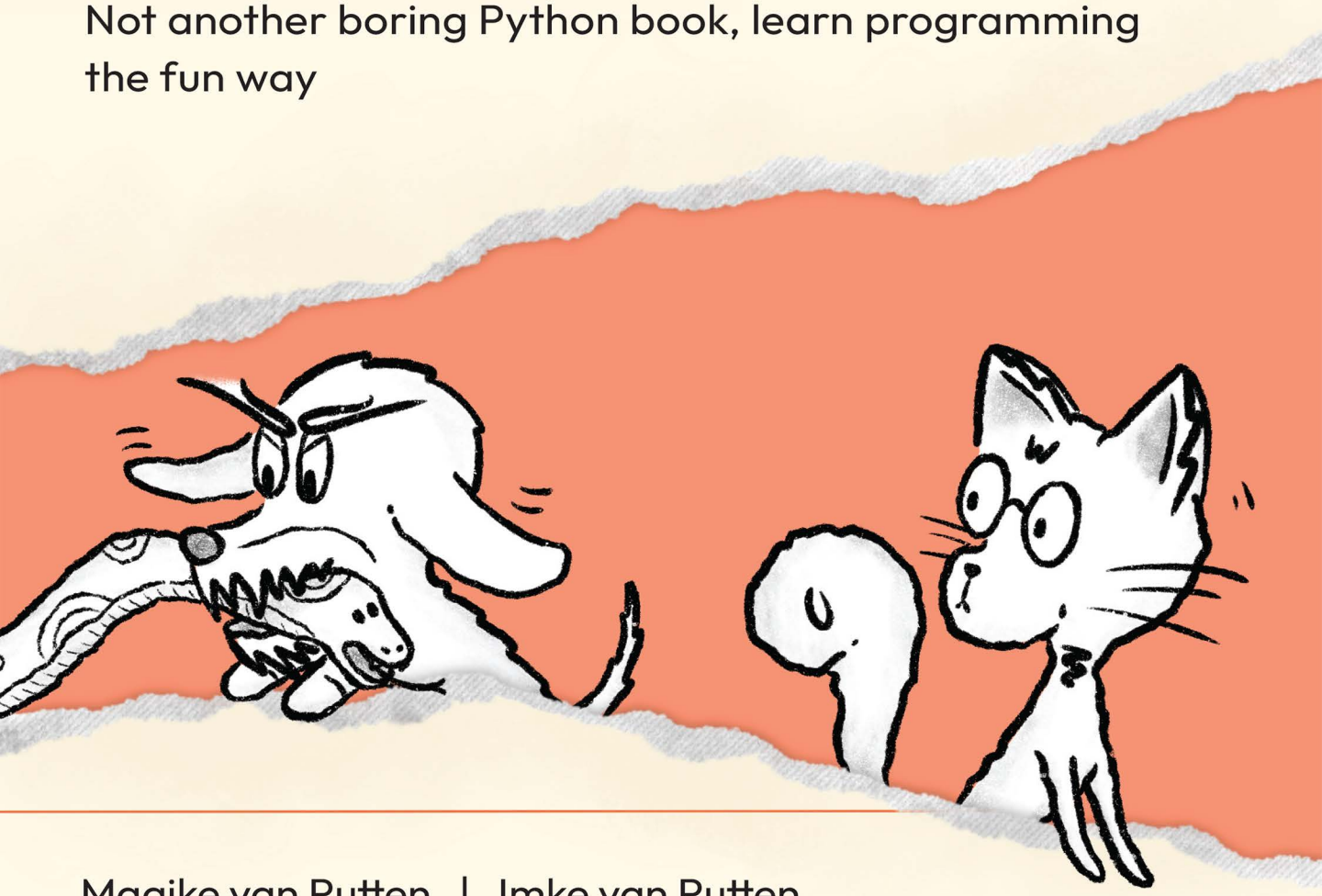
EXPERT INSIGHT

python™

<packt>

Python Illustrated

Not another boring Python book, learn programming the fun way



Maaike van Putten | Imke van Putten

Python Illustrated

Not another boring Python book, learn programming the fun way

Maaike van Putten

Imke van Putten

‹packt›

Python Illustrated

Written by Maaike van Putten

Illustrated by Imke van Putten

Portfolio Director: Kunal Chaudhari
Relationship Lead: Tushar Gupta
Project Manager: K. Loganathan
Content Engineer: Deepayan Bhattacharjee
Technical Editor: Irfa Ansari
Copy Editor: Safis Editing
Indexer: Manju Arasan
Proofreader: Deepayan Bhattacharjee
Production Designer: Ajay Patule
Growth Lead: Mansi Shah

First published: February 2026

Production reference: 1180226
Published by Packt Publishing Ltd.
Grosvenor House
11 St Paul's Square
Birmingham
B3 1RB, UK.

ISBN 978-1-83664-633-4

www.packtpub.com

Contributors

About the authors

Maaike is a software consultant and trainer with a passion for sharing her expertise to empower others in their careers. Her love for software development shows in the numerous software development projects she participated in and the many certifications she obtained. She has designed and delivered a broad spectrum of training courses catering to beginners and seasoned developers in Java, Python, C# and many other languages and frameworks. Next to that, she has authored multiple books and online courses through multiple platforms reaching over 500,000 learners across the globe.

Imke is a multi-talented content creator, co-teacher, and the creative force behind much of the illustrated content in *Python Illustrated*. With her unique combination of technical insight and artistic skill, she makes coding concepts fun, visual, and accessible to everyone.

Beyond her work in tech education, Imke channels her creativity and empathy into a variety of pursuits. She sews, designs, and runs wheelchair skills workshops that empower people with tools for independence, a mission deeply rooted in her own lived experience. Her passion for helping others navigate the world with confidence reflects the same care and thought she puts into making programming approachable.

About the reviewers

Dr. Seán Kennedy is a university lecturer with over 20 years of experience in teaching. His Ph.D. is in IT and he is OCP certified in Java. For the past 14 years, he has taught advanced Java on Ericssons' bespoke Master's program. He has co-authored two books with Maaike van Putten: *Learn Java with Projects* and *Java Memory Management*. Outside of work, he enjoys tennis, walking, TV, and nature.

