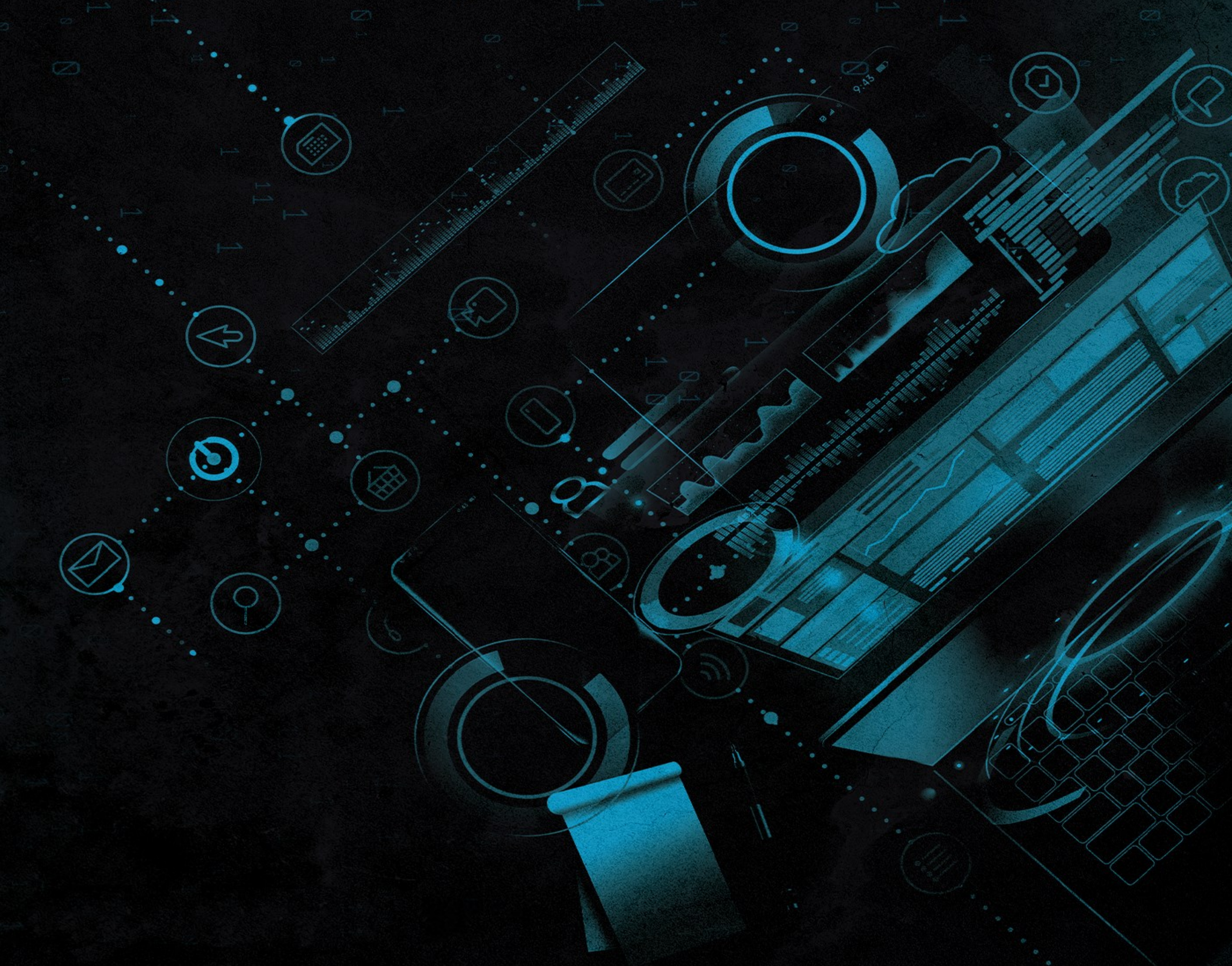
PROGRAMMING ESSENTIALS

A BEGINNER'S GUIDE WITH A HOLISTIC APPROACH



DENİZ GÜRSOY

Programming Essentials: A Beginner's Guide With A Holistic Approach

Paperback ISBN: 9798319237163

Hardcover ISBN: 9798280102392

Cover Design by Anze Ban Virant - ABV Atelier Design

First Edition: May 2025

Table of Contents

Preface 6
Who should read this book? 6
Teaching Style 7
More about chapters 7
What do you need throughout the book? 9
What is a container? 9
Creating the container 10
Stages of learning 12
Baking Pizza 13
1. Shu (守) - Follow the Rules 13
2. Ha (破) - Break the Rules 14
3. Ri (離) - Transcend the Rules 14
Shu Ha Ri in Learning Software Development 15
Executing Applications 16
What is an Application? 17
From text to output 17
Installing and executing applications 19
Runtime 20
Commands 20
Ls command 20
Arguments 21
Environment variables and the PATH variable 23
Running an application you developed 25
Logs 27
stdin (Standard Input) 28
stdout (Standard Output) 28
stderr (Standard Error) 28

