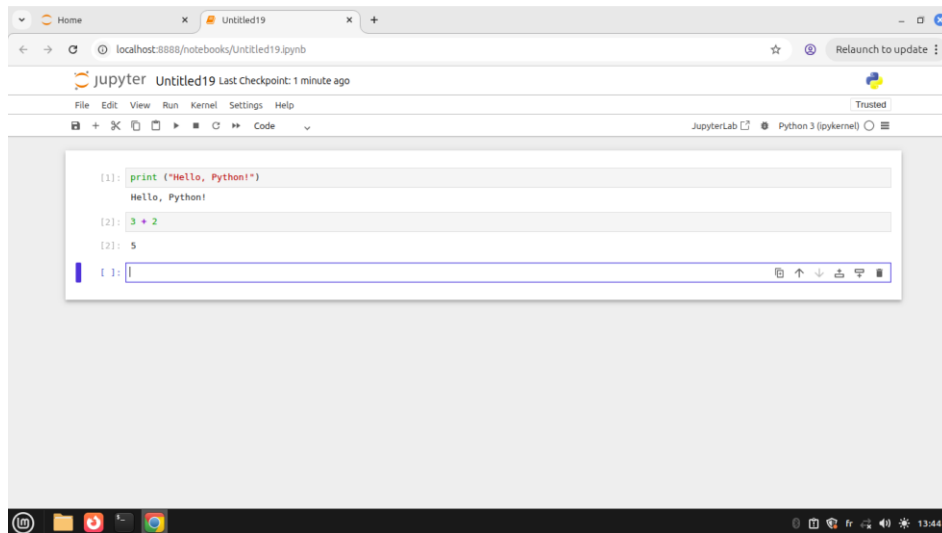


Figure 1.5 Jupyter Notebook

The following table compares the main types of tools that can be used to write python code.

Table 1.1 Python editor types

Tool type	Main purpose	Example	Beginner suitability
Text editor	Edit text files	Notepad	Simple, but limited
Code editor	Write and manage code	Visual Studio Code	Very good
IDE	Complete programming environment	Spyder, IDLE, PyCharm	Very good
Notebook	Interactive scientific/programming work	Jupyter Notebook	Very good for experimentation

A beginner can work successfully with **IDLE**, **Spyder**, **Visual Studio Code** or another suitable Python environment. The choice may depend on the user's operating system and future needs.

For example:

IDLE is simple and closely associated with Python itself.

Spyder is particularly convenient for scientific and engineering programming.

Visual Studio Code is a flexible and widely used code editor.

Jupyter Notebook is useful for interactive calculations and data analysis.

7. The Python interpreter

The high-level languages can be processed into low-level languages by two types of programs, **compilers** and **interpreters**. A compiler translates the **program (source code)** into an **object code (executable)**. Once the program is compiled the user can execute the obtained code. Unlike the compiler, the interpreter reads a high-level program and executes it immediately. Since Python programs are executed by an interpreter, Python is considered as an interpreted language. So, the **Python interpreter** is the software responsible for executing Python instructions. A simplified view is indicated in *Figure 1.6*:

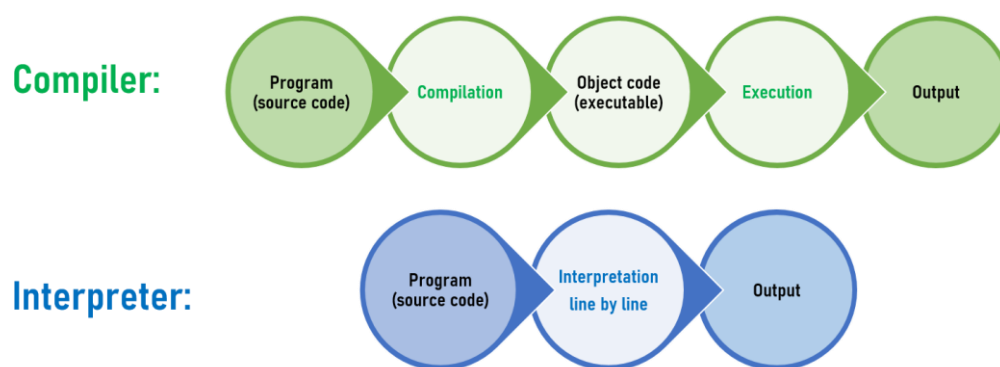


Figure 1.6 Difference between the compiler and the interpreter

There are two ways to use the Python interpreter: **interactive mode** and **script mode**.

Interactive use

Instructions can be entered directly and executed immediately:

```
>>> 5 + 3
8
```

This is useful for quick experiments and simple tests.

Script use

Python instructions can be saved in a file such as `calculation.py` and then executed as a program. The two approaches are complementary. For now, we remember only this distinction:

Interactive mode is useful for trying instructions immediately, while a script stores Python instructions in a file that can be saved and executed later.

A detailed study of both modes will be provided in **Chapter 2: Python basics**.