Ricardo Bánffy is an experienced engineer living in Ireland, with a long career in varied engineering and management functions in areas ranging from advertising and aerospace, all the way to web applications. He is still looking for what to do that starts with Y and Z, for the sake of completeness. Owning a yacht would be a stronger candidate if Ireland's weather were nicer.

Ricardo has been reviewing books about Python and many other interesting topics, and enjoying it immensely. If you are reading this, maybe you should try it someday.

I'd like to thank first and foremost my handlers at Packt. God knows what I've put them through and their endless patience saved the day more than once.

Join us on Discord!

Read this book alongside other users, developers, experts, and the author himself.

Ask questions, provide solutions to other readers, chat with the authors via Ask Me Anything sessions, and much more. Scan the QR or visit the link to join the community.

`https://packt.link/deep-engineering-python`

Table of Contents

Chapter 3: Working with Conditional Statements 61

Chapter 4: Using Lists, Tuples, and Dictionaries 85

Preface

> *"They said magic isn't real, but that was before they typed their first Python script and watched as the impossible became possible."*
>
> *- Me, Zia (portrayed in the picture below)*

Okay, I might be setting the expectations a little bit too high there. However, you've made a great decision choosing Python. You could arguably not pick a more versatile and developer friendly programming language. Python has a relatively easy syntax that looks a lot like plain English.

Ignore her for now. On top of that, you can do a lot of things with it, just to name a few:

- Web development
- Desktop tools
- Data analysis
- Artificial intelligence (AI) and machine learning
- Automation of all sorts of repetitive tasks
- Scientific computing

The community that uses it is amazing. You'll find that most of the questions and problems that you're going to run into have been answered by some friendly stranger on the web. Next to that, there is so much made that you can use with Python. We sometimes say it has a "rich ecosystem". This means that many ready made bits and pieces of code are available for you and that will help accelerate development of whatever it is you're creating.

Before we can get there though, we'll need to understand the basic syntax. And that's what this book is for.

Who this book is for

Anyone who wants to learn Python without wanting it to be boring can benefit from this book. This book aims to be the most accessible and fun book to teach you Python. It should be a light read, without compromising the quality or depth of our explanation. But I'm aware that fun is not for everyone.

If you know another language already, this book might be easy for you. That's why I try to make it interesting with humor. If you are new to coding, this book will probably be hard. That's why I try to make it lighter with humor.

What this book covers

Chapter 1: Get Your Computer Ready to Code Python

You'll prepare your computer for Python. Setting up Python is easier than catching a laser pointer (which is surprisingly hard). By the end of the chapter, you have everything installed and written your first lines of code.

Chapter 2: Understanding Variables and Data Types

In programming, we often need to remember something. This is where variables come in. The chapter deals with how to create variables and what different data types there are. This way, you can keep track of the data needed in your program.

Chapter 3: Working with Conditional Statements

I can't help you make difficult decisions for you, but I can help you understand how to make difficult decisions in code. After this chapter, you'll be writing if, elif, else code to control the flow of your program.

Chapter 4: Using Lists, Tuples, and Dictionaries

In this chapter we'll deal with collections, ways to store multiple values in one variable. This is great for organizing our data and opens up a whole world in which we can deal with more complex types.

Chapter 5: Iterating with Loops

Loops let you repeat actions in your code. By the end of this chapter, you'll be able to avoid some duplicate code and automate more tasks.

Chapter 6: Writing Functions and Using Built-In Functions

This chapter is going to open up a whole new level of code organizations. Typically not the easiest concept, but if anything is like magic, it is this chapter. We are going to create our own little magic spells! These magic spells are called functions. When working with real projects, files get too big easily. We can split up files to organize functionality in separate files. These are called modules. Inside one module (yes file), we can import another module (uhuh file) to use the functionality. We will also see some common built-in modules and functions once we understand that module concept.

Chapter 7: Handling Files and Exceptions

Learn to open, read, and write files, so you can manage files. You'll be able to store and retrieve information from your programs with ease.

Chapter 8: Creating and Using Classes

This chapter is going to be a tough one! You'll take your first steps into the world of object-oriented programming. Think of classes as blueprints for your own custom types. You'll learn to create objects that can have attributes and behaviors.

Chapter 9: Understanding Inheritance

Why reinvent the wheel when you can inherit it? This chapter is going to build on the previous one and focus on better code organization. You'll discover how to build new classes that borrow features from existing ones.

Chapter 10: Debugging Our Code

Even the best of us miss a jump now and then. This chapter teaches you how to find and fix errors in your code, ensuring your programs land on their feet every time.

Chapter 11: Next Steps

Now that you've caught the red dot, what's next? I'll guide you on how to continue your Python learning, explore advanced topics, and maybe even catch that elusive laser pointer for good.

To get the most out of this book

My human says this all the time: coding is something you learn by doing. Of course, just reading this book will give you some insights. But to make the most out of it, it's highly recommended that you code along.

In the chapter, you'll find coding examples. These are also on the GitHub repository:

`https://github.com/BrightBoost/python-illustrated`

You can use git properly, or just copy them from there and paste them in your editor. At the end of each chapter, you'll find a list of exercises and a quiz to test your understanding of the topics discussed. The answers and explanations are in the back of the book (and on Github of course, so you can run and verify them easily).

Get in touch

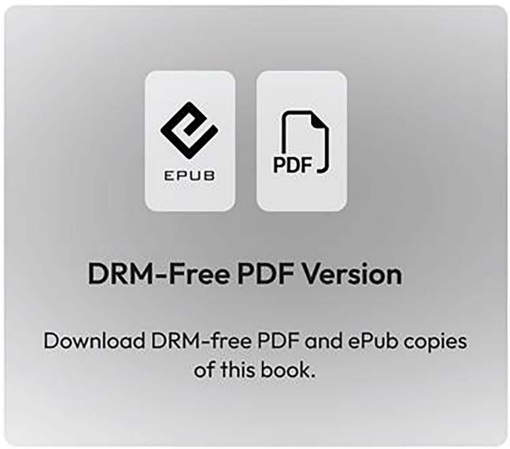
If you are stuck and can't continue your learning to code ambitions. Please don't be stuck. Reach out to me, as long as I can handle the load and it doesn't get too busy, I'll help you get unstuck.

