Python for Electrical and Electronics Engineering

Python for Electrical and Electronics Engineering

By

Abhijit Bora and Papul Changmai

Cambridge
Scholars
Publishing

Python for Electrical and Electronics Engineering

By Abhijit Bora and Papul Changmai

This book first published 2025

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN: 978-1-0364-4123-4
ISBN (Ebook): 978-1-0364-4124-1

TABLE OF CONTENTS

FOREWORD

In a world increasingly driven by technological advancements, the ability to harness the power of computation has become a cornerstone of innovation across all disciplines. For those in electrical and electronics engineering, this intersection of programming and engineering principles is not merely a convenience—it is a transformative force that reshapes how we design, analyze, and implement solutions to real-world problems. Python for Electrical & Electronics Engineering arrives at a pivotal moment, offering a meticulously crafted roadmap for engineers, students, and professionals to explore the dynamic synergy between computational thinking and the versatile Python programming language. This book goes beyond traditional programming tutorials. It invites readers to think critically and systematically, unraveling the complexity of engineering challenges through logical problem-solving and efficient coding practices. With Python as a lens, the book empowers its audience to demystify abstract concepts and translate them into actionable insights, bridging theory and practice in ways that are both accessible and profound. The beauty of this work lies in its universality and inclusiveness. Whether you're a novice taking your first steps into programming or an experienced engineer looking to expand your computational repertoire, this book caters to every level of expertise. Its blend of foundational theory, practical exercises, and real-world examples ensures that readers are not only equipped with technical knowledge but are also inspired to innovate. The author’s vision is clear: to cultivate a generation of problem solvers who are unafraid to explore, experiment, and excel in the ever-evolving landscape of technology. This book stands as a testament to that vision, a beacon guiding readers toward mastery in computational thinking and programming with Python. I am confident that Python for Electrical & Electronics Engineering will serve as a valuable companion on your journey, igniting a passion for learning and empowering you to make meaningful contributions to the digital age. It is not merely a book—it is an invitation to embrace a future brimming with possibilities.

Dr. Mrutyunjay Maharana.
Chief Engineer & Expert,
Hardware & High Voltage System Development,
BYD Auto Co Ltd, Shenzhen China

PREFACE

In the ever-evolving landscape of technology, where innovation is the heartbeat of progress, the fusion of computational thinking and programming skills becomes not just a necessity but a gateway to unlocking the limitless potential of our digital age. This book, “Python for Electrical & Electronics Engineering" is an endeavor to illuminate the pathways that traverse the realms of logic, problem-solving, and the art of coding.

This book is crafted with a dual purpose: to serve as a comprehensive guide for those embarking on their journey into the world of computational thinking and programming, and to provide a resource for those seeking to enhance their skills with Python. Whether you are a student stepping into the fascinating realm of computer science or electrical engineering, a professional navigating the demands of an increasingly digital workplace, or an enthusiast driven by the curiosity to decode the language of machines, this book aims to be your companion.

The journey begins with the fundamentals of computational thinking, unravelling the threads that weave through algorithms, abstraction, and problem decomposition. It then seamlessly transitions into the world of Python programming, offering hands-on exercises, real-world examples, and projects that bridge theory and application. Through this immersive approach, readers are not just passive learners; they are active participants in the construction of their computational toolkit.

We extend our gratitude to the vibrant community of educators, learners, and practitioners whose insights have shaped this book. Our hope is that, as you delve into the pages that follow, you will find not just knowledge but a source of inspiration and empowerment—a guide that propels you into the exciting realms of Electrical Engineering with Python.

