

Environment Variables

**Tip**

If you already know what "environment variables" are and how to use them, feel free to skip this.

An environment variable (also known as "**env var**") is a variable that lives **outside** of the Python code, in the **operating system**, and could be read by your Python code (or by other programs as well).

Environment variables could be useful for handling application **settings**, as part of the **installation** of Python, etc.

Create and Use Env Vars

You can **create** and use environment variables in the **shell (terminal)**, without needing Python:

Linux, macOS, Windows Bash

```
bash
```

You could create an env var MY_NAME with

```
$ export MY
```

[fast](#) →

Windows PowerShell

```
bash
```

Create an env var MY_NAME

[fast](#) →

```
$ $Env:MY_N █
```

Read env vars in Python

You could also create environment variables **outside** of Python, in the terminal (or with any other method), and then **read them in Python**.

For example you could have a file `main.py` with:

```
import os  
name = os.getenv("MY_NAME", "World")  
print(f"Hello {name} from Python")
```

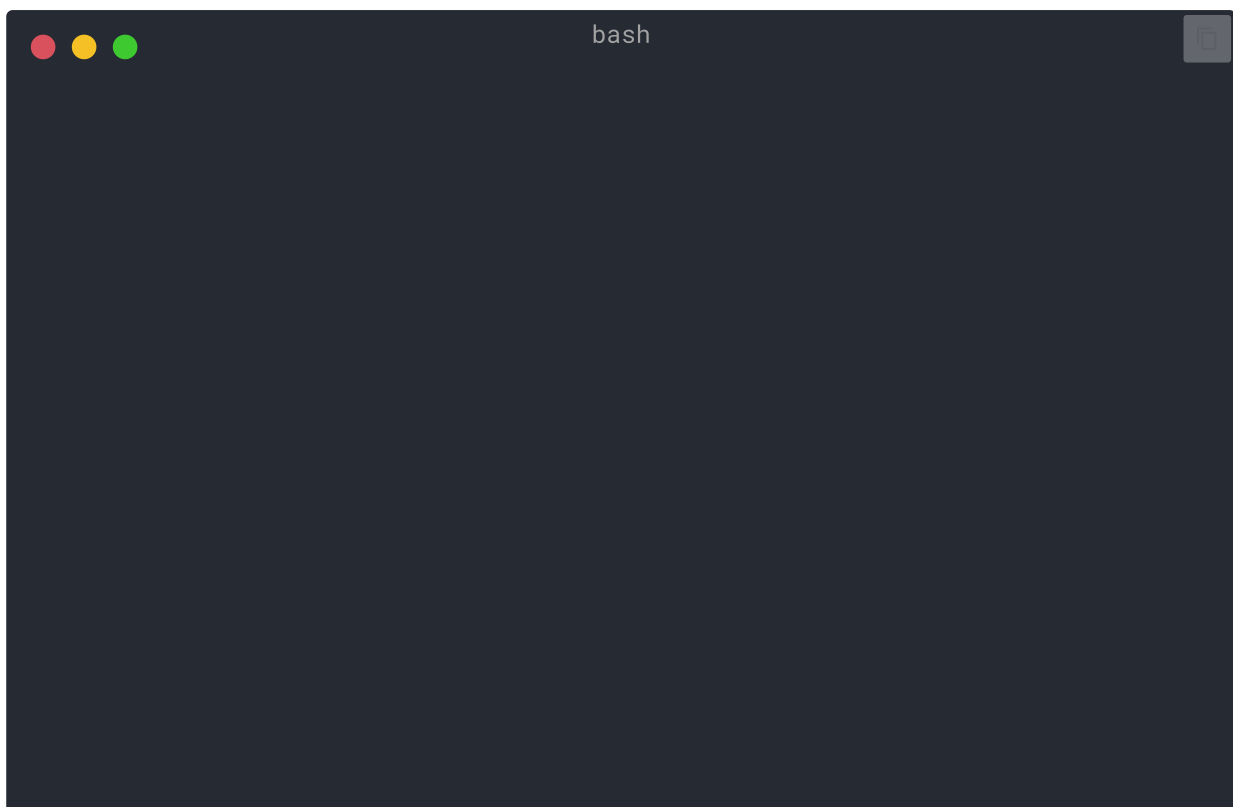
Tip

The second argument to `os.getenv()` [[↩](#)] is the default value to return.