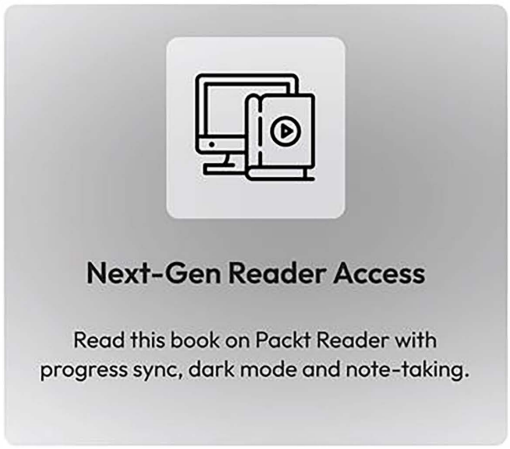
And don't ever be afraid to ask stupid questions. There's no such thing. It's stupid to not ask the question.

Free benefits with your book

This book comes with free benefits to support your learning. Activate them now for instant access (see the "*How to Unlock*" section for instructions).

Here's a quick overview of what you can instantly unlock with your purchase:





How to unlock

Scan the QR code (or go to `packtpub.com/unlock`). Search for this book by name, confirm the edition, and then follow the steps on the page.

UNLOCK NOW

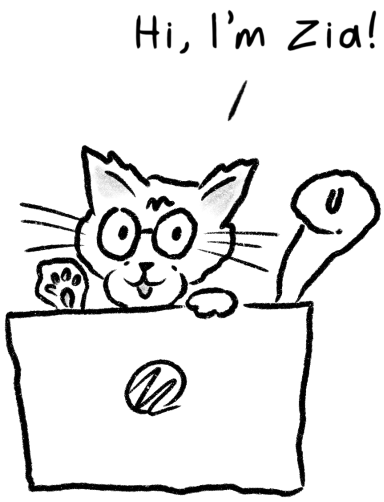
Note: Keep your invoice handy. Purchases made directly from Packt don't require one

Share your thoughts

Once you've read Python Illustrated, we'd love to hear your thoughts! Please `click here to go straight to the Amazon review page` for this book and share your feedback.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.

Introduction



Hi, my name is Zia.

Yes, I'm a cat.

You might think that cats have no business writing books on Python. Well, you're wrong. If you own a cat, you might know that we love to be on your keyboard. Partly because it's warm (yes, it feels really good, not going to lie), but also because we love to type all sorts of things.

My human is a teacher in the tech space. She accidentally taught me Python while I was on her lap during her lessons. Fast forward to now, and I'm employed (100% remotely, of course) at a big tech company. I cannot tell you which one because I've signed an NDA. Also, I always use my human filter on Zoom and they cannot know I'm a cat. I've tested the waters with HR, and exposing that secret would get me fired.

Oh right, one more person I need to introduce.

Wiesje is here too. It's a Dutch name, you'd pronounce it something like Weesha. She is the cutest dachshund dog, and I'm proud to say that she's my cousin. Not sure why, but she doesn't seem to share the joy of us being related and can come across a little grumpy.

Once you get to know her better, you'll find that she's actually very sweet. She can't code Python yet, so she'll join you in the learning process.

Anyways, Python is easy, and I can teach you. But wait... What even is Python? Python is a programming language. A programming language is a set of instructions that can be used to communicate with computers. Just like how we use English to talk to each other, we use programming languages to tell computers what to do. Python is one of the easiest languages to learn and use, which makes it great for beginners.

To give you some more concrete examples, here are a few things you can do with a programming language:

- Automating repetitive tasks such as renaming files
- Making applications, for example a calculator
- Creating a game, just to name something... a number guessing game

Spoiler alert! We are going to do all of these things in this book (and more of course). Here's an overview of the things we're going to do:

- Get your computer ready to code Python
- Understanding variables and data types
- Working with conditional statements
- Using lists, tuples, and dictionaries
- Iterating with loops
- Writing functions and using built-in functions
- Handling files and exceptions
- Creating and using classes
- Understanding inheritance
- Debugging our code
- Next steps

Okay, time's valuable, let's get started getting your computer ready to code.

1

Get Your Computer Ready to Code Python

To make the most out of this book, it's best to follow along on your computer. But you can't just go ahead and write code on any computer; it's often required to get your computer ready to go. You do this by installing some necessary programs. We'll start by making sure you're ready to code. After this chapter, you'll have written your first line of code and run your first program. Exciting, isn't it?

Exploring Python features

Before we get too excited and jump into the installation straight away, let's take a sneak peek at what Python can do. Think of this as sniffing around before committing to the full pounce.

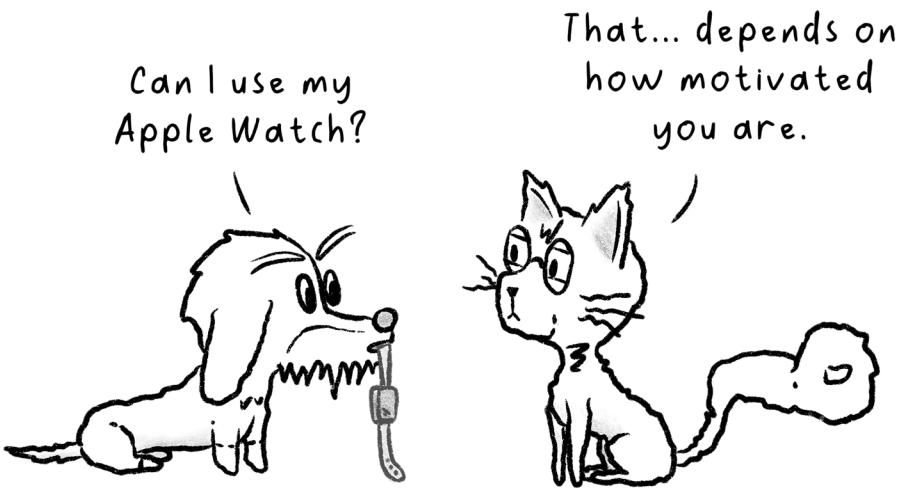
In short, here is why you'd want to learn Python as your programming language:

- Python's syntax is designed to be readable and straightforward. For a programming language, it's simple. It's like a warm sunbeam on the floor: inviting and comfortable.
- From web development to data analysis, Python wears many hats. Or should I say, many collars? It's a language that can be used for many different purposes and is therefore a great pick.
- Python has a supportive and loving community. An active community means tons of libraries and frameworks to play with. If you get stuck, there's practically always someone who has asked the same question before. And if not, they're rushing to answer your question on Stack Overflow. Stack Overflow is a forum that is used to solve mostly coding issues.