With Best Wishes.
Dr. Abhijit Bora & Dr. Papul Changmai

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Overview of Python and its applications in engineering

In the dynamic landscape of technology and problem-solving, Computational Thinking has emerged as a fundamental skill, empowering individuals to address complex challenges through structured and algorithmic approaches. This book, "Python for Electrical and Electronics Engineering," serves as an invaluable guide for both beginners and enthusiasts, navigating the realms of logical reasoning, problem-solving, and the art of programming using the versatile Python language. Computational Thinking is more than just writing code. It's a cognitive approach that involves breaking down complex problems into smaller, more manageable parts, identifying patterns, and designing algorithms to solve them. It lays the foundation for effective programming by fostering a mindset that emphasizes clarity, efficiency, and creativity in crafting solutions. Through this book, readers will not only learn the syntax of Python but also cultivate the thinking patterns essential for tackling real-world challenges. Programming with Python serves as the vehicle for putting Computational Thinking into practice. Python, with its clean and readable syntax, has become a favorite among programmers, educators, and professionals alike. It's an excellent language for beginners, providing a gentle learning curve while remaining powerful enough for advanced applications. The book introduces Python as a tool for expressing computational thinking concepts, from variables and control structures to data structures and algorithms.

The key feature of the book basically includes the foundational concepts of Python. The book begins with an exploration of fundamental concepts in Computational Thinking, providing a solid theoretical base. It covers algorithmic design, problem decomposition and abstraction, preparing readers for the challenges ahead. The journey then transitions into hands-on Python programming. Readers will learn the basics of Python syntax, data types, control structures, and functions. The emphasis is not just on writing code but on writing clean, efficient, and well-

structured code. The book also delves into essential data structures such as lists, dictionaries, and sets, while also introducing algorithmic thinking. Readers will explore sorting and searching methods, and more, understanding the principles that underlie efficient computation. The overall content of the book emphasizes the real world applications. Practical examples and projects throughout the book bridge the gap between theory and application. From simple scripts to more complex programs, readers will gain hands-on experience in applying Computational Thinking to real-world scenarios. Different problem-solving strategies are also discussed. The book equips readers with problem-solving strategies, teaching them how to approach unfamiliar problems, debug code effectively, and optimize solutions. These problem-solving skills are invaluable in various fields, from software development to data science. It also adopts an interactive learning approach, with exercises, review question, and projects to reinforce concepts. Readers are encouraged to actively engage with the material, promoting a deeper understanding of Computational Thinking and Python programming. Recognizing the importance of a supportive community, the book introduces readers to resources, forums, and communities where they can seek help, share knowledge, and continue their learning journey beyond the book. Imagine approaching the world like a programmer. You see problems as puzzles to be cracked, tasks as algorithms to be designed, and information as data to be analyzed.

The computational thinking empowers you to think logically and structurally. It can develop problem-solving skills that break down complex tasks into smaller, manageable steps. The computational thinking also supports abstraction and generalization. It helps in identifying patterns and similarities to develop solutions applicable to a wider range of problems. With acquisition of knowledge one can design algorithm and automation. Data analysis and interpretation is crucial part in computational thinking. It includes collection, organization, and understanding information to make informed decisions. It is advisable to combine these skills to design new solutions and explore possibilities.

The Magic of Python Programming

Python is not just a programming language; it's a powerful tool that unlocks a world of possibilities. It is easy to learn. Python's simple syntax and clear, readable code make it accessible to beginners, allowing you to focus on concepts rather than syntax hurdles. It is versatile and powerful. From web development and data analysis to game creation and scientific

computing, Python's diverse libraries and frameworks cater to your interests. The coding in Python can be an exciting experience as you see your ideas come to life, from solving puzzles to creating interactive programs. It is widely used and in demand. Understanding Python opens doors to countless opportunities in various fields like technology, finance, science, and research.

This course aims to be your interactive playground for learning both computational thinking and Python programming. You will be able to dive into the core concepts of computational thinking. One can explore abstraction, algorithms, data analysis, and problem-solving through engaging activities and real-world examples. One will be able to learn basic syntax, control flow, functions, data structures, and object-oriented programming, building a solid foundation for your coding journey. The provision and working sets are given that one can use to work on practical projects. Apply your newly acquired skills to fun and challenging projects, like building games, analyzing datasets, or automating tasks.