More about commands 29
More about processes 30
In A Nutshell 30
Summary 32
Learn more 33
Create Hello World Application 36
Hello World Application 37
Summary 44
Learn more 44
Memory, Bits, Bytes, Variables and Constants 46
Memory, Bits and Bytes 47
Variables 50
Copy by value 56
Constants 57
Enumeration 59
In A Nutshell 59
Summary 60
Learn More 60
Types 62
Creating New Type 82
Casting 86
Arithmetic Operations 88
Order Of Operations 90
String Concatenation 91
Overflow 91
In A Nutshell 92
Summary 93
Learn More 94

Scopes 96
Methods 102
Packages 105
Summary 112
Learn more 113
Flow of control 114
Asking questions with relational operators 116
If statements 117
Logical Operators 121
Loops 123
In A Nutshell 129
Summary 131
Learn More 131
More to functions and methods 132
Pointers 135
Pointers as methods' receivers 140
Reading Input From User 142
Summary 144
Learn more 144
Abstraction 146
Interfaces 147
Summary 152
Learn more 152
Handling errors 154
Errors 156
Summary 162
Learn more 162
Testing 164

Summary 169
Learn More 169
Way to Go 170
A tour of Go 171
Calculator project 171
Debugging 171
Learn Git and Github 171
Practice 172
Appendixes 174
Formating verbs 174
Naming conventions 175
Essential Language Learning Checklist 176
References 178

Preface

I wanted to write this book because formal coding courses can be challenging for learners. The main difficulty is that learners are often taught the implementation or syntax rules first, without a proper understanding of the underlying concepts. Therefore, this book introduces the essential concepts that every developer should learn upfront. In this regard, it stands out from other books.

The book begins by introducing all the Linux applications and commands you will use when writing code. Before diving into their usage, it also explains how these applications are executed within the Linux operating system (OS). This approach will help you become familiar with the Linux system and the fundamental concepts that you will use in your applications.

Additionally, the book aims to show you that you don't need any complex tools or setups to start coding — an average computer is sufficient. Learning to code in this way will also help you understand how more complex tools used in professional environments work under the hood and what they do behind the scenes.

Who should read this book?

The book is aimed at absolute beginners with no prior programming experience. It is not suitable for those who do not plan to write and run all the sample code as described.

Some concepts included are considered advanced topics even for experienced learners. They are incorporated based on the organization of the material, presented at appropriate points in the learning process. For example, instructions on cross-compilation are provided, but this is a technique that is usually not needed for everyday development. Cross-compilation is included as a reference for those who may need it later and to demonstrate the underlying concept.

Additionally, certain concepts are introduced to familiarize readers with the Linux-based system. For instance, the **kill** command is explained to illustrate that almost any action can be performed using Linux commands.

Teaching Style

The book presents each concept as a case study to illustrate why it is needed and what problem it solves. After a concept is introduced, related statements will be highlighted in red, such as:

Learn how to access to arguments

The red statements represent items on your to-do list when learning a new programming language. After these red statements, the concept will be implemented through sample code in Go. This teaching style is called "**abstract teaching**" a term derived from "**abstraction**" in programming, which will be explained later in the book.

All red statements in the book are compiled into the "**Essential Language Learning Checklist**" section of the Appendix for future reference.

More about chapters

A typical programming course or training often begins with a chapter about variables. However, learners should also understand more about programming languages, how to execute applications, and how operating systems run these applications. Learning the concepts first provides a stronger foundation for coding and helps you communicate with others using the correct terminology.

Therefore, this book starts with the key concepts you should become familiar with first. That is why the title emphasizes a **holistic approach**.