Setting up your system can be tricky and even a little frustrating if it doesn't take the happy path directly, but I promise you it's worth it!

Technical requirements

What do you need to run Python? Well, if you really, really had to, you could even learn it on your phone, tablet, or Chromebook. It's not ideal, but if you're motivated enough, you can make it work. At the end of this chapter, I will show you how to set that up. I just wanted to start by telling you not to be discouraged by not having fancy gear; we can make this work with almost anything. Just skip to the end of this chapter if that's you.



While phones and tablets might be possible, they're far from ideal for coding Python. A bit more of a mainstream setup for learning Python is some sort of personal computer. Ideally, you have a laptop or PC with the following specifications:

- Windows, macOS, or Linux as the operating system. Yes, even that old laptop in your closet might do.
- 8 GB RAM is comfy, but if you have 4 GB RAM and an extra splash of patience, it can work as well.
- An Intel i5 or equivalent processor is nice. Of course, anything faster is even better. An i3 will work too, but be prepared for some extra coffee breaks while things load.
- 2 GB of free disk space should suffice. That's like, what, a dozen cat videos?
- Install rights – sometimes, when it's a laptop from work or school, you might not be free to just go ahead and install applications without permission. If you can't install, hop to the end of this chapter, where I discuss how to use Python in the browser.

As you can tell, there's some flexibility here. I'm a big fan of using whatever you have. Once it gets more serious and you have the budget, upgrade your laptop. Python is not too heavy, but some of the programs we use for writing our Python code are. Not having the recommended laptop but using a more basic Python editor instead can be a perfectly fine solution to get started.

If you follow the steps, your system should be ready. The following installation steps are for Linux, macOS, and Windows laptops. For all of these steps, understanding and being able to use the terminal is important. Let's talk about that first.

The terminal

In movies, you might have seen scenes where hackers are doing their magic work, breaking into some sort of system. The application that they typically use when doing this is the terminal.

We might be less cool than those villains or superheroes in the movies, but we will use the terminal as well. When writing and running code, you'll be using a terminal every now and then, or in some cases quite often.

At this point, you might not know what a terminal even is. In the next image, you can see a picture of a terminal.

A terminal is a different way of controlling your computer. As a user, you are used to giving your computer instructions with the mouse and the keyboard. For example, if you want to open your mailbox, you can click on the application icon. This mouse click is an instruction your computer understands, and it will proceed to open the mailbox.

By typing text in the terminal, you can also give your computer instructions. As a bonus, it makes you look like one of those cool hackers from the movies. The terminal is the application that you are using to type these commands; it's sometimes also called the command line.

Here are some things you can do with the command line:

- Create, edit, and delete files
- Install applications
- Run applications
- Check versions
- And a lot more!

We'll be using the command line to install Python, add Python libraries, and run Python files. In order to use the command line, you'll have to know which commands to use. It's very possible that you don't know any. No problem! I will tell you what to type and what it means as we go.

Opening a terminal

Before we install Python, we'll need to check whether Python is already installed. To do that, we'll need a terminal. How to open a terminal depends on the system you're using.

For **Linux**, depending on your distribution, you can use the shortcut *Ctrl* + *Alt* + *T*. If that doesn't work, you can open the application menu (sometimes called **Activities**) and look for the applications. There, you can search for terminal. Click on the terminal icon. That should open the terminal.

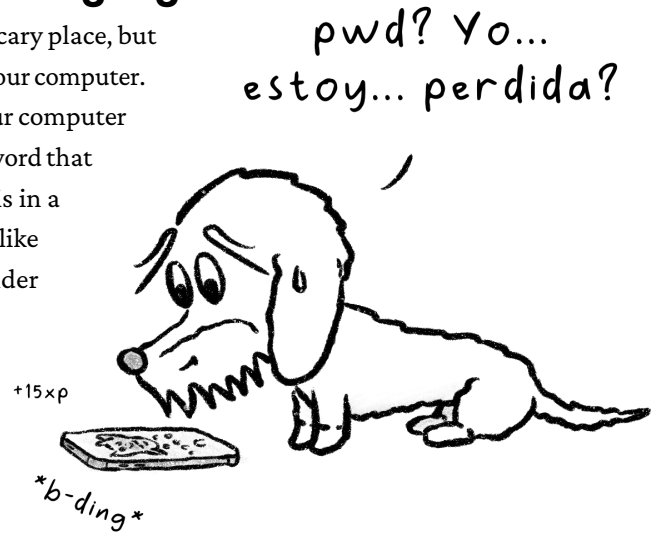
For **Windows**, there are two built-in terminals: the command prompt and PowerShell. Either one of those is fine. You can open the command prompt with the shortcut *Windows button* + *R*. Or, you can type `cmd` in the search box in the start bar. If you don't have a search box in the start bar, you'll have to click on the Windows icon first and type it in the search box that pops up as part of the menu. Again, the terminal should open.

Lastly, for **macOS,** I like to use the Spotlight search. This can be done by pressing *Cmd* + *Space*. This brings up a search box. In there, we can type `Terminal` and press *Enter*. If that doesn't work, you can use Launchpad. Click the Launchpad icon in your dock and use the search bar at the top to look for **Terminal**. Click the Terminal app and it should open.

Now that you have the terminal open, let's give it a spin.

