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.

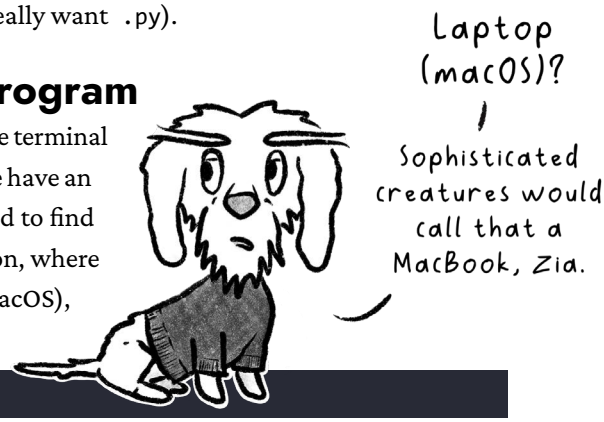
(If you're struggling for an unnecessarily long amount of time with the extension, you might consider just leaving it, as we'll take another approach soon. You can download the file from my GitHub instead.)

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.

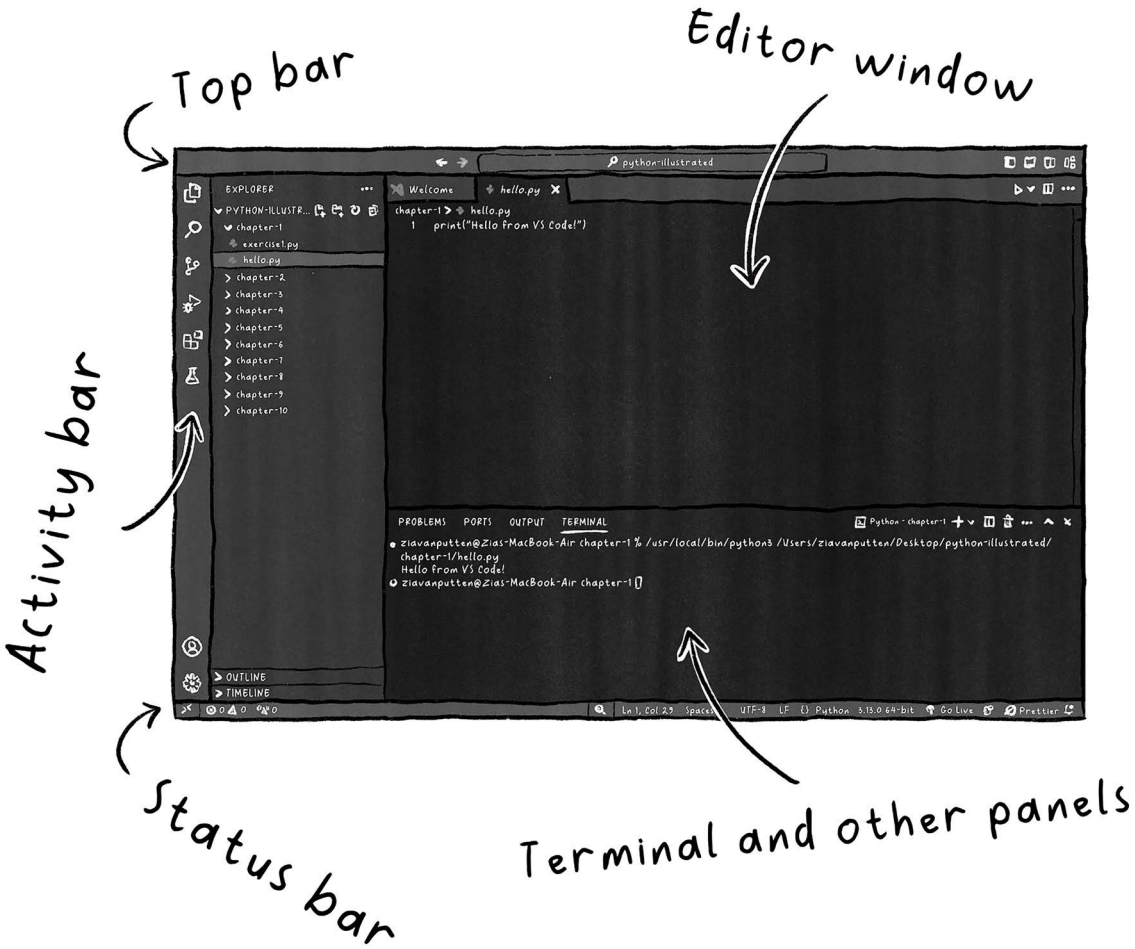
We'll install the Python extensions later – that's our last step. But let's take a moment to enjoy the view and explore what VS Code looks like.

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.

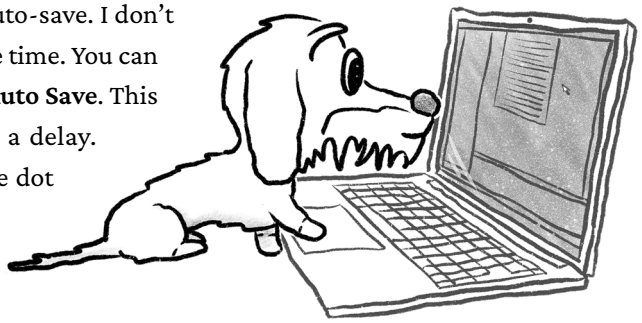
If you run your code, the focus will shift to the bottom. This bottom area houses the built-in terminal, debug console, output, and problems tabs. You can toggle this panel with shortcuts or menu options. I'm quite a fan of the built-in terminal; it's so nice not to have to change windows, but just access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

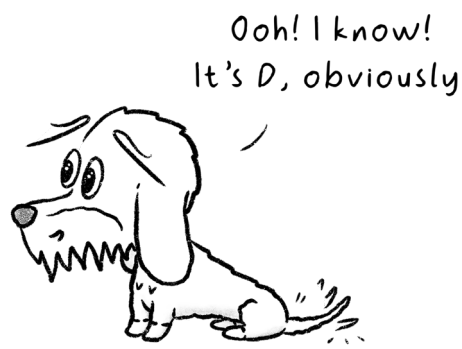
- **Pythonista (iOS)**: A full Python IDE for your iPhone or iPad
- **QPython (Android)**: Allows you to run Python scripts on your Android device

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?