

Deck 002

Python – Source Files

Dr. Hazel “[twitch.tv/hazeldotzone](https://www.twitch.tv/hazeldotzone)” Campbell

Copyright 2026, Dr. Hazel Victoria Campbell, All Rights Reserved

What is a Code Editor?

Essentially a word processor for computer program code...

Feel free to continue using your preferred editor if you already have one!

Installing Codium

<https://vscodium.com/>

VSCodium is an open-source version of VS Code with the AI disabled by default. Any online help relevant to VS Code usually applies to Codium too.



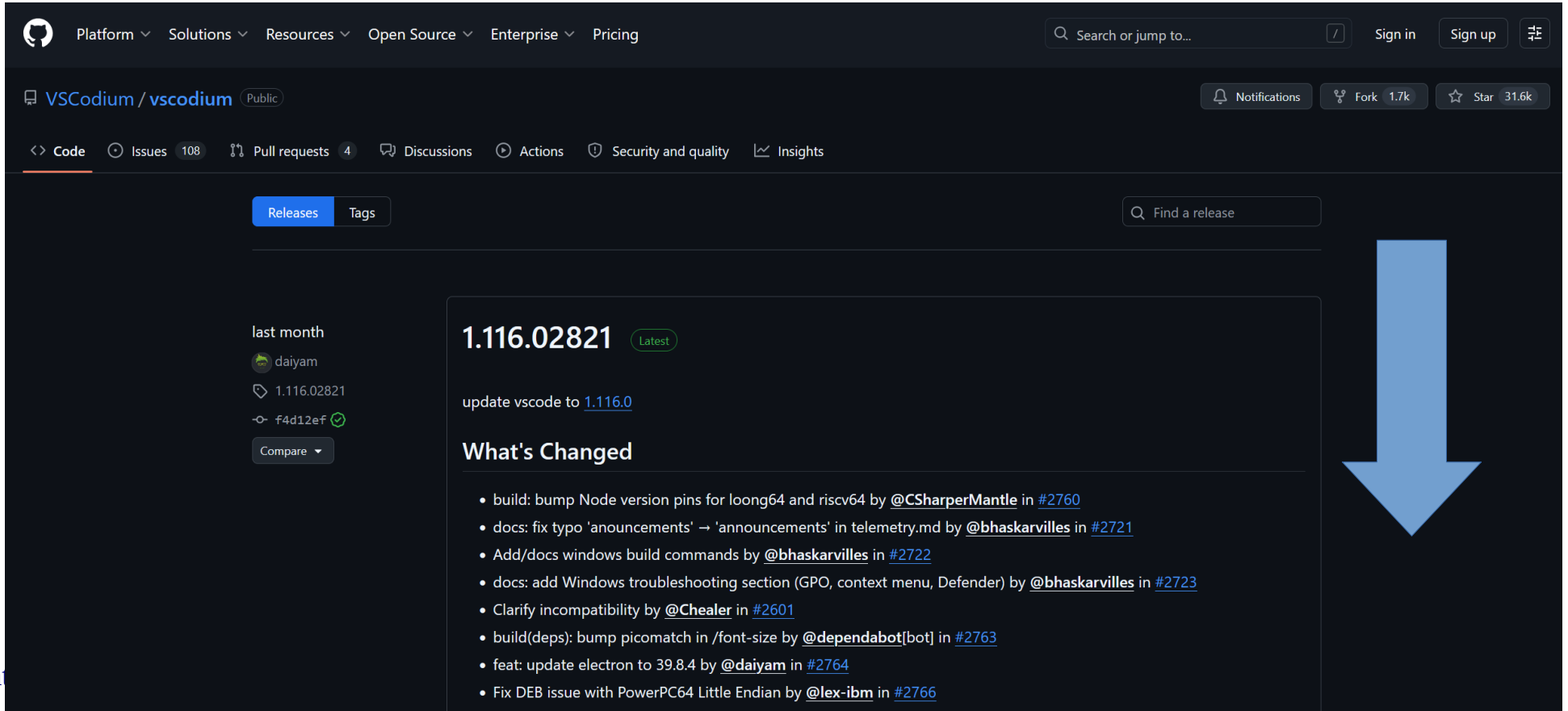
Download latest release

Available for Windows, Mac OS and Linux

Feel free to continue using your preferred editor if you already have one!

Installing Codium

<https://github.com/VSCodium/vscodium/releases>



The screenshot displays the GitHub interface for the VSCodium/vscodium repository. The top navigation bar includes links for Platform, Solutions, Resources, Open Source, Enterprise, and Pricing. A search bar is present in the top right. The repository name 'VSCodium / vscodium' is shown with a 'Public' label. Below this, there are buttons for Notifications, Fork (1.7k), and Star (31.6k). The main navigation bar includes links for Code, Issues (108), Pull requests (4), Discussions, Actions, Security and quality, and Insights. The 'Releases' tab is selected, and a search bar for releases is visible. The release details for version 1.116.02821 (marked as 'Latest') are shown, including the update instruction 'update vscode to 1.116.0' and a 'What's Changed' section with a list of updates. A large blue arrow points downwards on the right side of the page.

Platform Solutions Resources Open Source Enterprise Pricing

Search or jump to...

Sign in Sign up

VSCodium / vscodium Public

Notifications Fork 1.7k Star 31.6k

Code Issues 108 Pull requests 4 Discussions Actions Security and quality Insights

Releases Tags Find a release

last month

daiyam

1.116.02821

f4d12ef

Compare

1.116.02821 Latest

update vscode to [1.116.0](#)

What's Changed

- build: bump Node version pins for loong64 and riscv64 by [@CSharperMantle](#) in [#2760](#)
- docs: fix typo 'anouncements' → 'announcements' in telemetry.md by [@bhaskarvilles](#) in [#2721](#)
- Add/docs windows build commands by [@bhaskarvilles](#) in [#2722](#)
- docs: add Windows troubleshooting section (GPO, context menu, Defender) by [@bhaskarvilles](#) in [#2723](#)
- Clarify incompatibility by [@Chealer](#) in [#2601](#)
- build(deps): bump picomatch in /font-size by [@dependabot\[bot\]](#) in [#2763](#)
- feat: update electron to 39.8.4 by [@daiyam](#) in [#2764](#)
- Fix DEB issue with PowerPC64 Little Endian by [@lex-ibm](#) in [#2766](#)

Feel free to continue using your preferred editor if you already have one!

Windows	User Installer	VSCodiumUserSetup-x64-1.116.02821.exe
	System Installer	VSCodiumSetup-x64-1.116.02821.exe
	.zip	VSCodium-win32-x64-1.116.02821.zip
	.msi - updates enabled	VSCodium-x64-1.116.02821.msi
	.msi - updates disabled	VSCodium-x64-updates-disabled-1.116.02821.msi
	Remote Host	vscodium-reh-win32-x64-1.116.02821.tar.gz
	Web Host	vscodium-reh-web-win32-x64-1.116.02821.tar.gz
	CLI	vscodium-cli-win32-x64-1.116.02821.tar.gz
macOS	.dmg	VSCodium.x64.1.116.02821.dmg
	.zip	VSCodium-darwin-x64-1.116.02821.zip
	Remote Host	vscodium-reh-darwin-x64-1.116.02821..tar.gz
	Web Host	vscodium-reh-web-darwin-x64-1.116.02821.tar.gz
	CLI	vscodium-cli-darwin-x64-1.116.02821.tar.gz
Linux	.deb	codium_1.116.02821_amd64.deb
	.rpm	codium-1.116.02821-el8.x86_64.rpm
	.tar.gz	VSCodium-linux-x64-1.116.02821.tar.gz
	Applmage	VSCodium-1.116.02821.glibc2.30-x86_64.Applmage VSCodium-1.116.02821.glibc2.30-x86_64.Applmage.zsync
	Snap	codium_1.116.02821_amd64.snap
	Remote Host	vscodium-reh-linux-x64-1.116.02821.tar.gz
	Web Host	vscodium-reh-web-linux-x64-1.116.02821.tar.gz



I am choosing this one though many of the other options will also work.

Feel free to continue using your preferred editor if you already have one!

Windows	User Installer	VSCodiumUserSetup-x64-1.116.02821.exe
	System Installer	VSCodiumSetup-x64-1.116.02821.exe
	.zip	VSCodium-win32-x64-1.116.02821.zip
	.msi - updates enabled	VSCodium-x64-1.116.02821.msi
	.msi - updates disabled	VSCodium-x64-updates-disabled-1.116.02821.msi
	Remote Host	vscodium-reh-win32-x64-1.116.02821.tar.gz
	Web Host	vscodium-reh-web-win32-x64-1.116.02821.tar.gz
	CLI	vscodium-cli-win32-x64-1.116.02821.tar.gz
macOS	.dmg	VSCodium.x64.1.116.02821.dmg
	.zip	VSCodium-darwin-x64-1.116.02821.zip
	Remote Host	vscodium-reh-darwin-x64-1.116.02821..tar.gz
	Web Host	vscodium-reh-web-darwin-x64-1.116.02821.tar.gz
	CLI	vscodium-cli-darwin-x64-1.116.02821.tar.gz
Linux	.deb	codium_1.116.02821_amd64.deb
	.rpm	codium-1.116.02821-el8.x86_64.rpm
	.tar.gz	VSCodium-linux-x64-1.116.02821.tar.gz
	Applmage	VSCodium-1.116.02821.glibc2.30-x86_64.Applmage VSCodium-1.116.02821.glibc2.30-x86_64.Applmage.zsync
	Snap	codium_1.116.02821_amd64.snap
	Remote Host	vscodium-reh-linux-x64-1.116.02821.tar.gz
	Web Host	vscodium-reh-web-linux-x64-1.116.02821.tar.gz



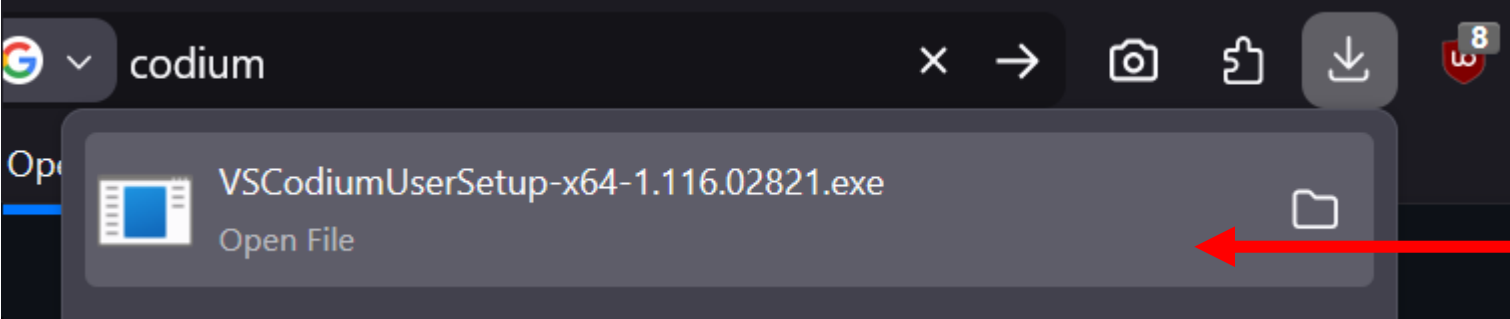
I am choosing this one though many of the other options will also work.

Don't worry too much about options you don't understand. I don't know what half of these, such as "reh" are either... sometimes it's best to just skip over some things, rather than try to understand every option in front of you.

Feel free to continue using your preferred editor if you already have one!

Installing Codium

<https://github.com/VSCodium/vscodium/releases>

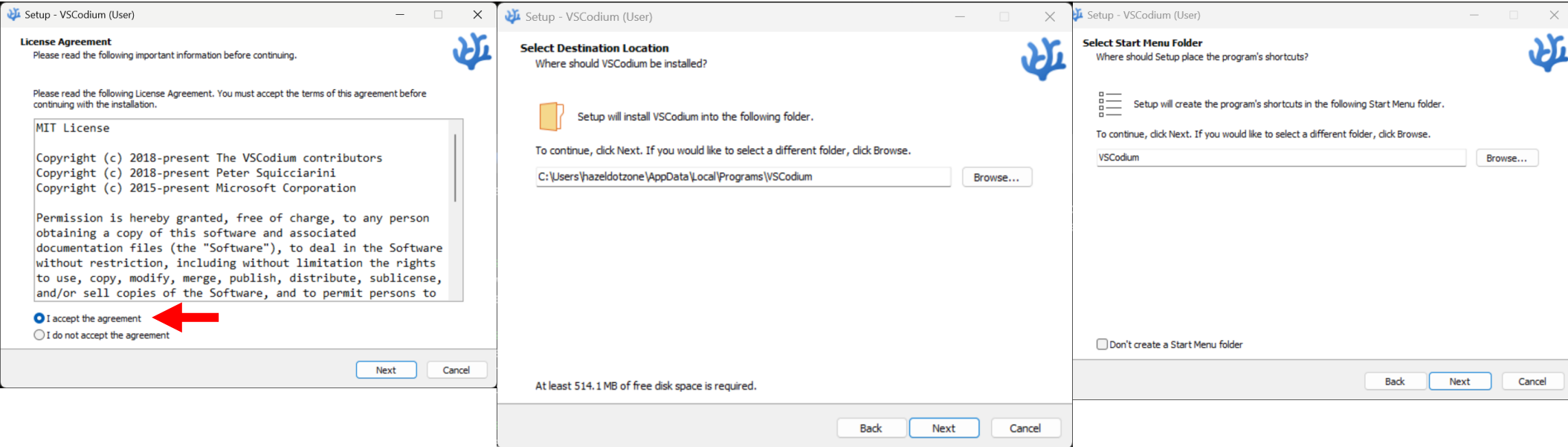


Run the installer...

Feel free to continue using your preferred editor if you already have one!