Speaking the terminal's language

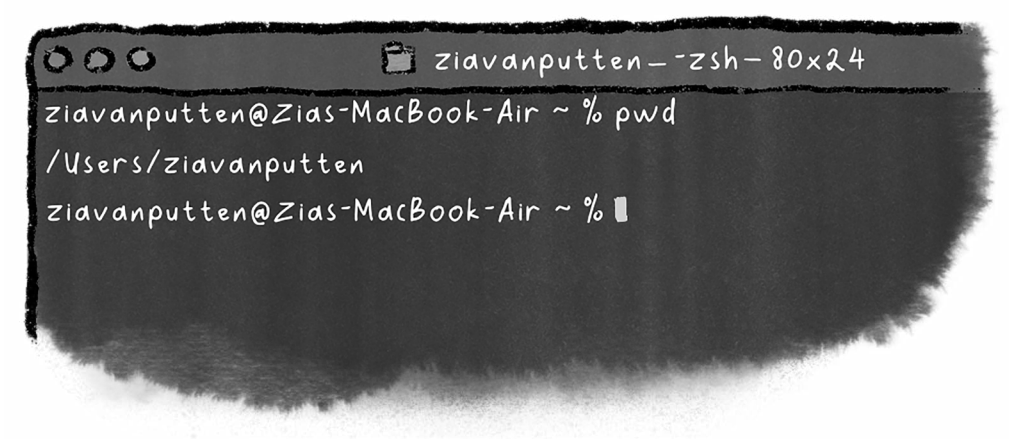
The terminal might look like a dark and scary place, but it's really just a simple conversation with your computer. You type a command, press *Enter*, and your computer responds. Each command is like a magic word that makes something happen. The terminal is in a certain folder on your computer. This is like when you have File Explorer or the Finder application open; you are in the folder that you see.



On mine, in the image with the command line, we are at the location `~`. Which means that I'm currently in my home directory. The home directory is known as the default user folder. If you're not sure where you are, you can ask the terminal where you are. Here's how it's done on macOS/Linux. In the terminal, type the following:

```
pwd
```

Then press *Enter*. This `pwd` stands for "print working directory." In my case, you can see I'm in the user's home directory, the user folder. This is the default folder where I am when I start the terminal like this.



On Windows, you can see the directory by looking at the terminal window. It tells you where you are.

If you want to change the directory, you can. There are three things you can do when it comes to changing directories: step inside a folder, step up a folder to the parent directory, or go to a complete path that starts from the root of the file system. You change the directory with the `cd` command. After the command, you specify where to go.

You can step inside a folder that is in the directory where you are. You can do this with the following:

```
cd foldername
```

The `cd` stands for "change directory." After executing the command, you're in the subfolder with the name `foldername`. You can go to the parent folder. This is done with the following:

```
cd ..
```

You can go to a location that you know the absolute path of. The absolute path starts from the root of the filesystem. This example is for Windows:

```
C:\Users\Zia
```

This example is for Linux and macOS:

```
/home/zia
```

If you want to know which files and directories are in the directory that you are in, there's a command for that too. On **macOS/Linux**, type `ls` and press *Enter* to see a list of the files and directories. On **Windows**, type `dir` and press *Enter*.

Don't worry if this is new; you'll get the hang of it with practice. It's time to find out whether Python is already installed on your system using the terminal!

Is Python already purring on your system?

We can check whether Python is already curled up inside your computer – yes, I just mean installed. In the terminal, type the following commands and press *Enter*. These commands are like asking, "Hey computer, do we have Python here? If so, which version?"

On Windows, it is probably not installed:

```
python --version
```

If that doesn't give a number, try the following:

```
py --version
```

On macOS/Linux, there might be a different version – for example, 2.7. In order to find that out, run the following:

```
python --version
```

If that doesn't give a version number that starts with a 3, try typing the following:

```
python3 --version
```

If you need to add the 3, that's no problem; `python3` ensures you're using Python 3. It's like making sure your human is opening the right can of food (clearly tuna), not the old sardines.

The syntax of Python 2 is very different from Python 3; you really need to have Python 3 installed for this book. Don't worry if it's not installed – we'll walk you through it.

Understanding the response if Python is found

If Python is installed, you'll see something like this:

```
Python 3.12.6
```

If that's the case, purrfect! This means Python is already lounging around, ready to play. You might wonder what these numbers mean. Let's break it down:

- **3**: The major version. We need Python **3**, as Python 2 is like an old scratching post: still there, but not as much fun.
- **12**: The minor version. Each minor update brings new features.
- **6**: The micro version. These are small fixes and tweaks.

So, `Python 3.12.6` tells us you're using Python version 3, update 12, patch 6.

Using the right version of Python is important because code written for Python 3 might not work in Python 2. Also, newer versions have cool features that we'll use in this book.

For our adventures, we need **Python 3.10** or newer. If your version is older, some of the exciting things we'll do might not work.

Understanding the response if Python isn't found

You might get a message such as the following:

- 'python' is not recognized as an internal or external command, operable program, or batch file (Windows)
- command not found: python3 (macOS/Linux)

Don't worry! This is not an error. It just means Python hasn't moved in yet. We'll invite it over in the next section.

Windows users: a tiny hiccup

Sometimes on Windows, typing `python` might open the Microsoft Store. That's your computer's way of saying Python isn't installed, or it's not set up quite right. No biggie – we'll sort it out soon. Let's close the Microsoft Store for now, and I'll talk you through it in the rest of this chapter.

So, if Python is already installed and up to date, you're ahead of the game! If not, we can install Python. It's easier than convincing me to chase a laser pointer.

Installing Python

Now, let's get Python installed on your machine. This works a little differently for each operating system. Let's walk through it.

First, these are the steps for **Windows** users:

1. We'll start by downloading Python:
    - Go to the official Python website: `python.org`
    - Click on the **Download Python 3.x.x** button. There are many versions that will work, but I'd recommend grabbing the latest stable release. The x.x is not literal; I just cannot include it in a book because the numbers change so fast.

2. Now that we have the executable downloaded, we can run the installer:
    - Open the downloaded `.exe` file.
    - **Important**: Check the box that says **Add Python 3.x to PATH**. This makes it easier to run Python from the command line.
    - Click on **Install Now**.
3. Verify the installation:
    - If you have the Command Prompt still open, close it.
    - Open Command Prompt (search for `cmd` in the **Start** menu) (again).
    - Type `python --version` and press *Enter*.
    - You should see something like `Python 3.x.x`

Yes, Wiesje, we'll do **macOS** too. There are different ways to do it. You could use Homebrew if that's set up. You can find out whether you have it by running the following:

```
brew --version
```

If that gives you something such as `Homebrew 4.5.13`, it has been set up.

In that case, it's just a matter of doing the following:

1. Open the command line and type the following:
    - `brew install python3`
2. Verify it's installed by doing the following:
    - Type `python3 --version` and press *Enter*.
    - You should see the version number.