Overall, Python is a versatile and powerful programming language widely used in various domains, including different branches of engineering. Its simplicity, readability, and extensive libraries make it a popular choice for engineers and researchers in the field. Here's an overview of Python and its applications in engineering:

1. General Python Features: Python is an interpreted, high-level, and dynamically-typed language, making it easy to write and test code quickly. Its code readability is enhanced by using indentation instead of braces, making it a more natural language to read and understand. Python has a large community and an extensive standard library that includes modules for various purposes.

2. Numerical Computing and Libraries: The NumPy library provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. The SciPy which is built on top of NumPy provides additional scientific computing tools, including optimization, integration, interpolation, and signal processing functions. A symbolic mathematics library known as SymPy allows manipulation and solving of mathematical expressions symbolically. The Pandas offers data structures like DataFrames, which are beneficial for handling and analyzing structured data in various formats.

3. Plotting and Visualization: This Matplotlib library of Python is widely used for creating 2D plots and charts, essential for visualizing data and presenting results. A higher-level library known as Seaborn built on top of Matplotlib, Seaborn provides a more aesthetically pleasing interface for creating statistical graphics. Whereas, this library Plotly is useful for creating interactive, web-based visualizations that can be embedded in web applications.

4. Simulation and Modeling: A discrete-event simulation library called as SimPy often used to model and simulate complex systems and processes. The PySpice library interfaces with popular circuit simulation tools, enabling engineers to simulate electrical circuits and analyze their behavior.

5. Control Systems: The SciPy's control module offers tools for analyzing and designing control systems, including functions for system representation, stability analysis, and controller design.

6. Signal Processing: SciPy's signal module provides a wide range of functions for signal processing tasks like filtering, spectral analysis, and wavelet transforms.

7. Internet of Things (IoT) and Embedded Systems: MicroPython, a version of Python optimized for microcontrollers and embedded systems, allowing easy programming and control of IoT devices.

8. Machine Learning and Artificial Intelligence: Python's machine learning libraries, such as scikit-learn and TensorFlow, find applications in various aspects of electrical engineering, such as pattern recognition, optimization, and predictive maintenance.

Overall, Python's flexibility and abundance of libraries make it a powerful tool for renewable energy measurement to tackle a wide range of challenges, from data analysis and modeling to control systems and embedded applications. Its ease of use and versatility have contributed significantly to its popularity in the field.

Python is a general-purpose programming language that is becoming increasingly popular in different sub branches of electrical engineering. It is used for a variety of tasks, including data analysis, signal processing, control system, machine learning and embedded system. Python has a wide range of libraries for data analysis, such as NumPy, SciPy, and

Pandas. These libraries can be used to perform tasks such as data cleaning, data visualization, and statistical analysis. It has libraries for dealing with both digital and analog signals, such as SciPy and PyWavelets. These libraries can be used to perform tasks such as filtering, denoising, and feature extraction. It has libraries for simulating and analyzing control systems, such as SciPy and ControlPy. These libraries can be used to design controllers for a variety of systems, such as robotic arms, power plants, and airplanes. It has libraries for training and deploying machine learning models, such as Scikit-Learn and TensorFlow. These libraries can be used to build machine learning models for a variety of tasks, such as image classification, natural language processing, and fraud detection. It has a small footprint and is easy to learn, making it a good choice for embedded systems that have limited resources. Python can be used to write firmware for microcontrollers, as well as to create graphical user interfaces for embedded systems.

It is a powerful and versatile language that can be used for a variety of tasks in electrical engineering. It is easy to learn and use, and it has a large community of users and developers. As a result, Python is becoming increasingly popular in electrical engineering, and it is likely to become even more popular in the future. Here are some additional benefits of using Python in electrical engineering:

(i) **Open source:** Python is an open-source language, which means that it is free to use and distribute. This makes it a cost-effective option for electrical engineers.

(ii) **Portable:** Python code can be run on a variety of platforms, including Windows, macOS, and Linux. This makes it a flexible option for electrical engineers who need to work on different platforms.

(iii) **Community:** Python has a large and active community of users and developers. This means that there are many resources available to help electrical engineers learn and use Python.

If you are an electrical engineer or researcher in renewable energy, we encourage you to learn Python. It is a powerful and versatile language that can be used to solve a wide range of problems in different sub branches of electrical engineering.