Installing Codium

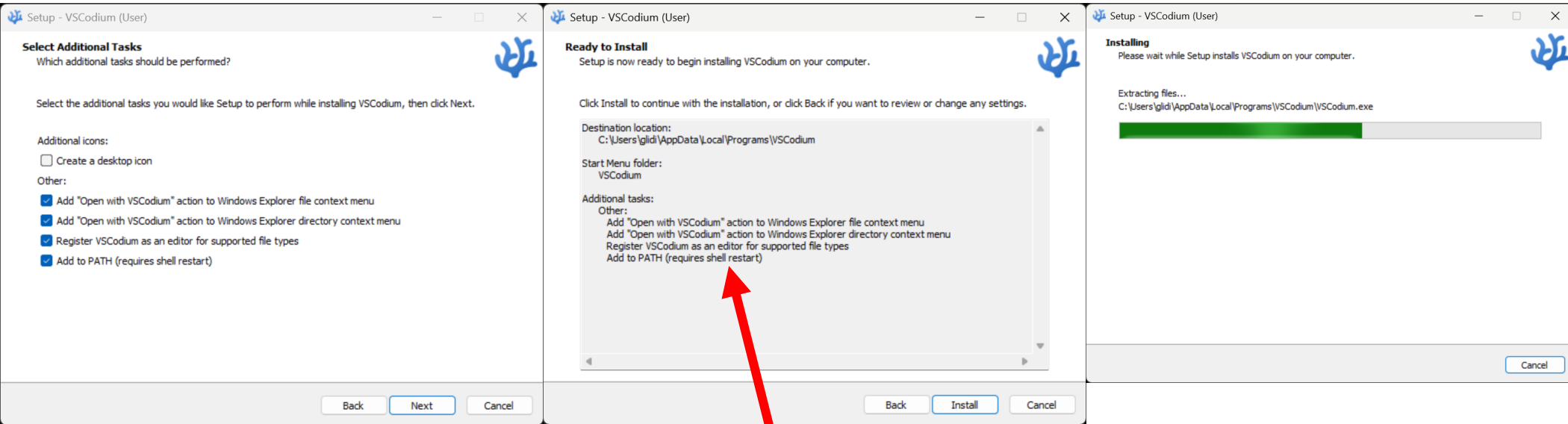
<https://github.com/VSCodium/vscodium/releases>



Feel free to continue using your preferred editor if you already have one!

Installing Codium

<https://github.com/VSCodium/vscodium/releases>

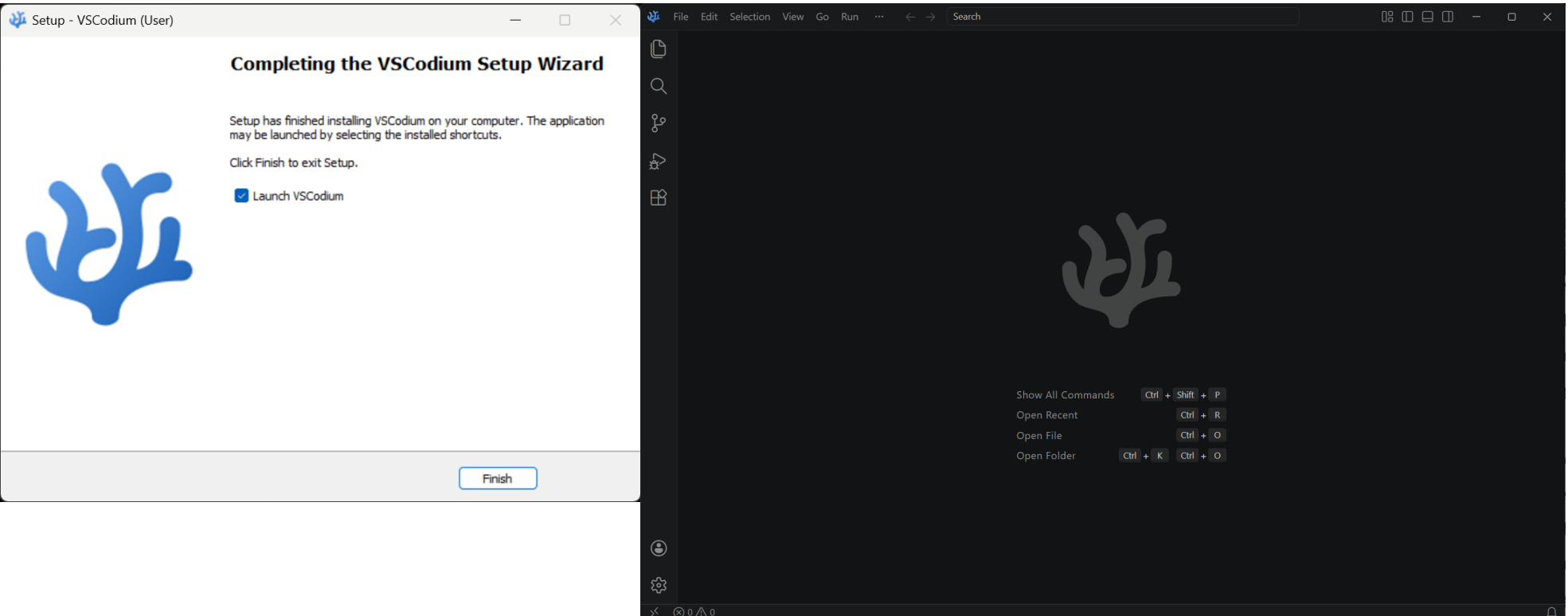


We will need to reboot or logout and log back in.

Feel free to continue using your preferred editor if you already have one!

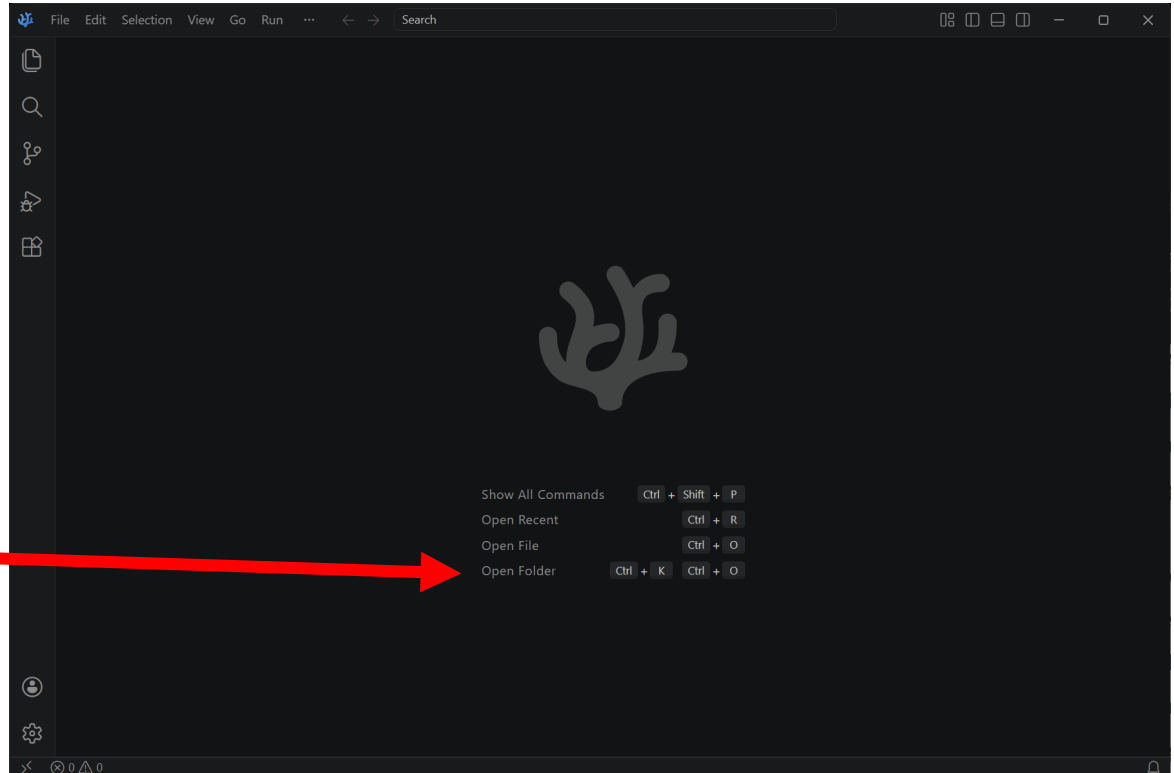
Installing Codium

<https://github.com/VSCodium/vscodium/releases>

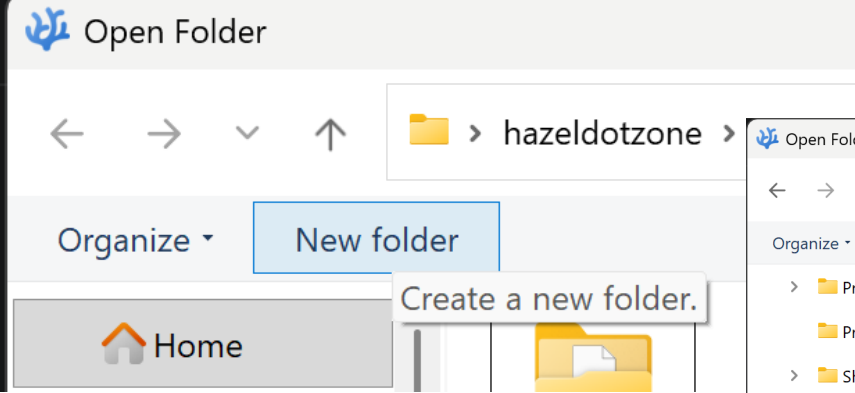


Using Codium

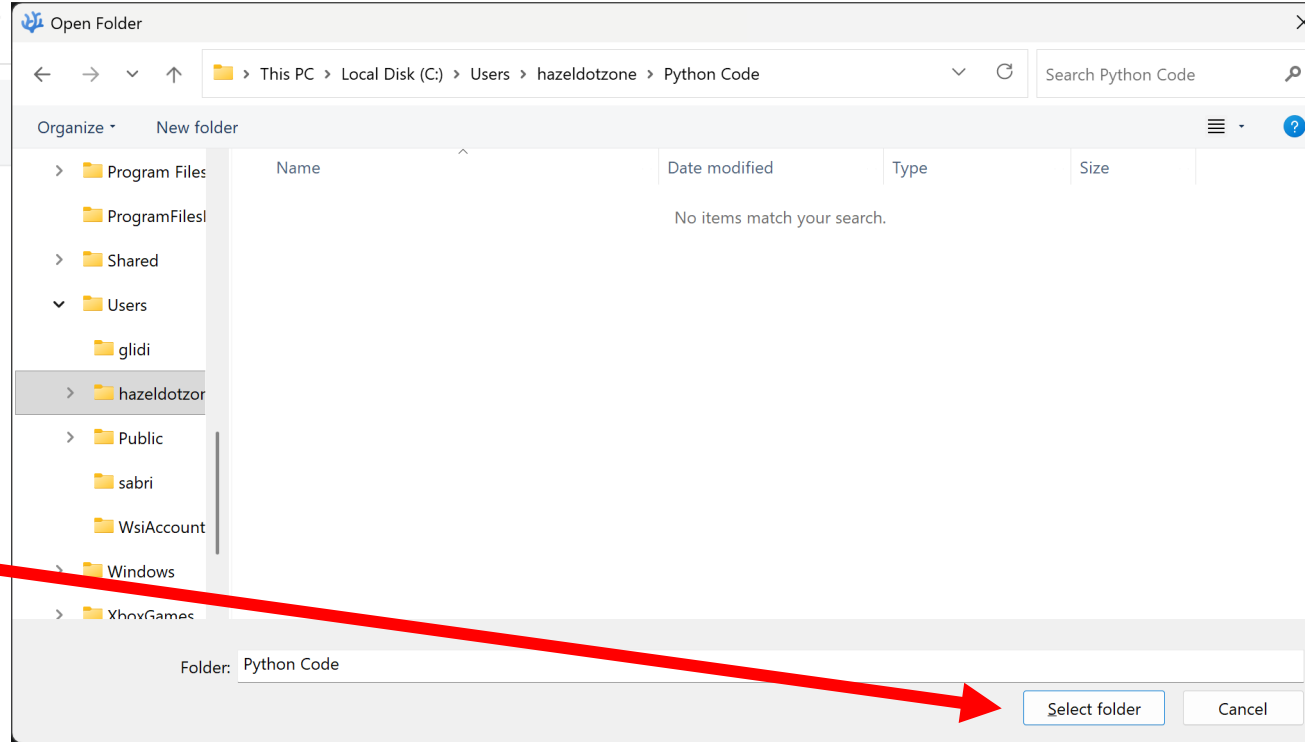
Usually projects are kept in folders, and we can open the entire folder at once in our editor... but we don't have a folder yet.