But if you don't use Homebrew, you can just download Python and install it:

1. Download Python:
    - Go to `python.org`
    - Download the latest Python 3 installer for macOS.
2. **Run** the Installer:
    - Open the downloaded `.pkg` file.
    - Follow the installation steps.
3. Verify installation:
    - Close and reopen Terminal if it's still open; otherwise, just open it.
    - Type `python3 --version` and press *Enter*.
    - You should see the version number.

Last but not least, let's talk about how to do it for **Linux** users. Most Linux distributions come with Python pre-installed. But it's often Python 2, which, while still lovable, is not what we're going to use.

1. Update the package lists:
    - Open Terminal.
    - Run `sudo apt update` (for Debian-based systems).
2. Install Python 3:
    - Run `sudo apt install python3`
3. Verify installation:
    - Type `python3 --version`
    - You should see the version number.

If that doesn't work, you can download the installer package from `python.org`, like Windows and macOS users, and proceed to install that. After installing, the version number should show when entering the following command in the terminal and pressing *Enter*:

```
python3 --version
```

At this point, we're ready to use Python. Exciting adventures are ahead – let's move on and write our first program!

Writing our first program

It's time to actually start coding. Traditionally, programmers start with a "Hello world!" program in whatever new language or framework they start using. But since I'm a cat, let's make it more relevant. Python only needs one line of code for the first program, and that makes it one of the shortest first programs you can write. If you ever try a language such as C++, Java, or C#, you'll really appreciate how straightforward Python is. Let's see the steps for creating our first little program.

First, we need to create a new file. You know how Word documents have the `.docx` extension, text files have a `.txt` extension, and ZIP files have a `.zip` extension. A Python file has its own special extension as well, the `.py` extension. Let's create a file and call it `hello.py`.

How do we create a file with this extension, you might wonder? Well, there are multiple options again. I'll walk through one option for each operating system that we've been talking about. It's easy to start by opening a text editor. If you have downloaded Python from the website, it probably came with a program called IDLE, which is great for this. Otherwise, you can use Notepad (Windows), TextEdit (macOS), or any basic text editor that your system has. Type the following line in this file:

```
print("Hello humans.")
```

Next, we're going to save the file as `hello.py`. Make sure it's saved with the `.py` extension and not `.txt`. Sometimes, it accidentally saves as `hello.py.txt`. This is still not correct, and that will not work. Also, for TextEdit (macOS), you don't have the option to choose the right extension – go for `.html` for now.

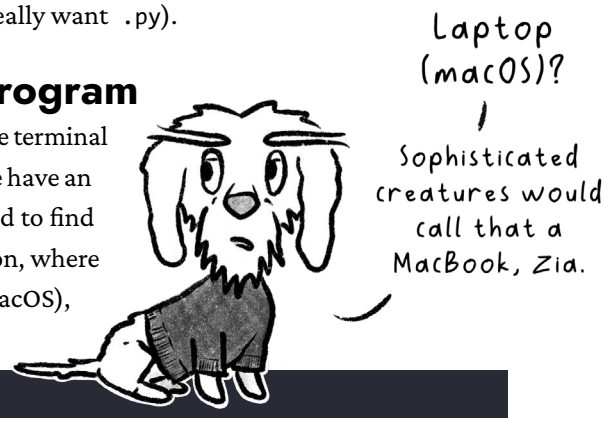
(If you're struggling for an unnecessarily long amount of time with the extension, you might consider just leaving it, as we'll take another approach soon. You can download the file from my GitHub instead.)

Then there's one extra thing you need to do: you want to make sure it's plain text, otherwise the quotes will be the wrong quotes. What do I mean?

“ is not the same as " . Do you see the first ones curling a little? You don't want those curly ones. So what we need to do is to go to the top menu bar, to **Format** and choose **Make plain text**.

Then, lastly, we can rename the file and just modify the extension. On Windows, you'll need to enable adding the extension to the file name to be able to do that (don't mind the warning about changing the extension that follows – we really want `.py`).

Running your first Python program



In order to run our program, we're going to need the terminal again. So, let's open the terminal as a first step. We have an additional challenge before we can run it: we need to find the file. Let's start by asking ourselves the question, where is the file saved? Let's say that, on my laptop (macOS), it's stored here:

```
/Users/zia/Documents/pythoncourse
```

The easiest way to go there is by entering the full (absolute) path in the terminal, like this:

```
cd /Users/zia/Documents/pythoncourse
```

Let's make sure we're in the correct folder by asking where we are:

```
pwd
```

If you want to see the files that are in there, you can run the `dir` command for Windows and the `ls` command for macOS and Linux.

Now that we're in the correct location, we'll run the program. We do this by typing the `python` command, this time not followed by `--version`, but by the name of the file. So please go ahead and type `python hello.py` for Windows and `python3 hello.py` on macOS/Linux. Next, press *Enter*.

We should now be able to see the output. The output should be `Hello humans.` printed in your terminal. And that's it! I suppose congratulations are in order; you've just written and executed your first Python application.

So far, we've been using tools that our computer has. And that's fine – it will work. However, there are nicer tools to work with Python. Let's talk about these tools, called IDEs, next.

Working with an IDE

As we have seen, writing code in a basic text editor works... But it's as if you're eating dry kibble when you could have had tuna. The tuna in this case would be the IDE. **IDE** stands for **Integrated Development Environment**. This special application, meant for coding, offers features such as syntax highlighting, code completion, and debugging tools.

It's a little bit like choosing between Word and Notepad for writing books. Notepad will work, but it's nicer to use a slightly more elaborate writing tool. Just like you have different applications that you could use for writing a book, there are different excellent options for writing Python. Here are some great choices that you can install on your computer:

- PyCharm (by JetBrains)
- IDLE
- Atom
- Sublime Text
- Visual Studio Code

If you have a somewhat more basic machine, it might be a great option to work with IDLE. If you downloaded Python from the python.org website, this probably came with it. In your programs, you can search for IDLE, and the editor will open.

For this book, I'll be using Visual Studio Code. It's typically a bit more lightweight than PyCharm, but when I'm developing web applications with Django (a Python framework), I do prefer PyCharm. When I'm writing a quick automation script or teaching a course, I'll always opt for Visual Studio Code.