Summary

This chapter provides an interactive idea for learning computational thinking and Python programming. It covers core concepts like abstraction, algorithms, data analysis, and problem-solving, as well as requirement of basic syntax, control flow, functions, data structures, and object-oriented programming. It highlights the implementation of practical projects, allowing students to explore their skills to various engineering domains. It is also discussed that Python's simplicity and extensive libraries make it a popular choice among students of different engineering branch.

Review Questions

1. What is Python?
2. Explain its features.
3. Describe applications of Python programming from the perspective of engineering domain.
4. How Python programming can help in visualizing data for data analysis?
5. Explain how Python programming can help in computational thinking.

CHAPTER 2

INSTALLING PYTHON AND DEVELOPMENT ENVIRONMENTS

Installing Python and setting up development environments can vary based on your operating system (Windows, macOS, or Linux) and your specific development needs. Here are general steps for installing Python and setting up a development environment:

Installing Python: Python installations is very easy in comparison to other software execution files. The following process needs to be followed.

1. Download Python:
   a) Visit the official Python website at python.org.
   b) Navigate to the "Downloads" section.
   c) Choose the version of Python you want to install (e.g., Python 3.9) and select the appropriate installer for your operating system.

2. Run the Installer:
   a) For Windows:
      Run the downloaded executable. Check the box that says "Add Python to PATH" during installation.
   b) For macOS and Linux:
      Follow the on-screen instructions to install Python.

3. Verify Installation:
   a) Open a command prompt or terminal.
   b) Type python --version or python -V to verify that Python is installed.
   c) Additionally, you can check pip by typing pip --version or pip -V.

One can set the developing environment for writing and executing the python program. They are as follows:

(a) Text Editors and IDEs: Choose a code editor or integrated development environment (IDE) for writing Python code. Some popular choices include Visual Studio Code, PyCharm, Atom, Sublime Text

(b) Virtual Environments: It's a good practice to use virtual environments to isolate your Python projects. This helps manage dependencies and avoid conflicts. One needs to create a virtual environment using the following commands in the terminal or command prompt:

```
# On Windows
python -m venv venv

# On macOS and Linux
python3 -m venv venv
```

Activate the virtual environment:
On Windows: venv\Scripts\activate
On macOS and Linux: source venv/bin/activate

(c) Package Management: One can use pip to install Python packages. Common commands are given as follows:

pip install package_name to install a package.

pip freeze > requirements.txt to save the project's dependencies.

pip install -r requirements.txt to install dependencies from a requirements file.

(d) Version Control: If we consider using version control systems like Git for source code management, then we can host the projects on platforms like GitHub, GitLab, or Bitbucket.

(e) Documentation and Resources: The documentation section of python program can be seen in the resource section for Python Documentation, Python Package Index (PyPI) and Online tutorials and courses for Python development.

It is necessary to observe that you have the latest version of pip. It can be ensured by running command as follows:

```
python -m pip install --upgrade pip
```

You can also configure the Python Interpreter in IDE. For that in your IDE, configure the Python interpreter to point to the one within your virtual environment. By following these steps, you should have Python installed and a basic development environment set up. Customize your environment based on your preferences and project requirements.

Introduction to Problem-solving

Problem-solving is a fundamental skill in computer science and various other disciplines. It involves the systematic approach of finding solutions to challenges or tasks. Effective problem-solving requires critical thinking, creativity, and logical reasoning. Here's an introduction to the key aspects of problem-solving:

1. Understanding the Problem

(a) Read and Analyze: Begin by thoroughly understanding the problem statement. Identify the inputs, outputs, constraints, and requirements. Break the problem down into smaller components if needed.

(b) Ask Questions: Seek clarification on any ambiguous aspects of the problem. Understand the problem thoroughly before attempting to solve it.

2. Developing an Algorithm

(a) Define Steps: Create a step-by-step plan or algorithm to solve the problem. Break down the problem into smaller, manageable tasks.