Using Codium



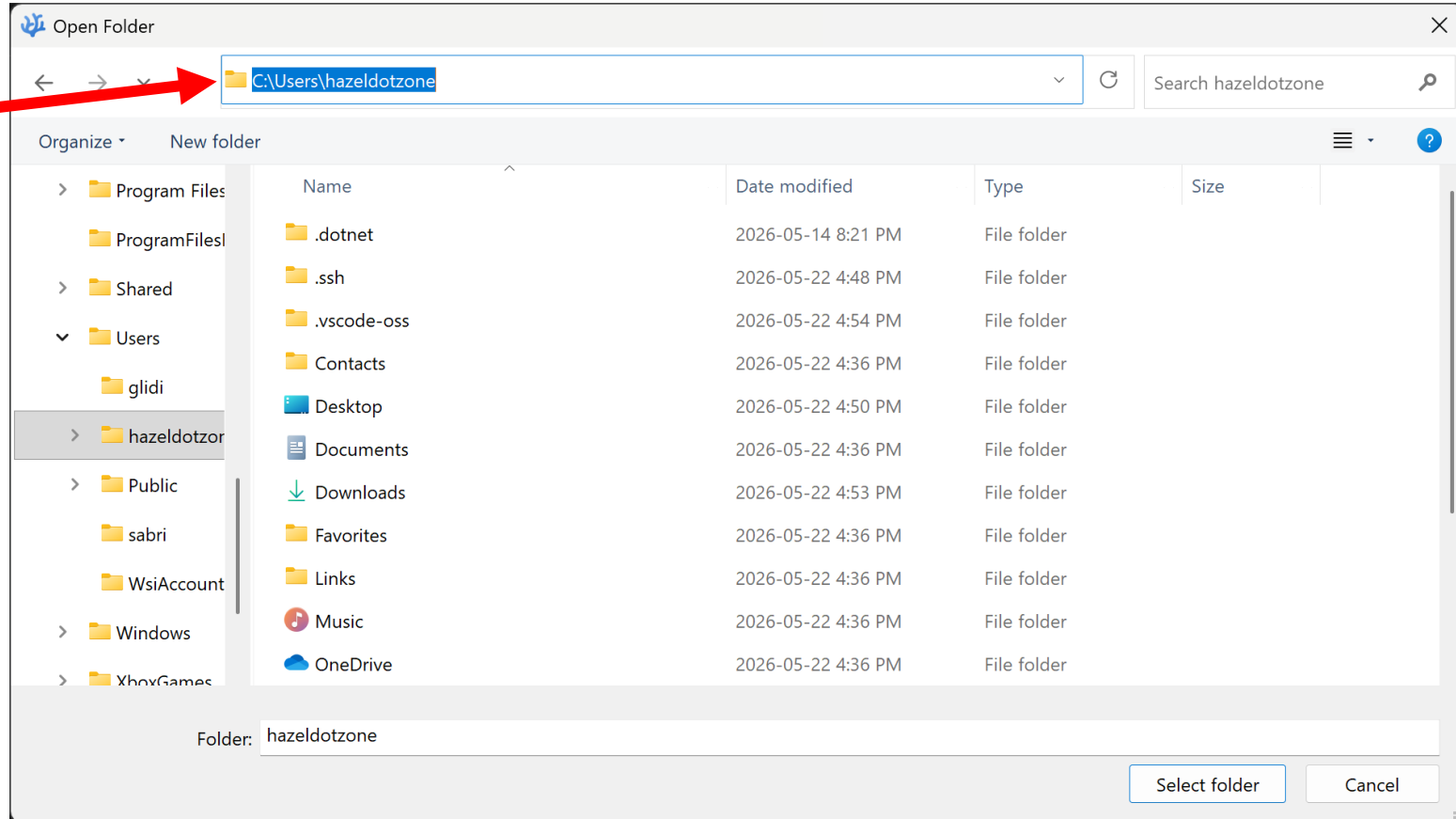
Create a new folder, go into it and then hit select.



Aside: Home Folder

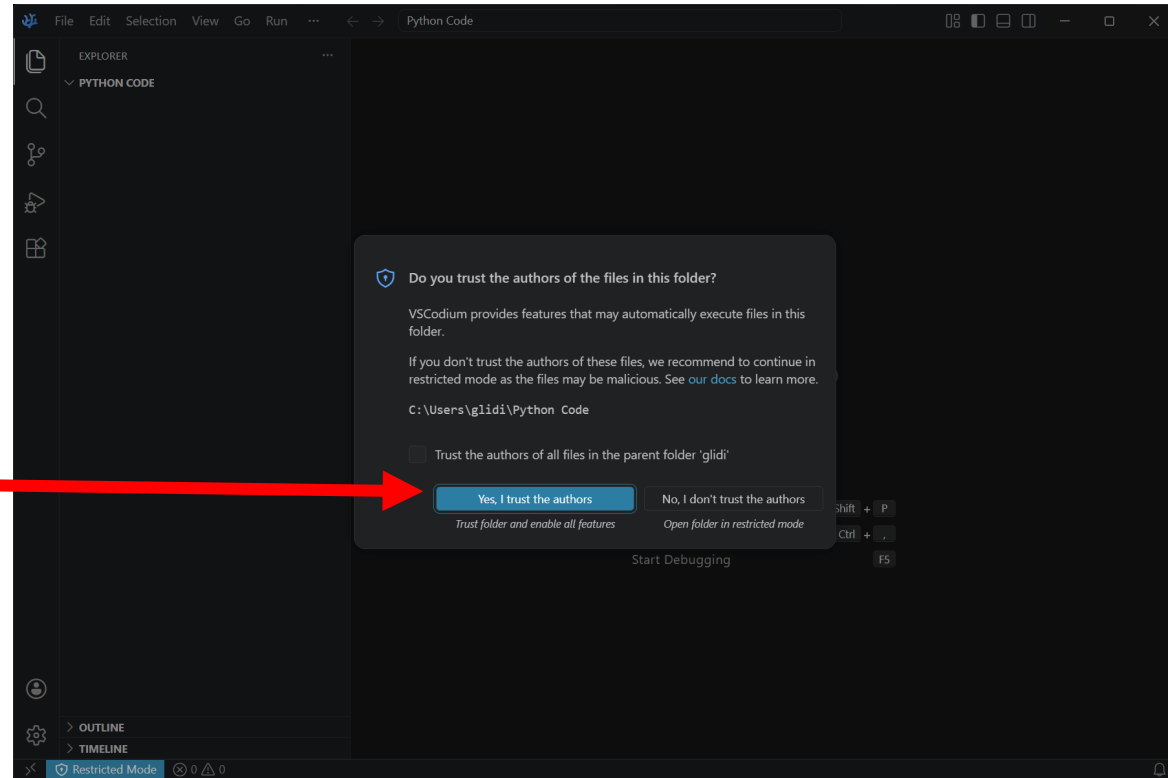
In windows your home folder is in C:\Users\
A red arrow points from the text to the address bar in the screenshot below.

This is where you can find a variety of files relevant to computer programming



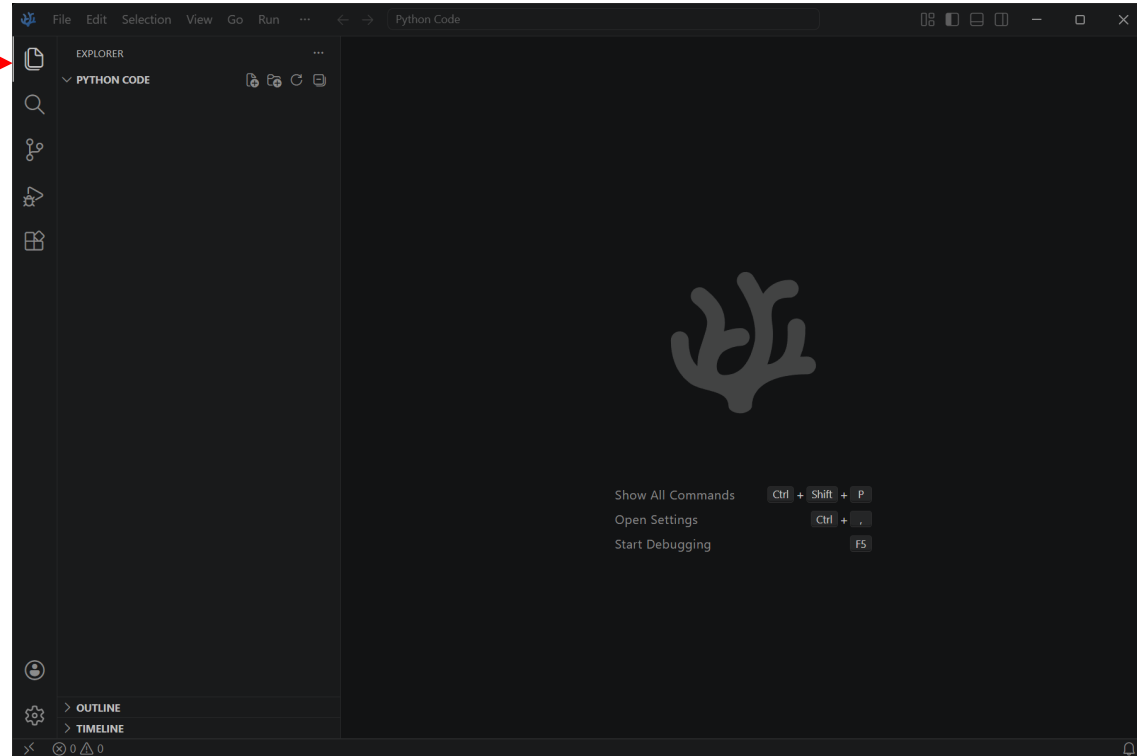
Using Codium

Click yes I trust the authors.



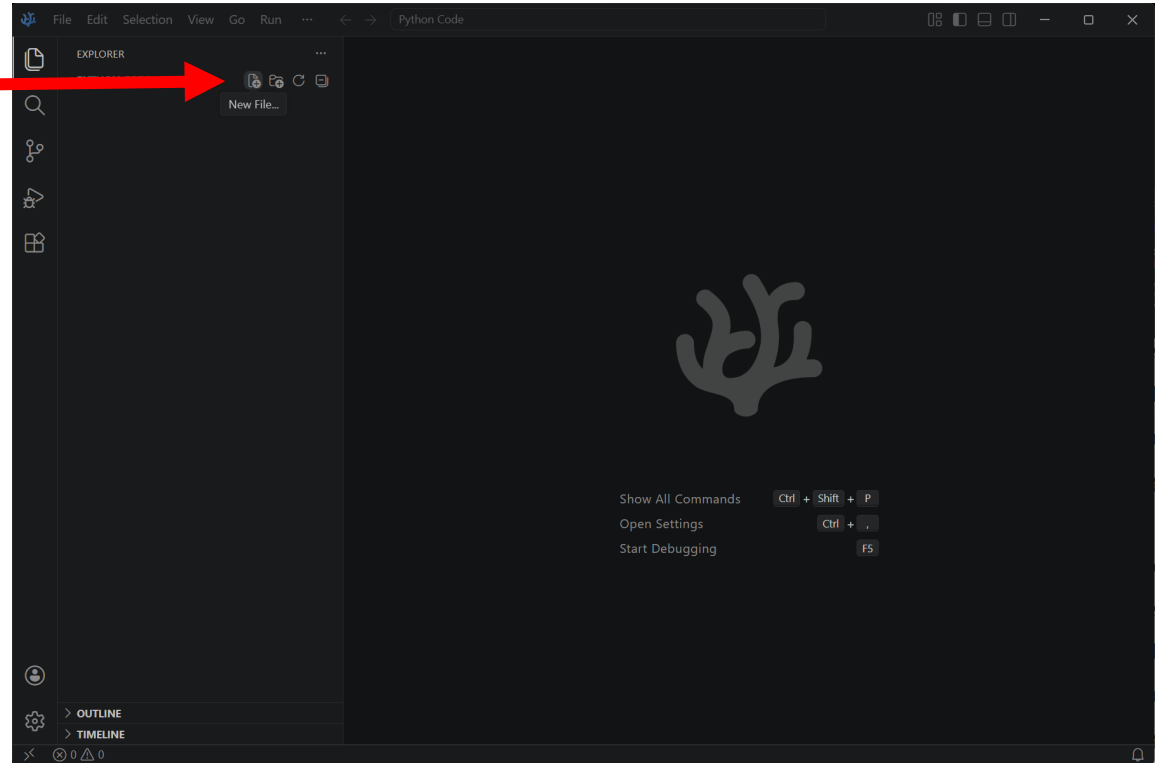
Using Codium

Explorer button...
We have an outline
view of our empty
folder.



Using Codium

Click here to make a new file

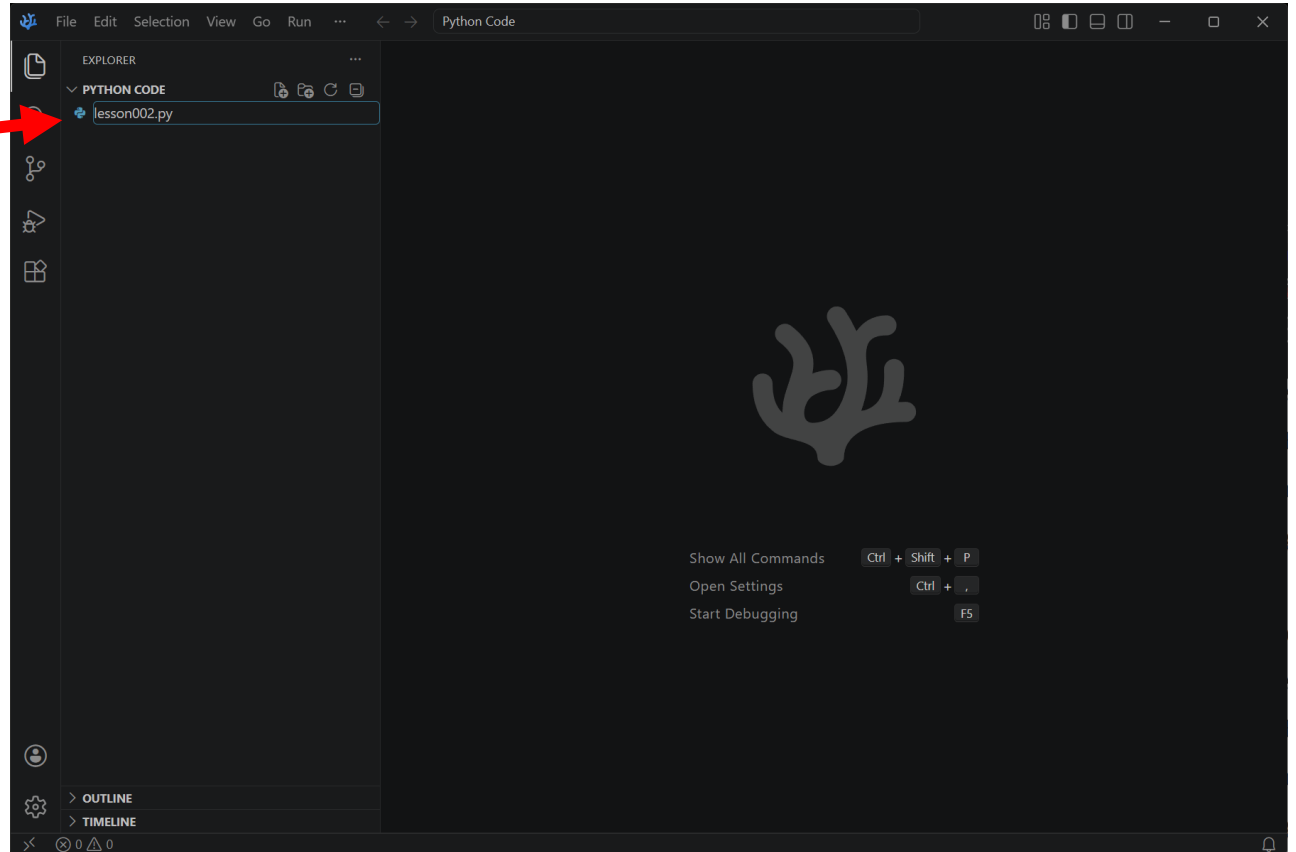
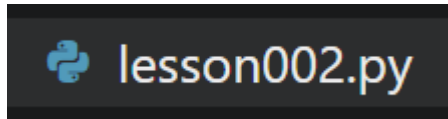