Introducing Visual Studio Code (commonly called VS Code)

VS Code is a great choice for an IDE in general. It is free and relatively lightweight for your computer. It supports so many languages and frameworks. You won't have to worry about frameworks yet; that is what we'll use when we make more complex applications, such as web apps, later. It consists of a basic application that can be extended with many (free) plugins, called extensions. With the right extensions, VS Code is a really great tool for writing Python. Let's walk you through how to install it.

First, we need to download it.

- Visit `code.visualstudio.com`
- Download the version for your operating system.
- Now that it's downloaded, it's time to install VS Code:
- Run the installer and follow the prompts.
- After it's installed, open VS Code.

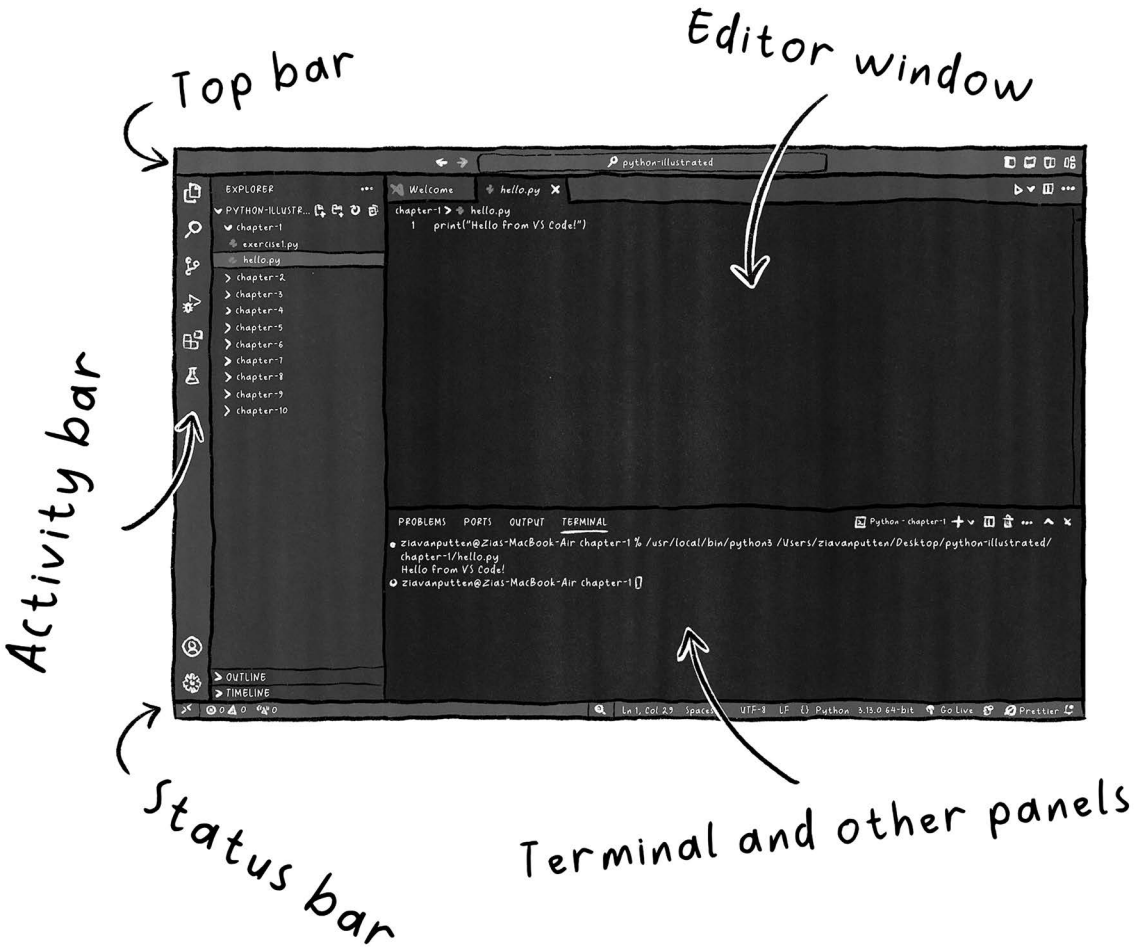
We'll install the Python extensions later – that's our last step. But let's take a moment to enjoy the view and explore what VS Code looks like.

What's where?

When you first open VS Code, you might feel a bit like when I moved to my new house – so many places to explore! Let's break down the main areas. VS Code is quite heavily focussed on the **activity bar**. This is the vertical toolbar on the very left. (It's shown in the picture on the next page.) In here, you'll see different icons:

- **Explorer (files icon)**: Access your files and folders here. This is where you'll have an overview of all the files in your project.

- **Search (magnifying glass)**: Search for text across your entire project. If you have lost some lines of code and you don't know in which file they are, you can search across multiple files here.
- **Source Control (branch icon)**: Manage your code versions with Git. Don't worry if you don't git... uhh get... that yet.
- **Run and Debug (bug with a play button)**: Run your code and debug errors. This is the place that helps you catch bugs. I don't mean spiders and mosquitoes, but tiny mistakes in your code. We call those bugs too.
- **Extensions (four squares)**: Find and install extensions to add new features. This is where we'll go in a little bit to fine-tune VS Code for helping us to write and run Python.



Depending on changes within VS Code and extensions you have installed, there could be more or fewer icons. But in general, whenever you click on one of these icons, a panel next to it (called the sidebar) will open with some more options or show information. If we move a little further to the right, we get the main area. This is the **editor window**. The central area where you'll write your code. You can open multiple files in tabs, just like browsing the web.

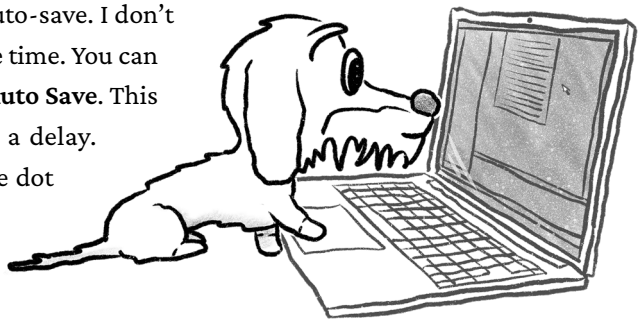
If you run your code, the focus will shift to the bottom. This bottom area houses the built-in terminal, debug console, output, and problems tabs. You can toggle this panel with shortcuts or menu options. I'm quite a fan of the built-in terminal; it's so nice not to have to change windows, but just access it right where I was working already.