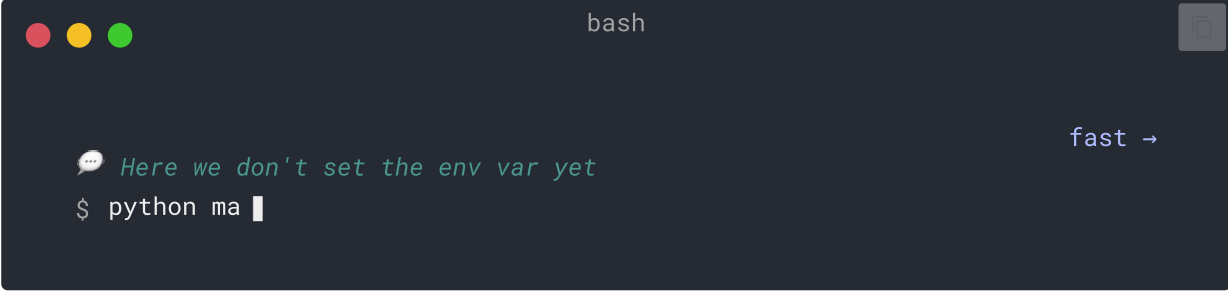
If not provided, it's `None` by default, here we provide `"World"` as the default value to use.

Then you could call that Python program:

Linux, macOS, Windows Bash



Windows PowerShell



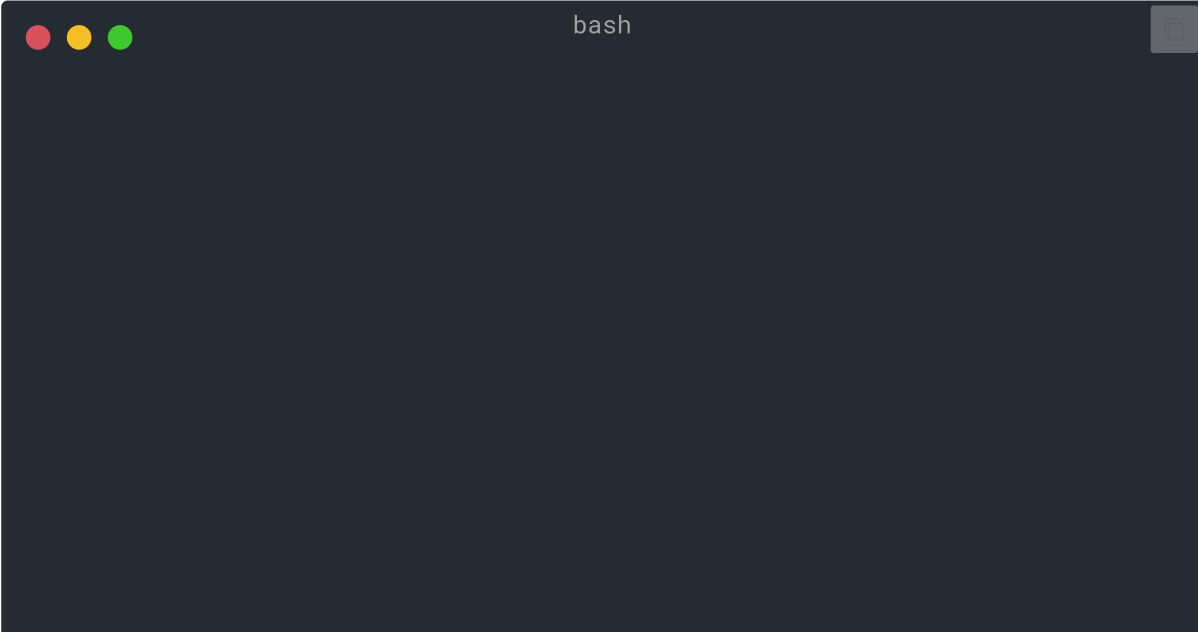
```
bash

Here we don't set the env var yet
$ python ma
```

As environment variables can be set outside of the code, but can be read by the code, and don't have to be stored (committed to `git`) with the rest of the files, it's common to use them for configurations or **settings**.