Using Codium

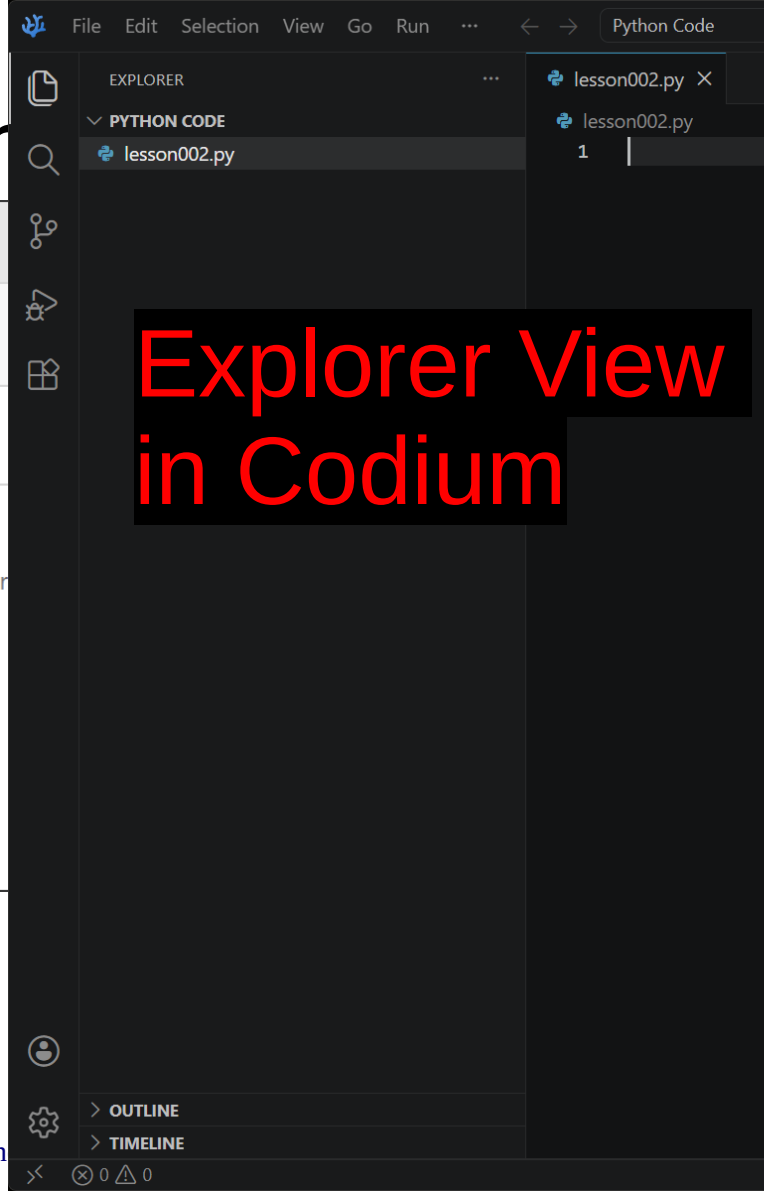
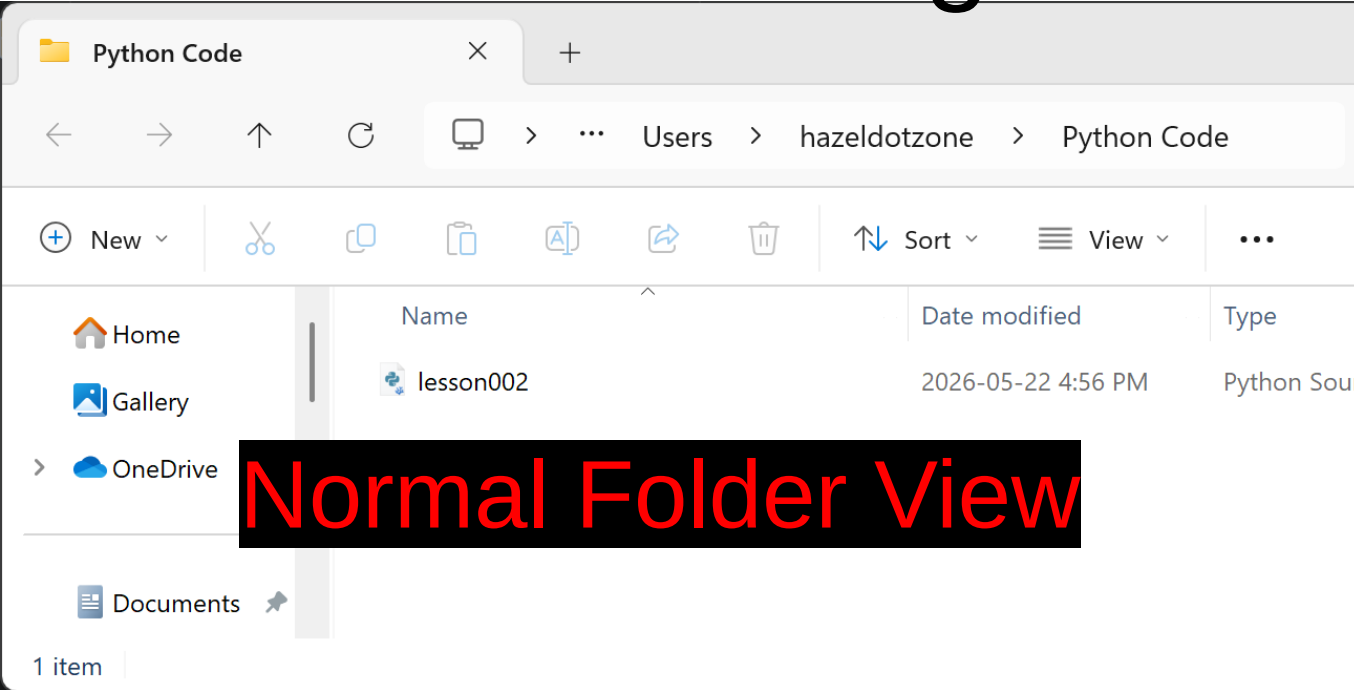
Click here to name the new file



If we want it to be recognized as a python code file we should make sure the name ends in .py then the icon will change to the python icon