If we move down a little further, we'll find the **status bar**. This is the strip at the very bottom. It shows useful information such as the current programming language, line and column numbers, and the Git branch (again, don't worry if this doesn't mean a lot to you yet).

There are some really cool things to say about VS Code, such as the command palette. This can be accessed via *Ctrl* + *Shift* + *P* (Windows/Linux) or *Cmd* + *Shift* + *P* (macOS). This is a feature that's a lot like a magic wand. You can type commands to do almost anything without touching the mouse. There are also a lot of great shortcuts and features, but I don't want to spend too much time on them, even though I'm very enthusiastic.

Alright, alright. Just this – there are lots of handy shortcuts. In the top bar, you have the regular menu. Next to the name of the options, you'll see the shortcuts. I'll share how to use VS Code as we walk through the book. Oh, and this: you can personalize VS Code as well. Go to **File** > **Preferences** > **Settings** (Windows/Linux) or **Code** > **Settings** > **Theme** (macOS). There are a lot of options here; you can change the theme to get a look and feel you're comfortable with.

One of my personal must-haves is auto-save. I don't want to have to save manually all the time. You can enable auto-save by going to **File** > **Auto Save**. This saves your files automatically after a delay. If files are unsaved, they have a little dot behind the title in the tab. If you want to be in full control, don't enable it.



Let's wrap up our VS Code setup by installing the Python extensions that we'll need. First things first, let's open up the right view. Click on the Extensions icon in the Activity Bar (this is the one that looks like four squares).

Here, we can browse extensions and install them. In the search bar on the extensions sidebar, search for `Python`. Open the one by Microsoft and click on **Install**. If you have made a Python file in VS Code already, it may have prompted you to add the extension, and it's possible that you already have it.

Creating and running a program with an IDE

Let's recreate our `hello.py` program in VS Code. In order to do this, we'll open a folder first. We have different options to do this. If you click on the file icon, there's probably an **Open Folder** button that allows you to navigate to a folder that you'd like to open. It's probably best to create a special folder for the examples in this book. Let's say you choose the name `python-illustrated`. You'll navigate to this folder (or create it on the spot), select it, and click **Open**.

Then, when the folder is open, make sure to be in the **Explorer** view by clicking on the files icon on the left. Next to the name of the folder, you will see icons. One of them is to create a new file. You can also right-click the mouse and choose **New file** in the Explorer view. Name the file `hello.py`.

It should open in the main area, the editor. In there, type the following:

```
print("Hello from VS Code!")
```

Make sure that it's saved (there should not be a dot beside the name of the file in the tab, which indicates it's not saved). Now we're ready to run the program. Of course, why have only one way to run the program? We have several ways:

- **Option 1**: Right-click in the editor window and select **Run Python File in Terminal**.
- **Option 2**: Press *Ctrl* + *Shift* + *P* (Windows/Linux) or *Cmd* + *Shift* + *P* (macOS) to open the command palette, type `Run Python File`, and select the option that pops up. If the shortcut doesn't work, open the terminal using the user interface of your operating system.
- **Option 3**: Add the Code Runner extension. Use the play button icon in the top-right corner of the editor. If you have the extension already, you'll see the play button in the top-right corner.

Any of these ways should show the terminal within VS Code with the following text:

```
Hello from VS Code!
```

And that's it. You're all set for this book.

Before we move on, let's talk about what to do when you don't have a computer on which you can install Python.

Alternatives for when you can't use a computer

Maybe you're on the go, or perhaps your pet parrot spilled water on your laptop (it happens). Or, you have never had a PC or laptop. As long as you have a device with a browser, you can use any of these alternatives.

It all comes down to online Python interpreters. Not all of them are free, and they have different options. Some allow you to create an account where you can store and manage full projects (those are typically paid). There are basic online interpreters where all your progress is lost with a page refresh. These are typically free.

- **Repl.it**: Visit `repl.it` to write and run Python code in your browser. You'll have to sign up for an account.
- **Google Colab**: This one is great for data science and machine learning, accessible via `colab.research.google.com`. However, it is perfectly fine for most of the examples in the book.

There are also mobile apps that you can install. Current examples of these are as follows:

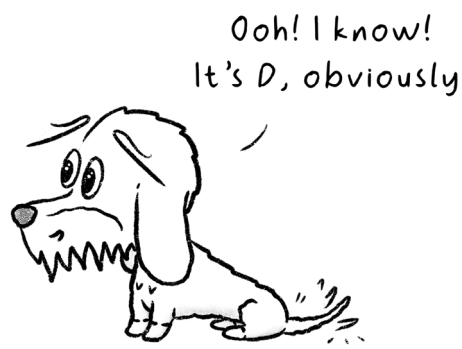
- **Pythonista (iOS)**: A full Python IDE for your iPhone or iPad
- **QPython (Android)**: Allows you to run Python scripts on your Android device

These alternatives aren't pawfect, but they'll do.

And there you have it! We are all ready for it. Remember, every great coder started with a simple "Hello world!" or, in our case, "Hello humans." If you want to try something, just do it. Experimenting and playing around are great ways to learn unexpected things. Coding is something you'll learn. As most people know, it requires logical thinking, but what is less known is that it will require a good dose of creativity as well.

Before we move on to the next chapter, about variables and data types, let's get some more practice by doing the quiz and exercises first.

Quiz



1. What does `cd` mean in the terminal's language?

    a. Command Directly
    b. Change Directory
    c. Create Directory
    d. Cute Dachshund

2. What does the `pwd` command do in the terminal?

    a. It shows you the folder you're in
    b. It prints all files in the current folder
    c. It runs your Python program
    d. It allows you to change your location using an absolute path

3. Wiesje is trying to speak to the terminal on her macOS laptop – sorry, MacBook. She runs the `dir` command to see which files are in her current folder. Why is this not working?

    a. She hasn't selected a folder first
    b. There might not be any files in the current folder
    c. MacBooks don't have a terminal
    d. She's using a Windows command

4. What does the ~ mean in the terminal's language?

    a. Python
    b. Current location
    c. The home directory
    d. It's purely decorative

5. True or false? You need a modern computer to learn how to code Python.
6. What is an IDE?