You can also create an environment variable only for a **specific program invocation**, that is only available to that program, and only for its duration.

To do that, create it right before the program itself, on the same line:



```
bash
```

Tip

You can read more about it at [The Twelve-Factor App: Config \[↔\]](#).

Types and Validation

These environment variables can only handle **text strings**, as they are external to Python and have to be compatible with other programs and the rest of the system (and even with different operating systems, as Linux, Windows, macOS).

That means that **any value** read in Python from an environment variable **will be a `str`**, and any conversion to a different type or any validation has to be done in code.

You will learn more about using environment variables for handling **application settings** in the [Advanced User Guide - Settings and Environment Variables ↩](#).

PATH Environment Variable

There is a **special** environment variable called **PATH** that is used by the operating systems (Linux, macOS, Windows) to find programs to run.

The value of the variable **PATH** is a long string that is made of directories separated by a colon `:` on Linux and macOS, and by a semicolon `;` on Windows.

For example, the **PATH** environment variable could look like this:

Linux, macOS

```
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
```

This means that the system should look for programs in the directories:

- `/usr/local/bin`
- `/usr/bin`
- `/bin`
- `/usr/sbin`
- `/sbin`

Windows

```
C:\Program Files\Python312\Scripts;C:\Program Files\Python312;C:\Windows\System32
```

This means that the system should look for programs in the directories:

- `C:\Program Files\Python312\Scripts`
- `C:\Program Files\Python312`
- `C:\Windows\System32`

When you type a **command** in the terminal, the operating system **looks for** the program in **each of those directories** listed in the **PATH** environment variable.

For example, when you type `python` in the terminal, the operating system looks for a program called `python` in the **first directory** in that list.

If it finds it, then it will **use it**. Otherwise it keeps looking in the **other directories**.

Installing Python and Updating the PATH

When you install Python, you might be asked if you want to update the PATH environment variable.

Linux, macOS

Let's say you install Python and it ends up in a directory `/opt/custompython/bin`.

If you say yes to update the PATH environment variable, then the installer will add `/opt/custompython/bin` to the PATH environment variable.

It could look like this:

```
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/custompython/bin
```

This way, when you type `python` in the terminal, the system will find the Python program in `/opt/custompython/bin` (the last directory) and use that one.

Windows

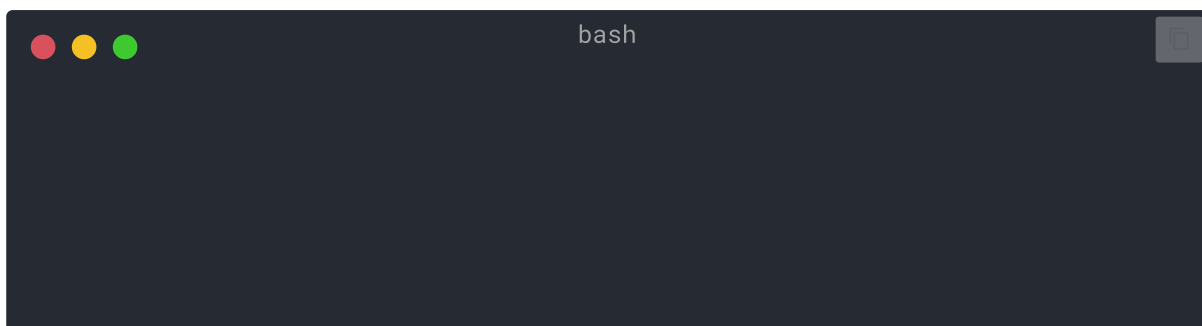
Let's say you install Python and it ends up in a directory `C:\opt\custompython\bin`.