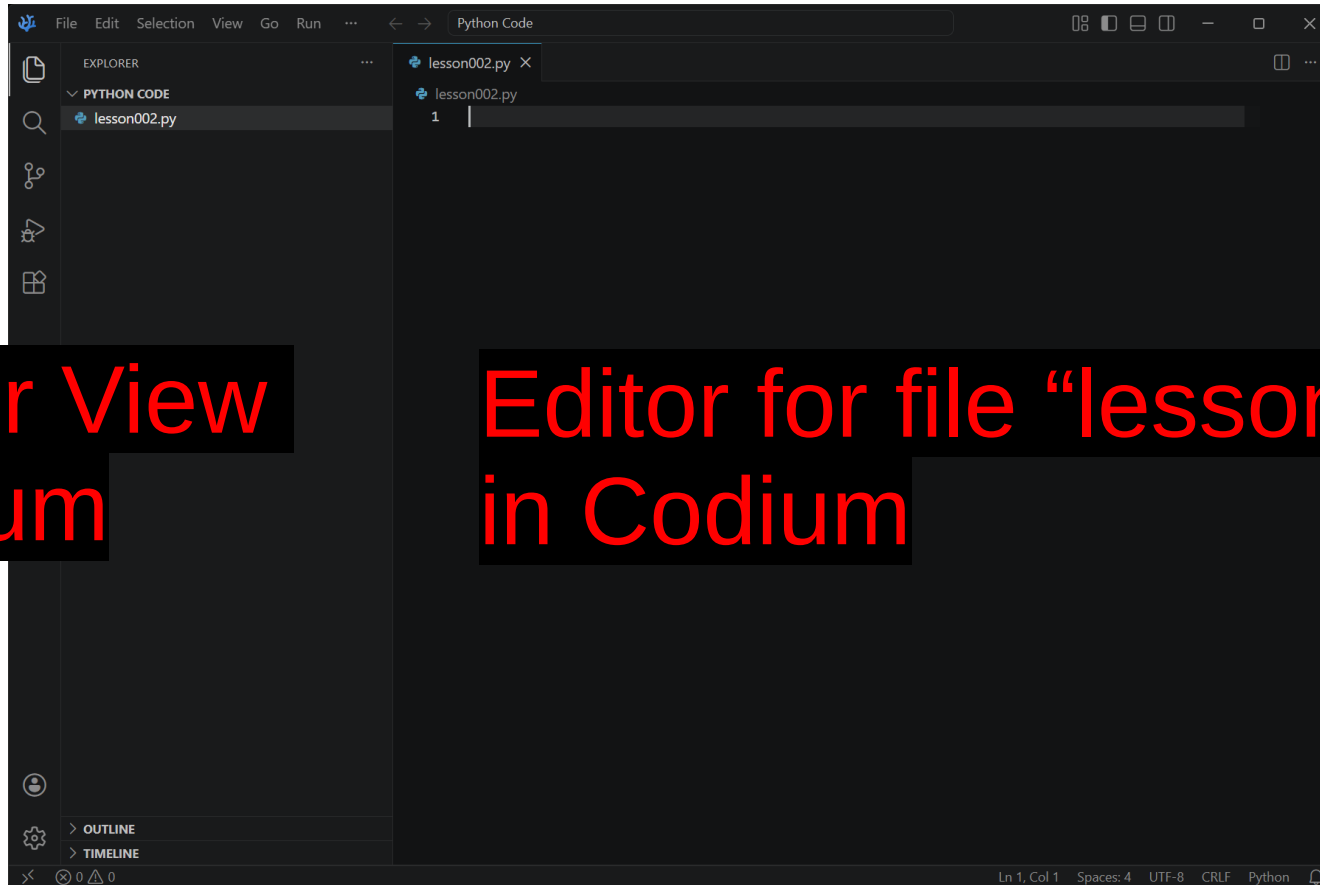


Using Codium



Explorer View
in Codium

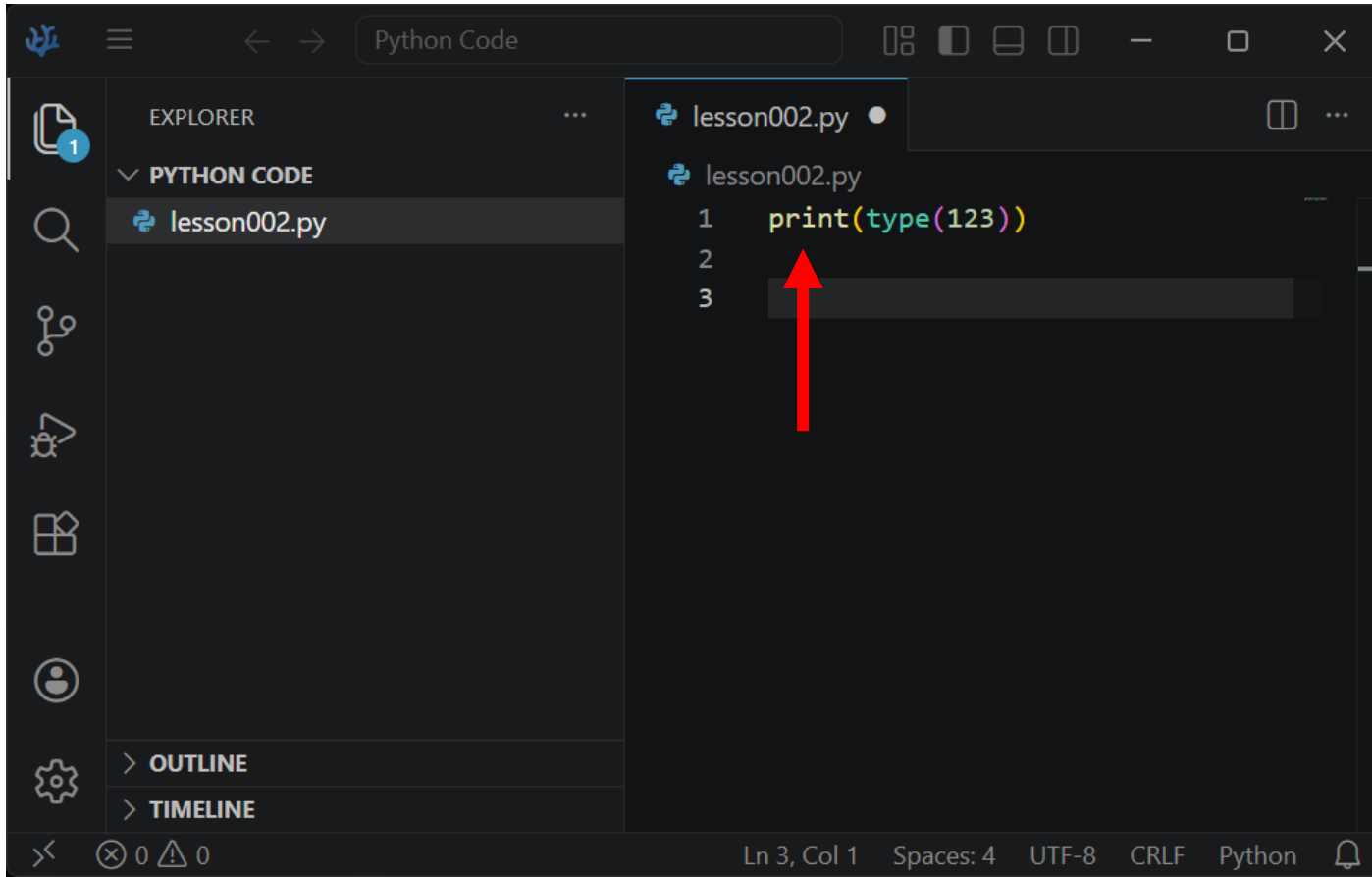
Using Codium



Explorer View
in Codium

Editor for file "lesson002.py"
in Codium

Using Codium



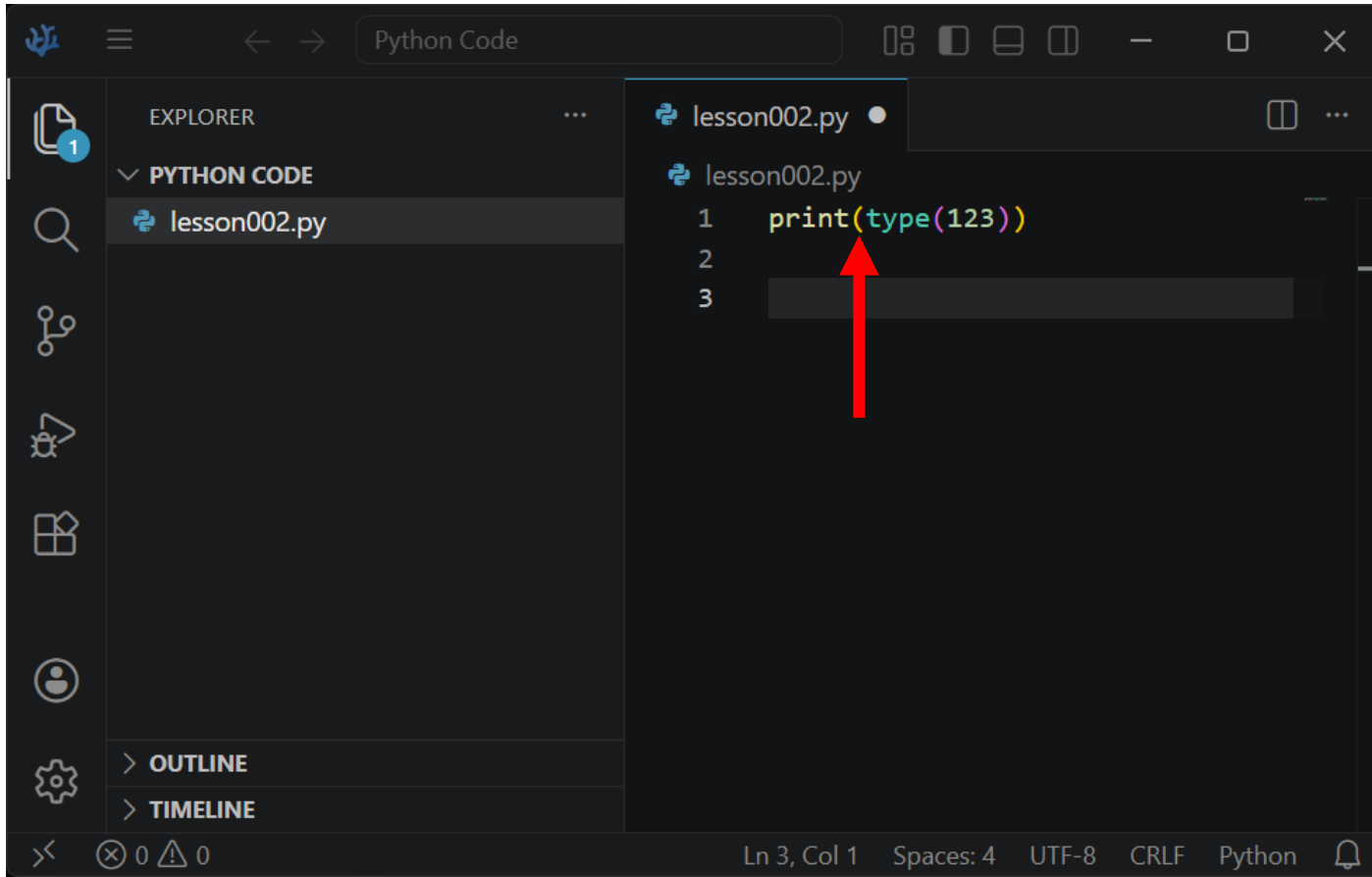
The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'PYTHON CODE' with a file 'lesson002.py'. The main editor window displays the content of 'lesson002.py' with the following code:

```
1 print(type(123))
2
3
```

The code is syntax-highlighted: 'print' is in blue, 'type' is in green, and '123' is in yellow. A red arrow points to the opening parenthesis of the 'type' function call on line 1. The status bar at the bottom indicates 'Ln 3, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python'.

The colors help us visually distinguish elements (identifiers, delimiters, ...)

Using Codium



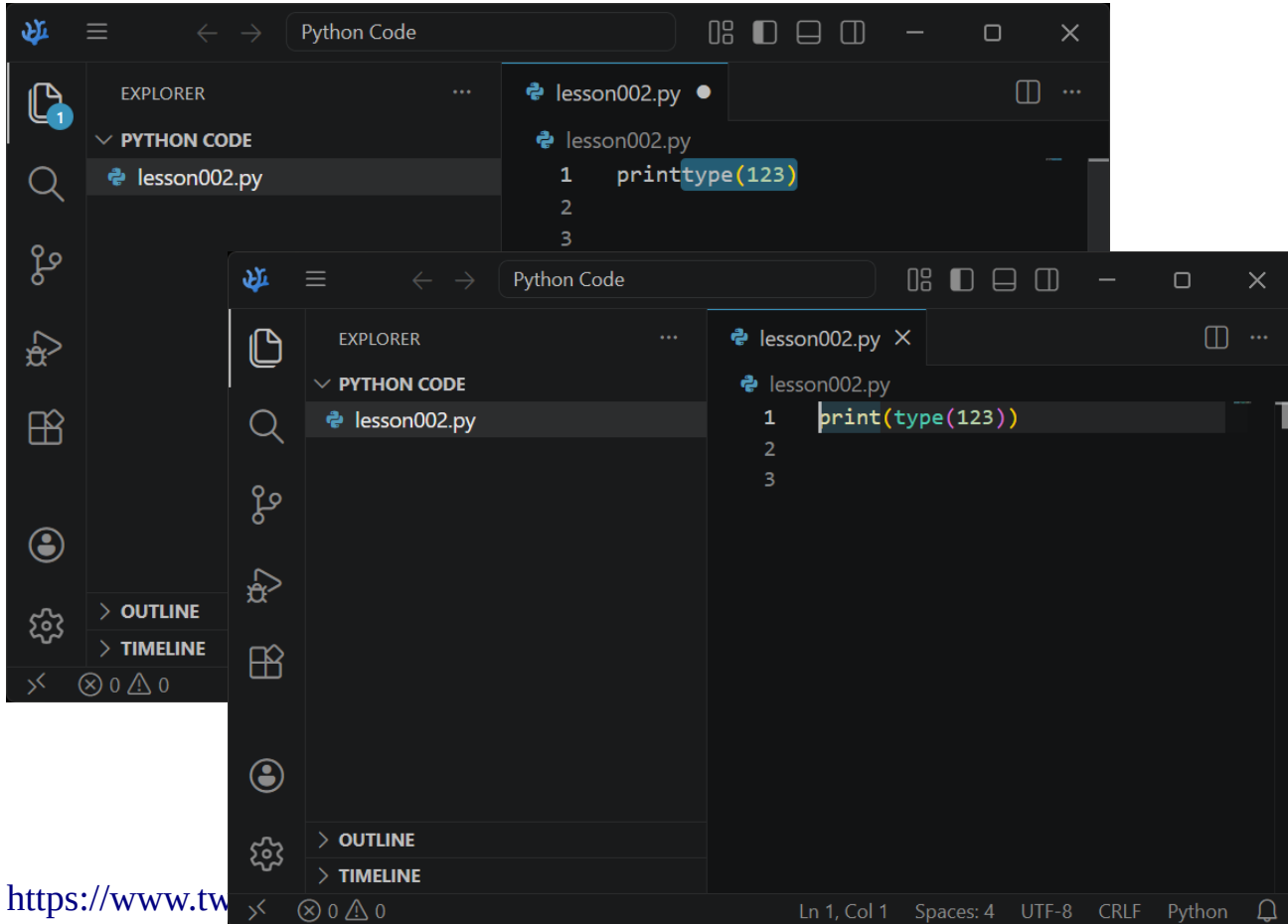
```
Python Code  
EXPLORER  
PYTHON CODE  
lesson002.py  
lesson002.py  
1 print(type(123))  
2  
3
```

Ln 3, Col 1 Spaces: 4 UTF-8 CRLF Python

Typing (will place both parentheses in some situations...

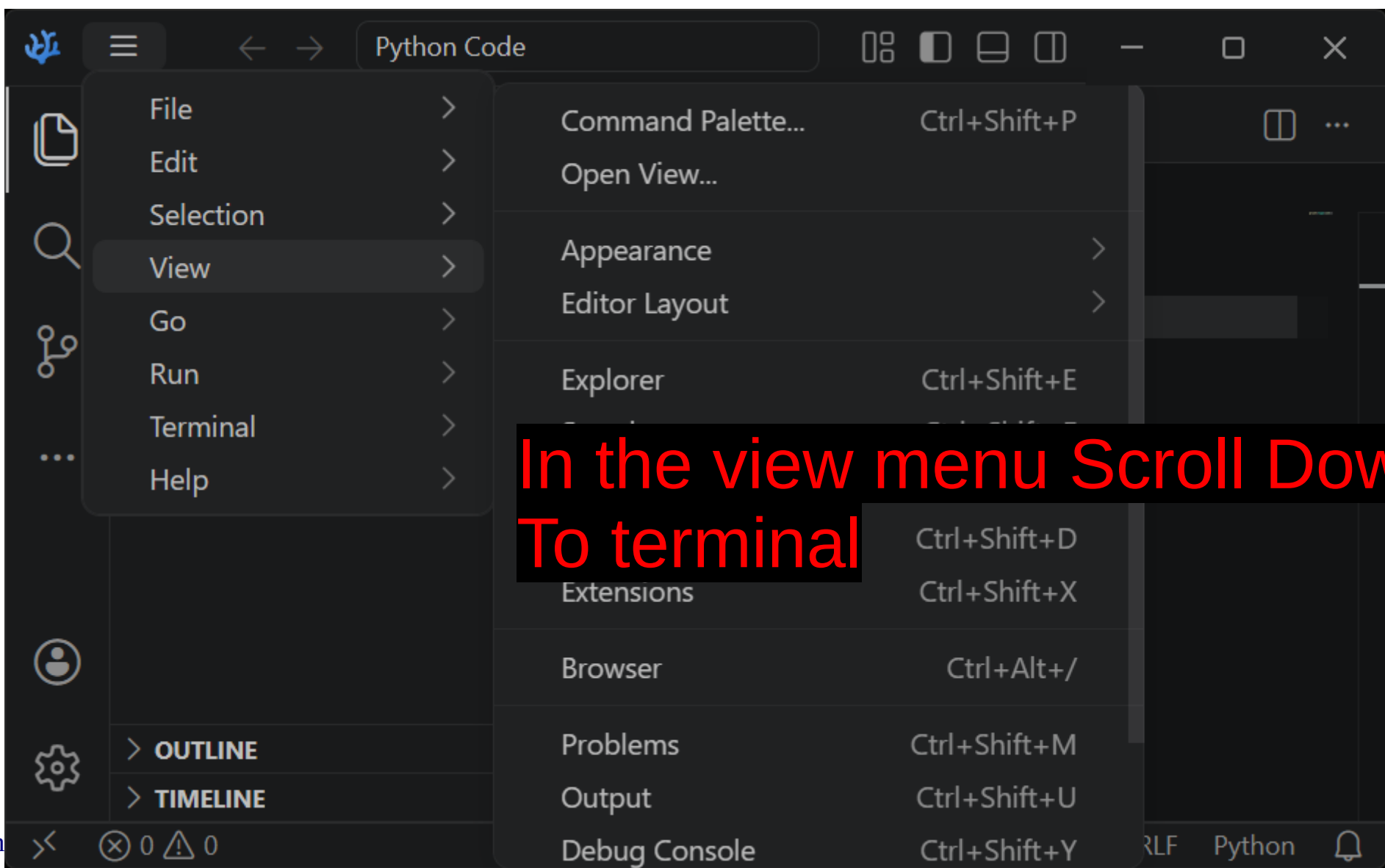
Try highlighting some text and hitting the (key

