Introduction to Programming in Python

John Karuitha, PhD

February 27, 2025

1 Week 1: Introduction to Programming and Python

1.1 What is Computer Programming?

Computer programming is the process of designing and building an executable computer program to perform a specific task or solve a problem. It involves writing instructions that the computer can understand, using a programming language like Python. In essence, programmers communicate with computers to make them do something—whether it's displaying a message, solving complex calculations, or controlling hardware.

1.2 The World of Programming Languages

Programming languages are like human languages—they allow you to communicate with the computer. However, just as there are many human languages (English, French, Mandarin), there are many programming languages (Python, Java, C++, JavaScript). Each language is designed for specific tasks or offers unique features, though they share many common elements.

1.2.1 Common Types of Programming Languages:

- **Low-Level Languages**: These include assembly and machine code, which are closest to the hardware and harder to write.
- **High-Level Languages**: These are easier to read and write and are closer to human languages. Python, Java, and C++ are examples of high-level languages.

Python is considered a high-level, general-purpose language, designed to be easy to read and write while still powerful enough for complex tasks.

1.2.2 Why Python?

Python has become one of the most popular programming languages because of its simplicity and versatility [1]. Here are a few reasons why Python is often the first choice for beginners:

1. **Easy to Learn**: Python's syntax is simple and readable, which allows beginners to focus more on learning programming concepts than struggling with complex syntax.
2. **Large Community Support**: Python has a vast user community. Whether you're stuck on a problem or need advice, you'll likely find a solution in forums or tutorials.
3. **Versatility**: Python can be used for web development, data analysis, automation, machine learning, artificial intelligence, and more.
4. **Cross-Platform Compatibility**: Python works on different operating systems like Windows, macOS, and Linux, making it easy to run your code across platforms.

1.3 Installing Python

To begin programming in Python, the first step is to install the Python interpreter on your machine.

- **Python.org**: You can download the latest version of Python from the official website (python.org). The installation process is simple, and Python's installer will guide you through setting it up on your operating system.

After installing, you can check that Python is installed correctly by typing `python --version` (or `python3 --version` on some systems) in your terminal or command prompt.

---

[1] See TIOBE Index of the most popular programming languages https://www.tiobe.com/tiobe-index/

1.4 Interactive Development Environments (IDEs) and Text Editors

Before you can start coding, you'll need a tool to write your Python programs. This is where **code editors** and **Integrated Development Environments (IDEs)** come into play. Both tools allow you to write and edit code, but they serve slightly different purposes.

1.4.1 1. What is a Code Editor?

A **code editor** is a lightweight tool designed specifically for writing and editing source code. They are often simpler and focus on helping you write code with features like syntax highlighting and autocomplete.

- **Example of Text Editors**:
  - **VS Code**: A highly customizable and popular editor with support for multiple languages, including Python.
  - **Sublime Text**: A fast, flexible editor with a sleek interface.
  - **Atom**: A free, open-source editor developed by GitHub.

1.4.2 2. What is an IDE?

An **Integrated Development Environment (IDE)** is a more comprehensive tool that includes a code editor alongside many additional features designed to help with the entire software development process. IDEs often come with built-in tools like debuggers, terminal access, version control, and code suggestions, making them ideal for larger projects.

- **Example of IDEs**:
  - **PyCharm**: A popular IDE for Python, offering powerful code navigation, refactoring tools, and an integrated debugger.
  - **Jupyter Notebook**: An IDE designed for data science and machine learning projects, allowing you to write and execute Python code in a web-based notebook format.
  - **Eclipse with PyDev**: A versatile IDE for many languages, including Python.

1.4.3 Differences Between Code Editors and IDEs

1. **Complexity**:
   - **Code Editors**: Simpler and lightweight. They focus primarily on code editing and are generally faster and more responsive.
   - **IDEs**: More feature-rich and complex. IDEs provide a complete environment with tools for testing, debugging, and project management.
2. **Features**:
   - **Code Editors**: Typically include syntax highlighting, basic code completion, and support for multiple programming languages.
   - **IDEs**: Include advanced features like code debugging, version control integration, and project management tools.
3. **Resource Usage**:
   - **Code Editors**: Use fewer system resources, making them faster and better for smaller projects or quick edits.
   - **IDEs**: Heavier on system resources, which can be slower to load but offer many built-in tools to manage larger, more complex projects.
4. **Learning Curve**:
   - **Code Editors**: Easier to start with, as they have a simpler interface and fewer features to learn.
   - **IDEs**: Have a steeper learning curve due to the wide range of features available, but they provide more functionality for professional-level development.

1.4.4 When to Use Which?

- **Use a Code Editor**: If you're just starting with smaller projects or want a quick and lightweight tool for writing simple scripts, a text editor like VS Code or Sublime Text is ideal.
- **Use an IDE**: For larger projects where you need tools like debugging, code linting, or version control, IDEs like PyCharm or Jupyter Notebook offer a more robust environment.

1.5 Installing an IDE or Text Editor

Here are some common IDEs and editors you can install for Python development:

1. **Jupyter Notebook**: Popular in data science. Install via the command line using `pip install notebook` or as part of the Anaconda distribution.
2. **PyCharm**: Available in two versions: the free community version and the paid professional version. Download it from jetbrains.com/pycharm.
3. **VS Code**: Download from code.visualstudio.com. VS Code requires Python extension packs to work with Python code.
4. **Sublime Text**: Download from sublimetext.com.

1.6 Installing Popular Code Editors and IDEs

1.6.1 Jupyter Notebook:

Install via pip (Python's package installer):

```
pip install notebook
```

To launch:

```
jupyter notebook
```

This will open the notebook interface in your web browser [2].

1.6.2 PyCharm:

- Download the community version from JetBrains: https://www.jetbrains.com/pycharm/download/
- Install and configure it with default settings.
- Once installed, you can create a new Python project and start coding.

VS Code:

- Download from: https://code.visualstudio.com/
- After installation, you can add the Python extension to enhance Python coding.

[2]See more details about Jupyter Notebooks here, https://jupyter.org/install.

1.7 Writing Your First Python Program

Once you've installed Python and an editor or IDE, it's time to write your first Python program.

1. **Open your text editor or IDE**.
2. **Create a new file**. In most editors, this can be done by selecting "File -> New" or simply clicking the new file button.
3. **Write your Python code**. Type the following simple program, which prints "Hello, World!" to the screen:

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

4. **Save your file** with a `.py` extension, for example, `hello.py`.
5. **Run your program**. Open your terminal or command prompt, navigate to the folder where your file is saved, and type the following command:

   ```
   python hello.py
   ```

If everything is set up correctly, you'll see the text "Hello, World!" printed in the terminal.

---

2 Conclusion

In this lecture, we've covered the basics of computer programming, explored the reasons why Python is an excellent first language, and discussed the tools (code editors and IDEs) that you'll need to start coding. With your Python installation ready and your editor set up, you're now equipped to dive into writing your first Python programs and begin your coding journey (McKinney 2022; Downey 2024).

References

Downey, Allen B. 2024. *Think Python*. 3rd ed. Green Tea Press.
McKinney, Wes. 2022. *Python for Data Analysis*. " O'Reilly Media, Inc.".