If you say yes to update the PATH environment variable, then the installer will add `C:\opt\custompython\bin` to the PATH environment variable.

```
C:\Program Files\Python312\Scripts;C:\Program  
Files\Python312;C:\Windows\System32;C:\opt\custompython\bin
```

This way, when you type `python` in the terminal, the system will find the Python program in `C:\opt\custompython\bin` (the last directory) and use that one.

So, if you type:

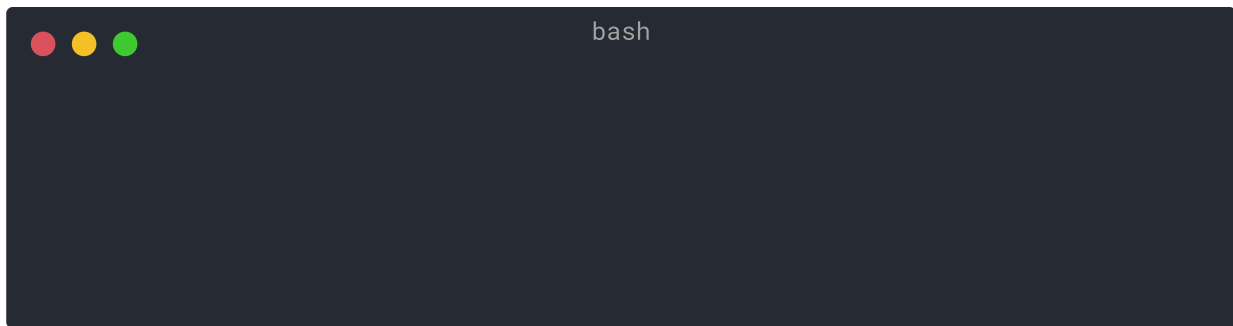


Linux, macOS

The system will **find** the `python` program in `/opt/custompython/bin` and run it.

It would be roughly equivalent to typing:

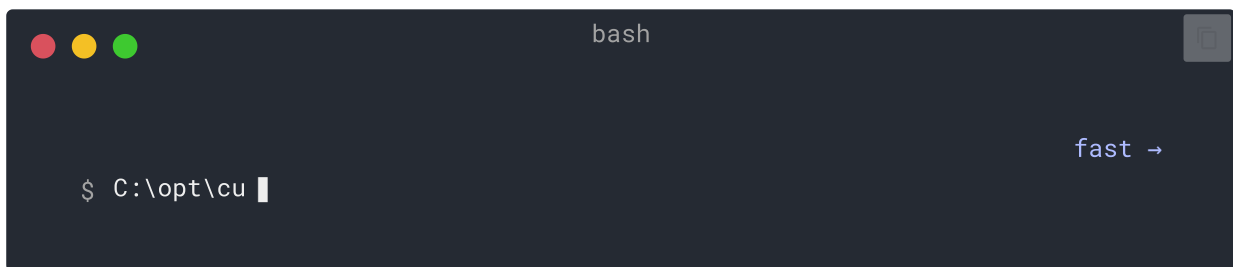




Windows

The system will **find** the `python` program in `C:\opt\custompython\bin\python` and run it.

It would be roughly equivalent to typing:



This information will be useful when learning about [Virtual Environments ↩](#).

Conclusion

With this you should have a basic understanding of what **environment variables** are and how to use them in Python.

You can also read more about them in the [Wikipedia for Environment Variable \[↩\]](#).

In many cases it's not very obvious how environment variables would be useful and applicable right away. But they keep showing up in many different scenarios when you are developing, so it's good to know about them.

For example, you will need this information in the next section, about [Virtual Environments](#).

Was this page helpful?