Using Codium

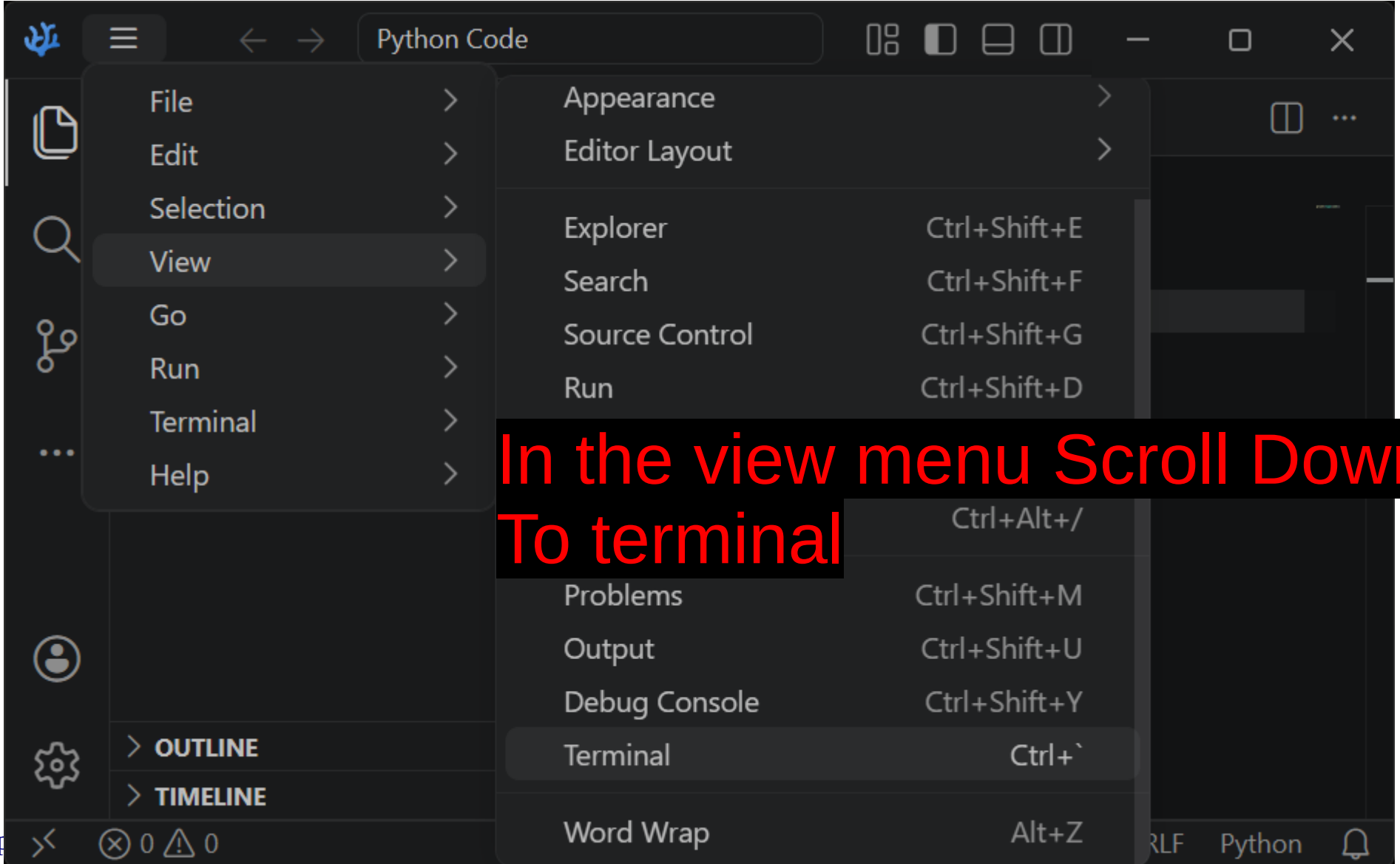


Typing (will place both parentheses in some situations...

Try highlighting some text and hitting the (key



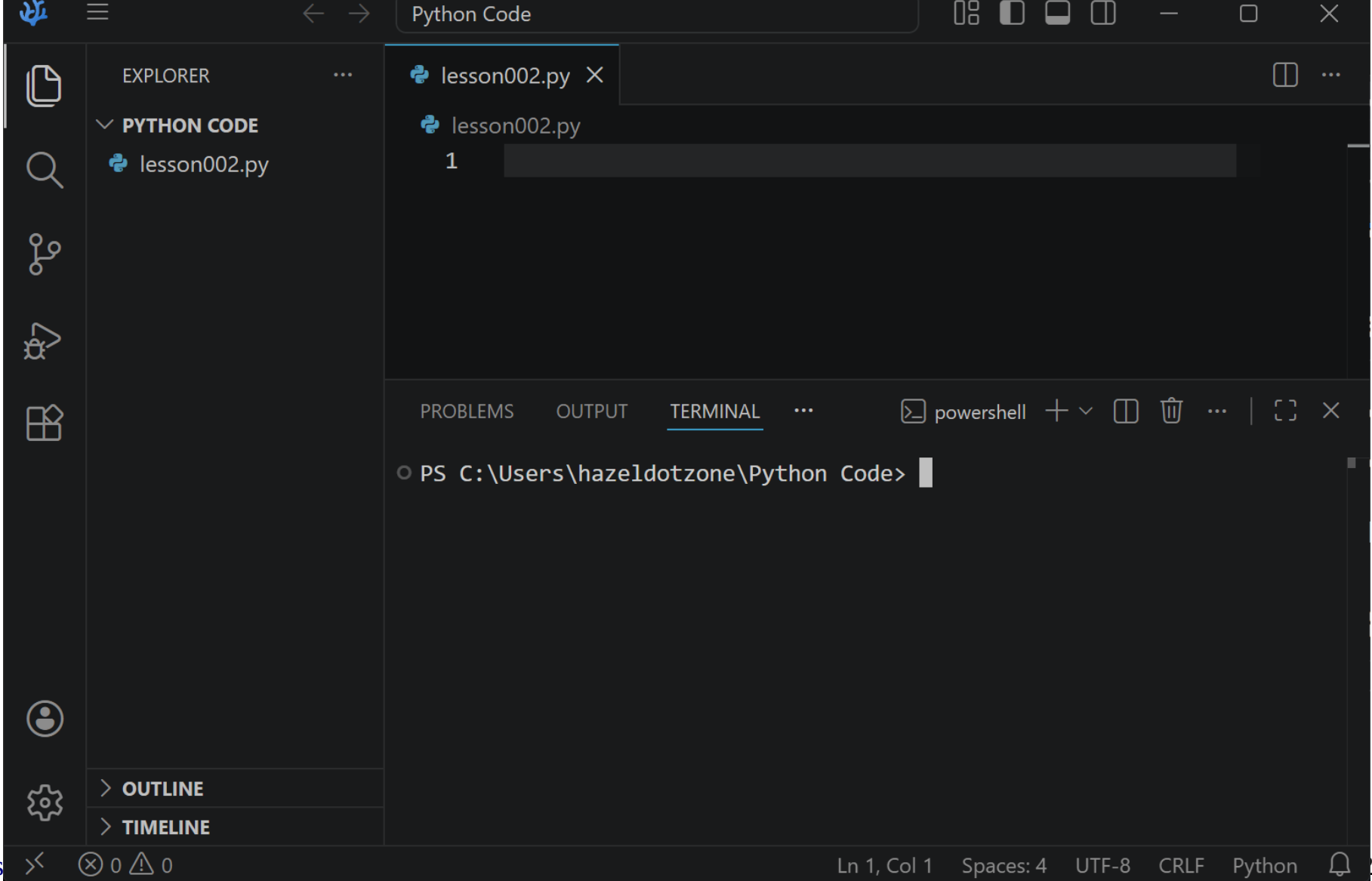
In the view menu Scroll Down To terminal

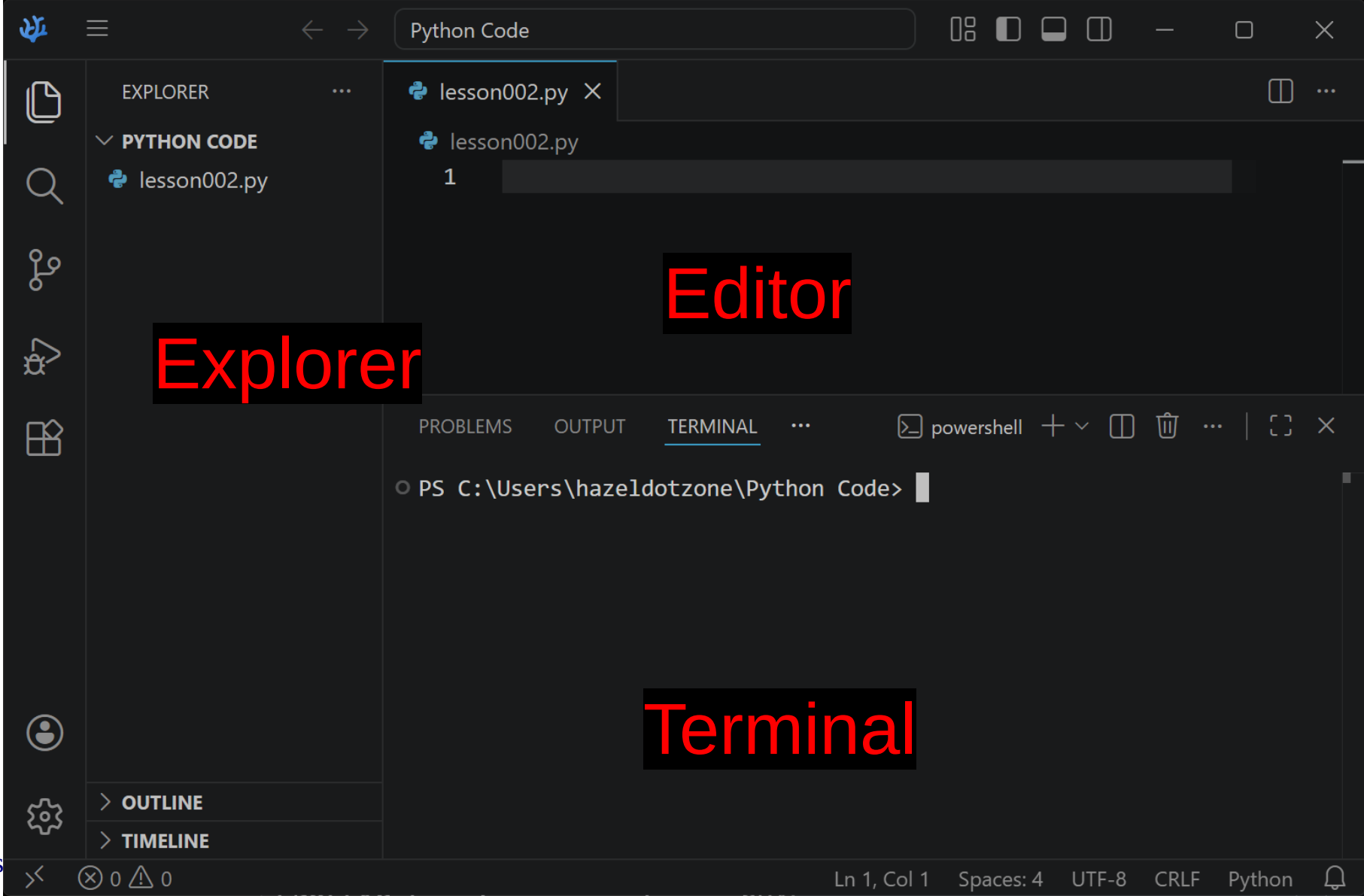


- File >
- Edit >
- Selection >
- View >
- Go >
- Run >
- Terminal >
- Help >

- Appearance >
- Editor Layout >
- Explorer Ctrl+Shift+E
- Search Ctrl+Shift+F
- Source Control Ctrl+Shift+G
- Run Ctrl+Shift+D
- Ctrl+Alt+/
- Problems Ctrl+Shift+M
- Output Ctrl+Shift+U
- Debug Console Ctrl+Shift+Y
- Terminal Ctrl+`
- Word Wrap Alt+Z