(b) Pseudocode or Flowchart: Express your algorithm using pseudocode or flowcharts. This helps in visualizing the logical flow of your solution without getting bogged down by syntax.

3. Coding

(a) Select a Language: Choose a programming language suitable for the problem. Common languages include Python, Java, C++, etc.

(b) Translate Algorithm to Code: Write the actual code based on the algorithm. Pay attention to syntax, structure, and coding conventions.

(c) Incremental Development: Code incrementally and test each part to ensure it works as expected before moving on.

4. Testing

(a) Test Cases: Create a set of test cases to validate your solution. Include both normal and edge cases to cover a range of scenarios.

(b) Verify Outputs: Run your program with the test cases and verify that the outputs match the expected results. Testing helps identify errors and ensures correctness.

5. Debugging

(a) Identify Issues: If your program doesn't produce the expected results, use debugging techniques to identify and fix errors. Common debugging tools include print statements, breakpoints, and interactive debugging tools.

(b) Iterative Process: Debugging is often an iterative process. Fix one issue at a time and retest to ensure changes don't introduce new problems.

6. Optimization (Optional)

(a) Improve Efficiency: Once you have a working solution, consider optimizing your code for better performance. This step is optional and depends on the problem's requirements and constraints.

7. Documentation

(a) Document Code: Add comments and documentation to explain your code. This makes it easier for others (and yourself) to understand the purpose and functionality of your solution.

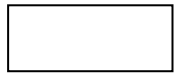
(b) Learn from Experience: Reflect on the problem-solving process. Understand what worked well and what could be improved for future problem-solving endeavours.

Problem-solving is a skill that develops with practice. It involves a combination of analytical thinking, creativity, and a solid understanding of programming concepts. However, representing algorithms using flowcharts, pseudocode, and decomposition are effective ways to communicate the logic of a solution. Let's look at each representation:

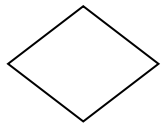
1. Flowchart:

A flowchart is a graphical representation of an algorithm that uses various symbols to represent different steps or processes. It typically uses shapes like rectangles, diamonds, and arrows to indicate the flow of control. For example, the flowchart for finding the maximum of two numbers. Here, in this flowchart, the following symbols can be used:

The rectangles represent processes or actions.



The diamonds represent decision points (e.g., checking if A > B).



The arrows indicate the flow of control.

2. Pseudocode:

Pseudocode is a high-level, human-readable description of the logic of an algorithm. It is not tied to any specific programming language syntax and focuses on expressing the steps in a natural language. For example, the pseudocode for finding the maximum of two numbers

```
Input A, B
If A > B then
    MaxNumber = A
Else
    MaxNumber = B
Output MaxNumber
```

Pseudocode allows one to express the algorithm's logic in a way that is easy to understand, regardless of the programming language.

3. Decomposition:

Decomposition involves breaking down a complex problem into smaller, more manageable subproblems. It promotes modularity and makes the solution more understandable and maintainable. For example, the decomposition for sorting an array using bubble sort is given as follows.

Algorithm: BubbleSort(Array)

1. Input: Array of elements
2. Set n as the length of the array
3. Repeat n times (outer loop)
   a. Repeat (n - 1) times (inner loop)
      i. If Array[i] > Array[i + 1], swap them
4. Output: Sorted Array

In this example, the sorting algorithm is decomposed into the main sorting logic inside the BubbleSort function. Each step is broken down into smaller parts, making the algorithm easier to understand.

Combining these representations, one can create a comprehensive view of an algorithm. For example, you might start with a flowchart to visualize the overall structure, then use pseudocode to describe the details, and finally, decompose complex parts into smaller pseudocode or flowcharts for clarity. The goal is to convey the algorithm's logic in a clear and understandable manner.

Summary

We discussed the implementation methods for installing Python followed by a discussion on environment parameter setting. The problem solving skill and its different aspects are also discussed.

Review Questions

1. Explain the environment suitable for installation of Python.
2. What is problem solving skill and how it can be solved with Python programming?
3. Explain the different symbols of flowchart.

4. What is Pseudocode?
5. Explain decomposition with example.