8. First example

Example 1 - First interactive command

In interactive mode, the user types Python programs and the interpreter displays the result:

```
>>> print("Introduction to Python")  
Introduction to Python
```

Example 2 - Creating the First Script

Task: Create and run a Python program that displays three messages.

Step 1: Create a file

Create a file and name it `first_program.py`

Step 2: Write the program

Write in the created file the following commands

```
print("Introduction to Python")  
print("Engineering Student")  
print("My First Program")
```

Step 3: Save the file

Save the file.

Step 4: Run the program

Run it using your Python environment.

Output

```
Introduction to Python  
Engineering Student  
My First Program
```

9. Common beginner mistakes

In what follows some common beginner mistakes:

1. Confusing Python with the editor

Python is the programming language where an editor or IDE is a tool used to write Python programs. Changing the editor does not change the Python language.

2. Installing Python from an unknown website

Students should obtain Python from the official Python website or from the trusted package-management system of their operating system.

3. Not checking the installation

After installation, always verify that Python is available using:

```
python --version
```

or:

```
python3 --version
```

depending on the operating system.

4. Using the wrong command

The command used to start Python is not identical on every operating system.

For example:

```
python
```

may work on one system, while:

```
python3
```

is required on another.

5. Forgetting the .py extension

A Python source file should normally have a .py extension (For example: `program.py`)

6. Including the >>> prompt in a script

When Python displays:

```
>>>
```

this is the interactive prompt and it is not normally part of the Python program. So, in the script we write

```
print("Hello")
```

and not:

```
>>> print("Hello")
```

7. Thinking that errors mean Python is broken

Beginners will inevitably encounter errors. An error usually means that Python has detected something that needs to be corrected. Learning to read and understand error messages is an important programming skill.

10. Exercises

Exercise 1

Open a terminal or Python environment and determine:

1. Whether Python is installed.
2. Which Python version is installed.
3. Which command is used to start Python on your computer?

Exercise 2

Open the Python interpreter and calculate:

`20 + 12`, `50 × 4` and `225 / 12`

Exercise 3

Create a file named: `student.py` and write a program that displays:

My name is ...

I am an engineering student.

I am learning Python.

Replace ... with your name and run the program.

Exercise 4

Match each situation with the most appropriate type of tool:

1. Writing a very simple text file.
2. Developing a Python program with syntax highlighting and code completion.
3. Working interactively with scientific calculations and data.
4. Using a complete programming environment with debugging tools.

Possible answers:

- Text editor
- Code editor

- IDE
- Notebook environment

Explain your choices.

11. Chapter summary

In this chapter, we introduced Python and prepared the environment needed to use it. We learned that:

- Python is a high-level programming language.
- Programming consists of writing instructions that a computer can execute.
- Python is used in many scientific, engineering, and practical applications.
- Python must be installed before it can normally be used to run programs.
- The installation procedure depends on the operating system.
- The Python version can be checked from a terminal.
- Python should be obtained from a trusted source, preferably the official Python website or the operating system's package-management system.
- Python code can be written using different types of tools.
- A text editor is different from a code editor.
- An IDE provides an integrated programming environment.
- A notebook environment is useful for interactive and scientific work.
- The Python interpreter executes Python instructions.
- Python can be used interactively.
- A Python source file should normally have a `.py` extension
- A simple Python program can be written, saved, executed, and tested.

12. Review questions

1. What is Python?
2. What is a programming language?
3. Why is Python suitable for beginners?
4. Give four examples of applications of Python.
5. Why must Python be installed before running Python programs?

6. How can you check whether Python is installed?
7. What is the purpose of the Python interpreter?
8. What is a text editor?
9. What is a code editor?
10. What is an IDE?
11. What is a notebook environment?
12. What does the following command generally do? `python -version`
13. What is the meaning of the `>>>` prompt?
14. What is the basic difference between interactive use and a saved Python script?
15. Why should the `>>>` prompt not normally be included in a Python script?
16. Why is it useful for an engineering student to learn programming?

13. Key Terms

Term	Meaning
Programming	Writing instructions that a computer can execute
Programming language	A formal language used to write computer programs
Python	A high-level programming language
Python interpreter	Software that executes Python instructions
Source code	Human-readable code written by a programmer
Python script	Python code stored in a file
Text editor	Software mainly used to edit text
Code editor	Software designed to help write and edit source code
IDE	Integrated Development Environment
Notebook	An interactive environment in which code can be written and executed in cells
Terminal	A command-line interface used to interact with the operating system and programs
Prompt	A symbol indicating that an interactive interpreter is ready for input (<code>>>></code>)

.py

The usual file extension for Python source files

14. References

1. **Python Software Foundation.** *Python 3 Documentation — The Python Tutorial and Python User Guide.*
2. **Severance, C. R.** *Python for Everybody: Exploring Data Using Python 3.*
1. **Downey, A. B.** *Think Python. How to Think Like a Computer Scientist. 2nd Edition, Version 2.4.0*
2. **Sweigart, A.** *Automate the Boring Stuff with Python.*