The first chapter will explore a concept from Japanese martial arts related to the stages of learning. In the IT industry, learning is a continuous process—you will always encounter new tools, languages, or technologies. Understanding these stages is essential, as it helps you determine which stage you want to reach or consider sufficient for each technology you use.

The second chapter focuses on how to run applications. It explains what an application is and how it operates within the OS.

But why is this important? Throughout your software development career, you will use various applications such as editors, compilers, and interpreters. These tools help you develop, build, and deliver your code. To effectively use these tools, it's essential to understand how they are executed. This understanding will also assist you in running the applications you develop in the subsequent chapters.

Additionally, when an application runs in an OS, it follows a set of common steps regardless of the application or programming languages. Several concepts relate to these steps, and understanding them will help you access and use the concepts introduced in this chapter. As you learn about application execution, you will gradually become familiar with Linux commands, which will be used throughout the book.

The chapter also aims to help overcome any fear of working with a command-line interface or terminal. Furthermore, it demonstrates how developers transform code into executable applications. Since this process varies across programming languages, your code will go through different procedures depending on the language you use.

The third chapter will focus on creating a "Hello World" application from scratch. You will learn how to create files and directories, as well as how to write your code into these files. Finally, building on chapter two, you will learn how to run your code, similar to how applications are executed earlier. Use this chapter as a reference point while running code samples in the subsequent chapters.

All subsequent chapters will focus on various technical programming concepts.

Finally, the book concludes with a roadmap to guide your continued learning. One of the most critical skills you need to develop is **debugging**. Improving debugging skills is an essential aspect of programming, yet it is often underrepresented in programming education. While this book does not cover debugging in detail, your development tools will provide the necessary information to learn more about this skill. For additional guidance, refer to the debugging section in the last chapter.

What do you need throughout the book?

You can access everything you need with the help of Docker (https://www.docker.com). Docker is a helpful containerization tool. At this point, you do not need to know anything about Docker or containerization. Docker will let all readers perform the same tasks, regardless of their OS or computer.

There are two ways to use Docker. Firstly, you can install it on your machine. However, you may need to change some BIOS settings to enable virtualization. You can find information about how to install it on the internet.

Alternatively, use the online Docker playground (https://labs.play-with-docker.com/). It lets you create containers for a limited duration of three hours. Do the following to access the playground:

1. Go to https://labs.play-with-docker.com/).
2. Click login.
3. Select docker and log in after creating an account.
4. Click **Start**, and your session starts.
5. On the left menu, click **Add New Instance**.
6. Now, you can use the terminal on the right panel.

What is a container?

You do not need to know the detailed workings of containerization, but understanding what a container does and why to use it can be helpful. Think of a container as a brand-new, isolated computer that runs inside your existing system, with preinstalled software and files. The container provides access to all the necessary software and files for the book from your computer. The containers you create will include the following contents:

- Linux OS
- Go toolchain
- All files and software you need to practice

Creating the container

In the examples in this book, you will need to create a container from an image using Docker. Afterward, you can follow the instructions for each example.

You can create a container with the following command. Open the command prompt or terminal with superuser privileges. Then, run:
docker run -it --rm denizgursoy/dev-lab

```
user@local:~$ docker run -it --rm denizgursoy/dev-lab
Unable to find image 'denizgursoy/dev-lab:latest' locally
latest: Pulling from denizgursoy/dev-lab
d25f557d7f31: Already exists
8098e2c93c77: Pull complete
59f7b7f4c914: Pull complete
4f4fb700ef54: Pull complete
Digest:
sha256:442b359be2f88cab9ee1167b15b2aea74d3e59d65d8852d8650370bea58
0c9ed
Status: Downloaded newer image for denizgursoy/dev-lab:latest
/projects/src #
```

If you haven't downloaded the image (denizgursoy/dev-lab), Docker will download it automatically. The message "Status: Downloaded newer image for denizgursoy/dev-lab:latest" means the system downloaded the image. After that, Docker will start a container. Now you have a terminal from the container created. If you want to exit the container, you need to write:

```
exit
```

And then press Enter. You will be back on your command prompt/terminal.

Note:

When you exit the container, Docker will destroy the container because of the --rm argument. The files/directories you created will be destroyed with the container. To continue from the same container without losing files, use this command to create a named container:

```
docker run -it --name dev-lab denizgursoy/dev-lab
```

After you exit the container, you can resume it with this command:

```
docker start -ai dev-lab
```

That was all the setup you needed to do; if it is successful, you are now ready.