In the view menu Scroll Down To terminal

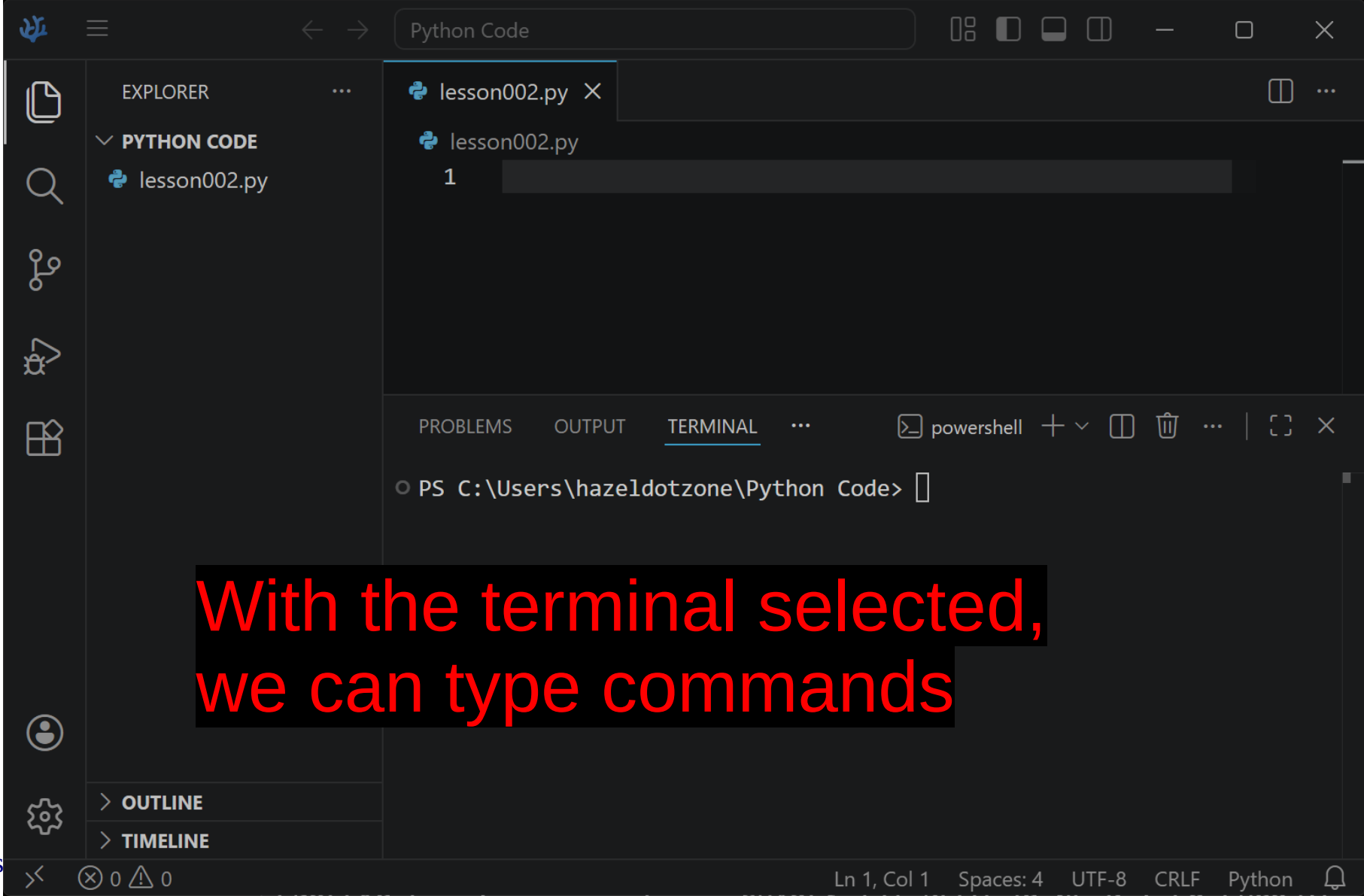




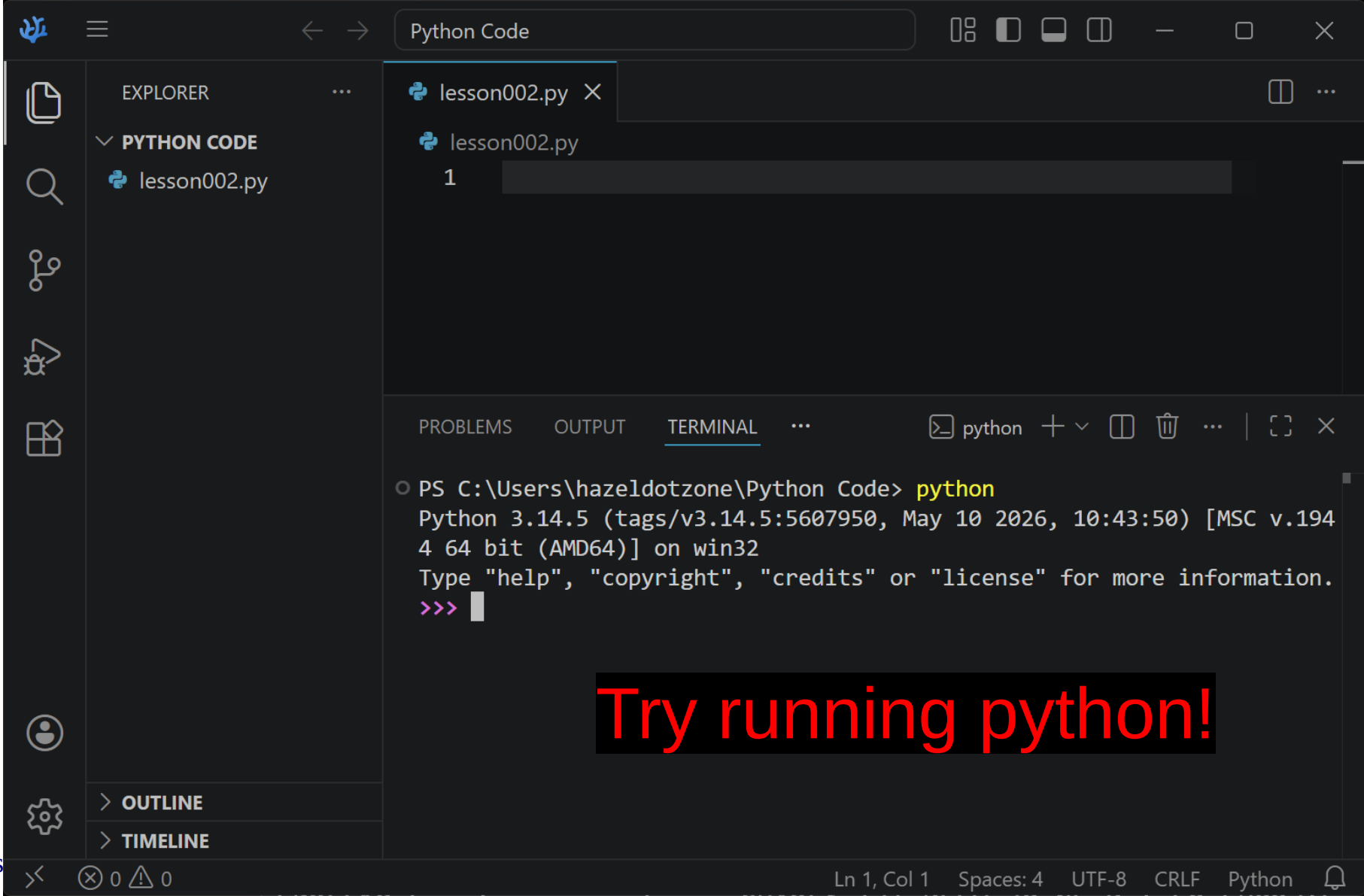
Explorer

Editor

Terminal



With the terminal selected,
we can type commands



Python Code

EXPLORER

PYTHON CODE

lesson002.py

lesson002.py

lesson002.py

1

PROBLEMS

OUTPUT

TERMINAL

python

PS C:\Users\hazeldotzone\Python Code> python

Python 3.14.5 (tags/v3.14.5:5607950, May 10 2026, 10:43:50) [MSC v.1944 64 bit (AMD64)] on win32

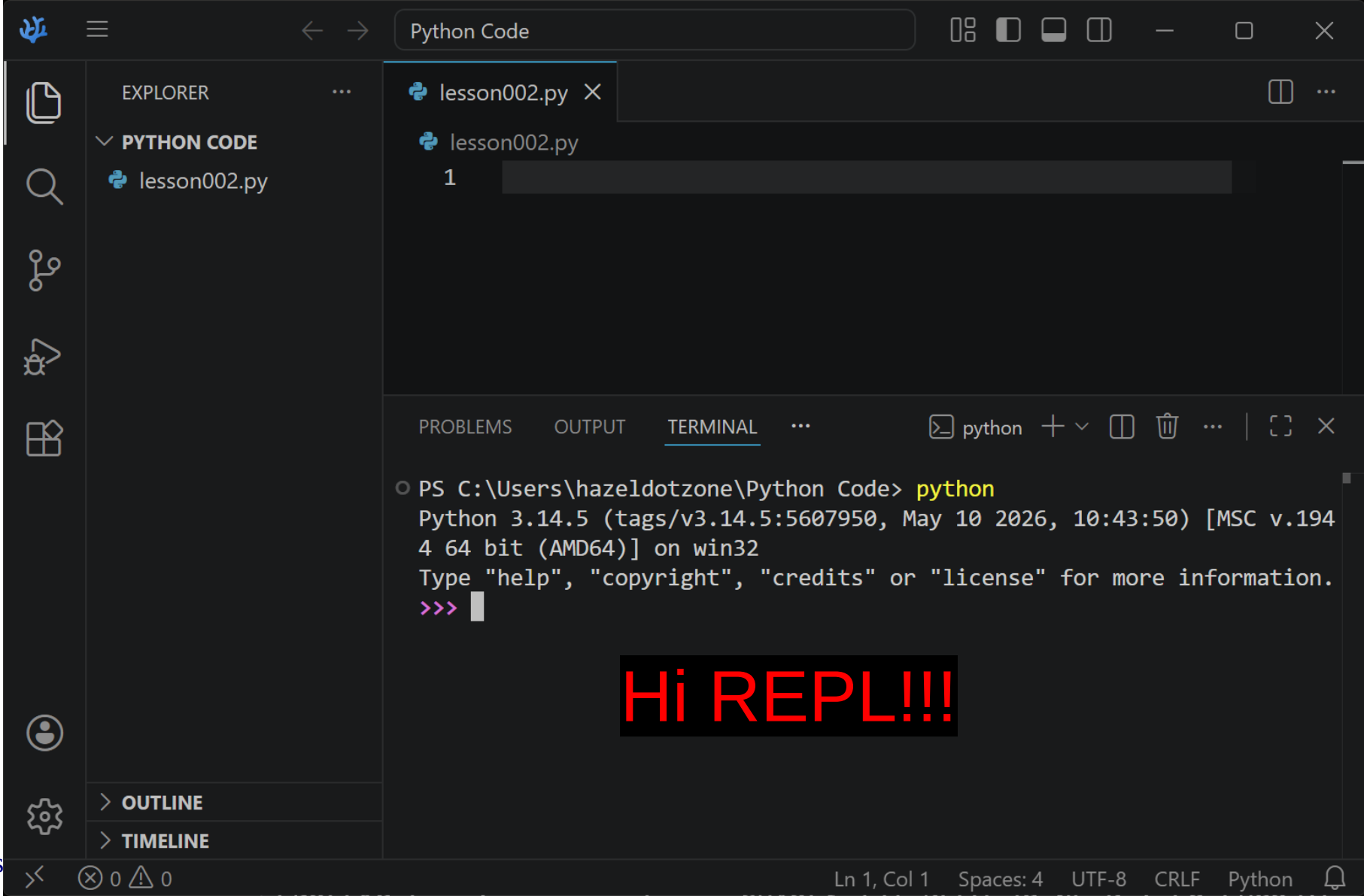
Type "help", "copyright", "credits" or "license" for more information.

>>>

Try running python!

https

8

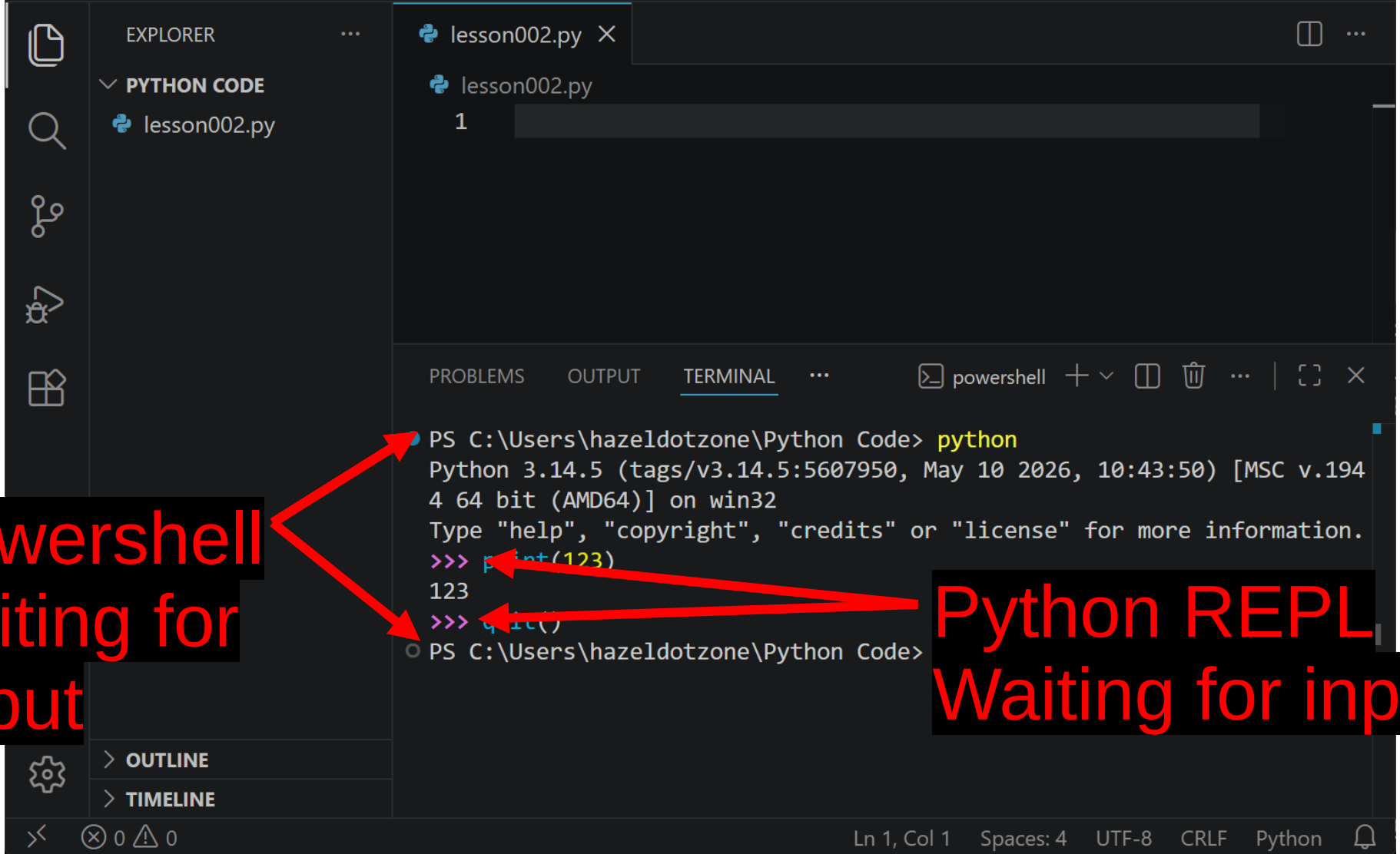


Hi REPL!!!

The image shows the Visual Studio Code interface. The Explorer sidebar on the left shows a file named 'lesson002.py' under the 'PYTHON CODE' folder. The main editor window displays the code for 'lesson002.py' with a single line containing the number '1'. Below the editor is the TERMINAL panel, which shows the following output:

```
PS C:\Users\hazeldotzone\Python Code> python
Python 3.14.5 (tags/v3.14.5:5607950, May 10 2026, 10:43:50) [MSC v.194
4 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print(123)
123
>>> quit()
PS C:\Users\hazeldotzone\Python Code>
```

quit() quits the REPL



**Powershell
waiting for
input**

**Python REPL
Waiting for input**

Each type of REPL or command-line *shell* shows a “prompt string” which is some output that the REPL or shell uses to tell you that its waiting for your input.

- PowerShell
 - PS C:\blah\blah\blah>
- Python REPL
 - >>>

```
PS C:\Users\hazeldotzone\Python Code> python
Python 3.14.5 (tags/v3.14.5:5607950, May 10 2026, 10:43:50) [MSC v.194
4 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print(123)
123
>>> quit()
PS C:\Users\hazeldotzone\Python Code> █
```

> OUTLINE

> TIMELINE

⏪ ⊗ 0 ⚠ 0

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 🔔

The image shows a screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows a folder named 'PYTHON CODE' containing a file named 'lesson002.py'. The main editor window displays the code for 'lesson002.py':

```
1 print(123)
2 print(type(123))
```

The bottom panel shows the 'TERMINAL' view with a PowerShell prompt. The command entered is:

```
PS C:\Users\hazelzone\Python Code> python .\lesson002.py
```

The output of the command is:

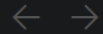
```
123
<class 'int'>
```

Red arrows point from a text box at the bottom to the 'python' command, the file path, and the file name in the terminal command.

python space filename
To run a Python file



File Edit Selection ...



Python Code



EXPLORER



PYTHON CODE



lesson002.py



lesson002.py X



lesson002.py

```
1 print(123)
2 print(type(123))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

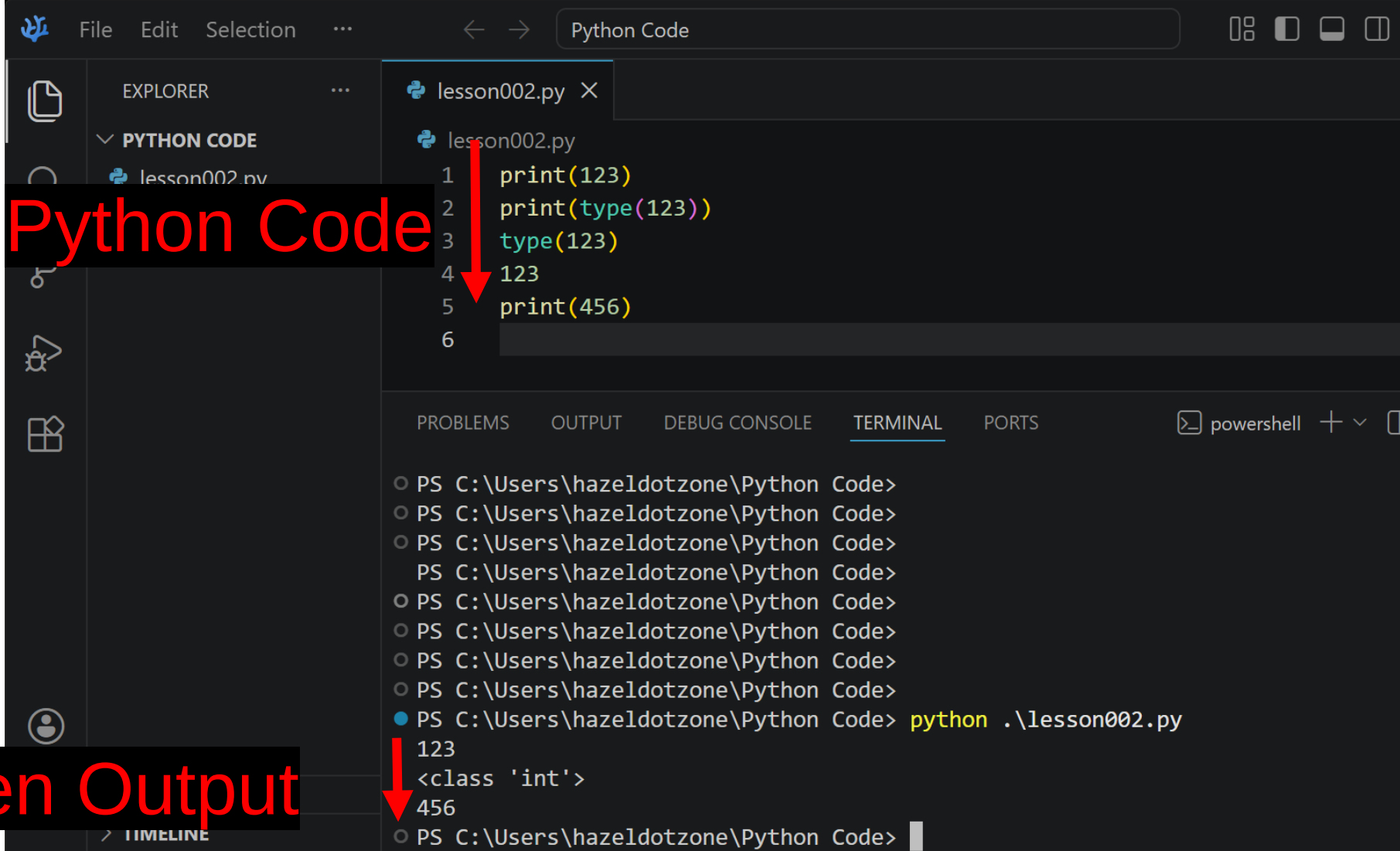


```
● PS C:\Users\hazel\dotzone\Python Code> python .\lesson002.py
123
<class 'int'>
○ PS C:\Users\hazel\dotzone\Python Code> █
```

Python Written Output

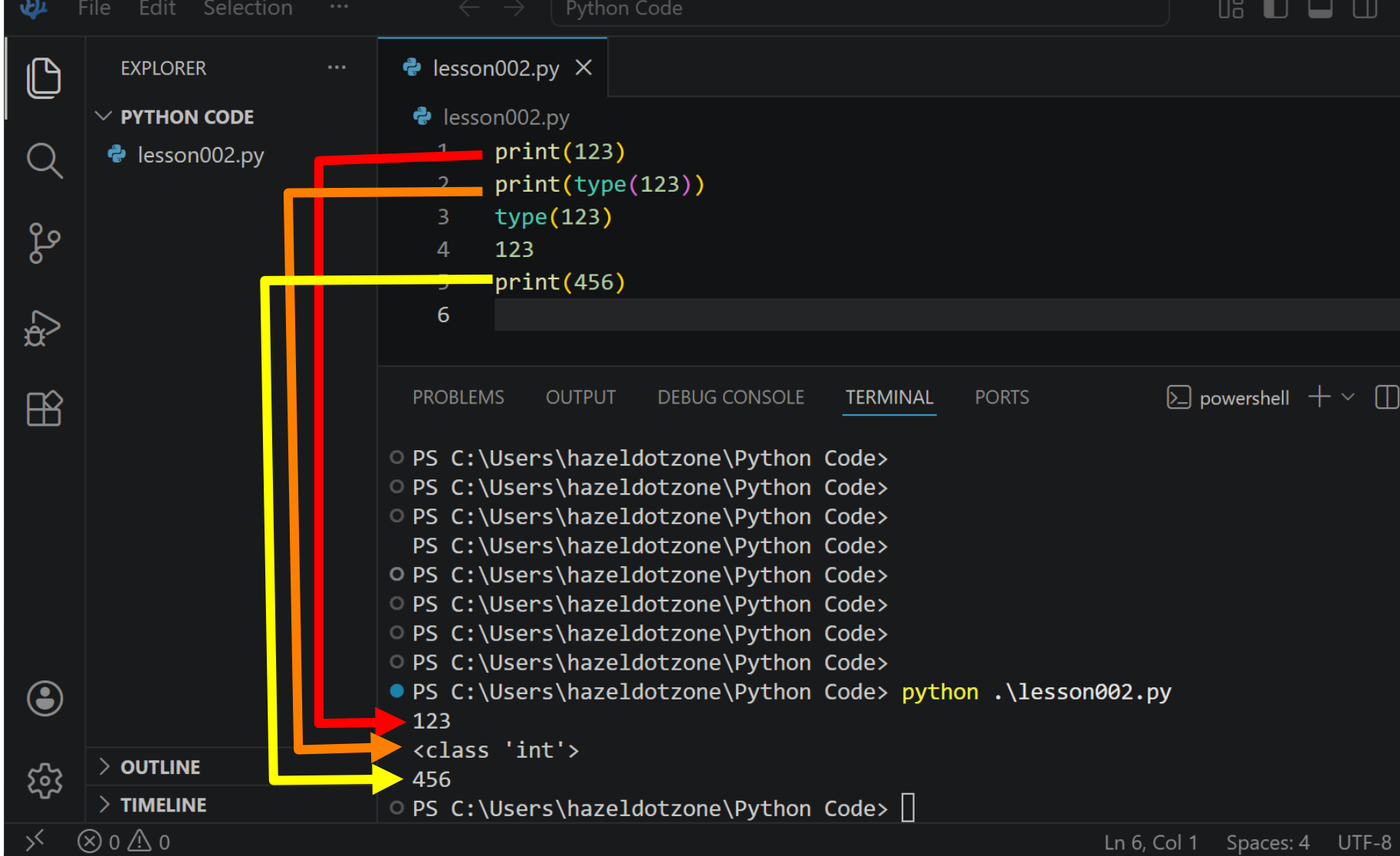
Running Python Files

- When we are running a Python code file...
 - no `>>>`, no REPL
 - Our only written output came from `print()`
 - Runs the Python code from first line to the last



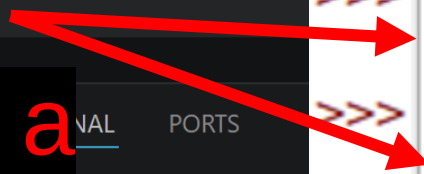
Python Code

Written Output



We no longer have a REPL's Output
Because
We no longer have a REPL!

```
Enter "help" below
>>> print(123)
123
>>> print(type(123))
<class 'int'>
>>> type(123)
<class 'int'>
>>> 123
123
>>> print(456)
456
>>> |
```



lesson002.py

EXPLORER

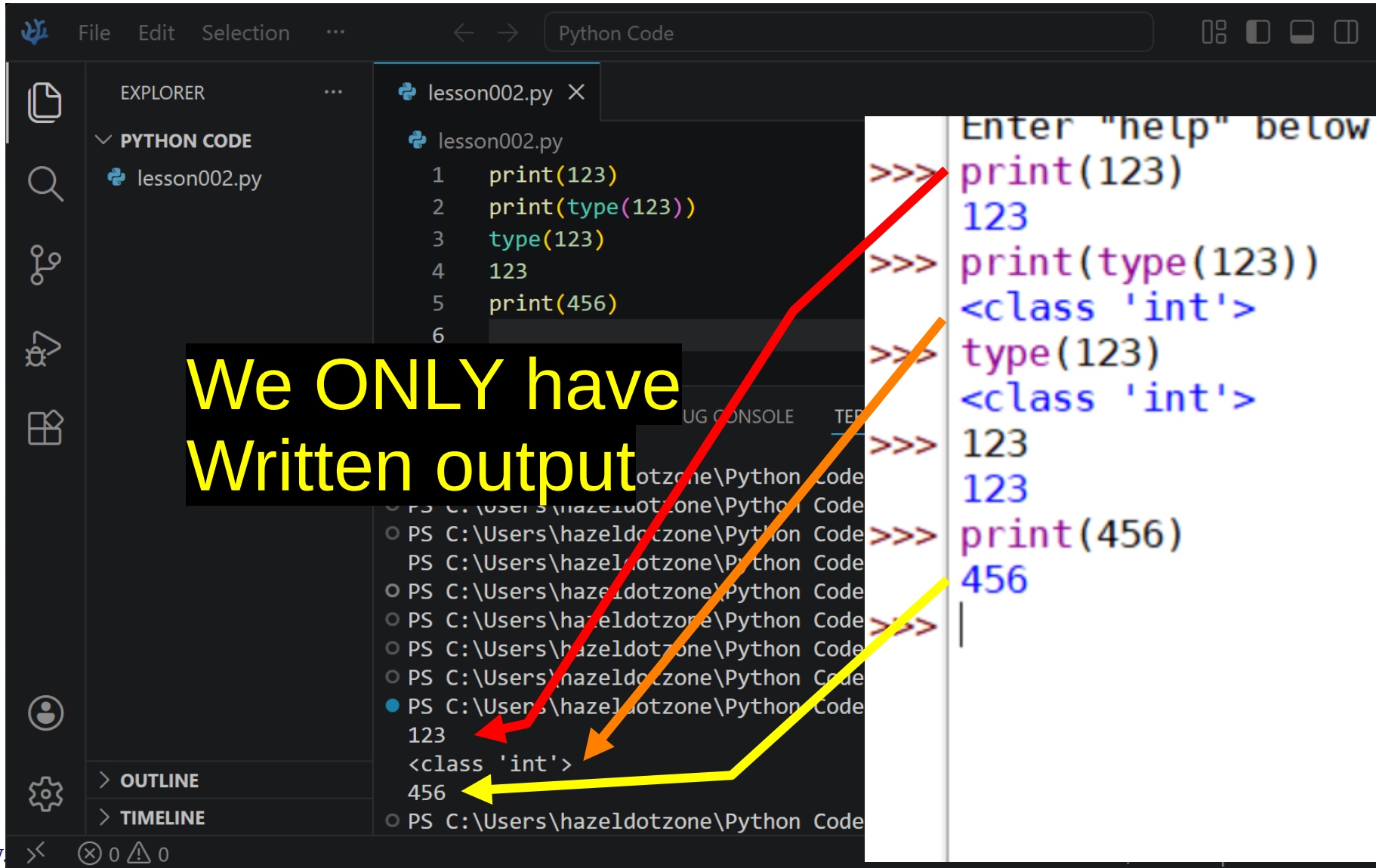
PYTHON CODE
lesson002.py

print(456)

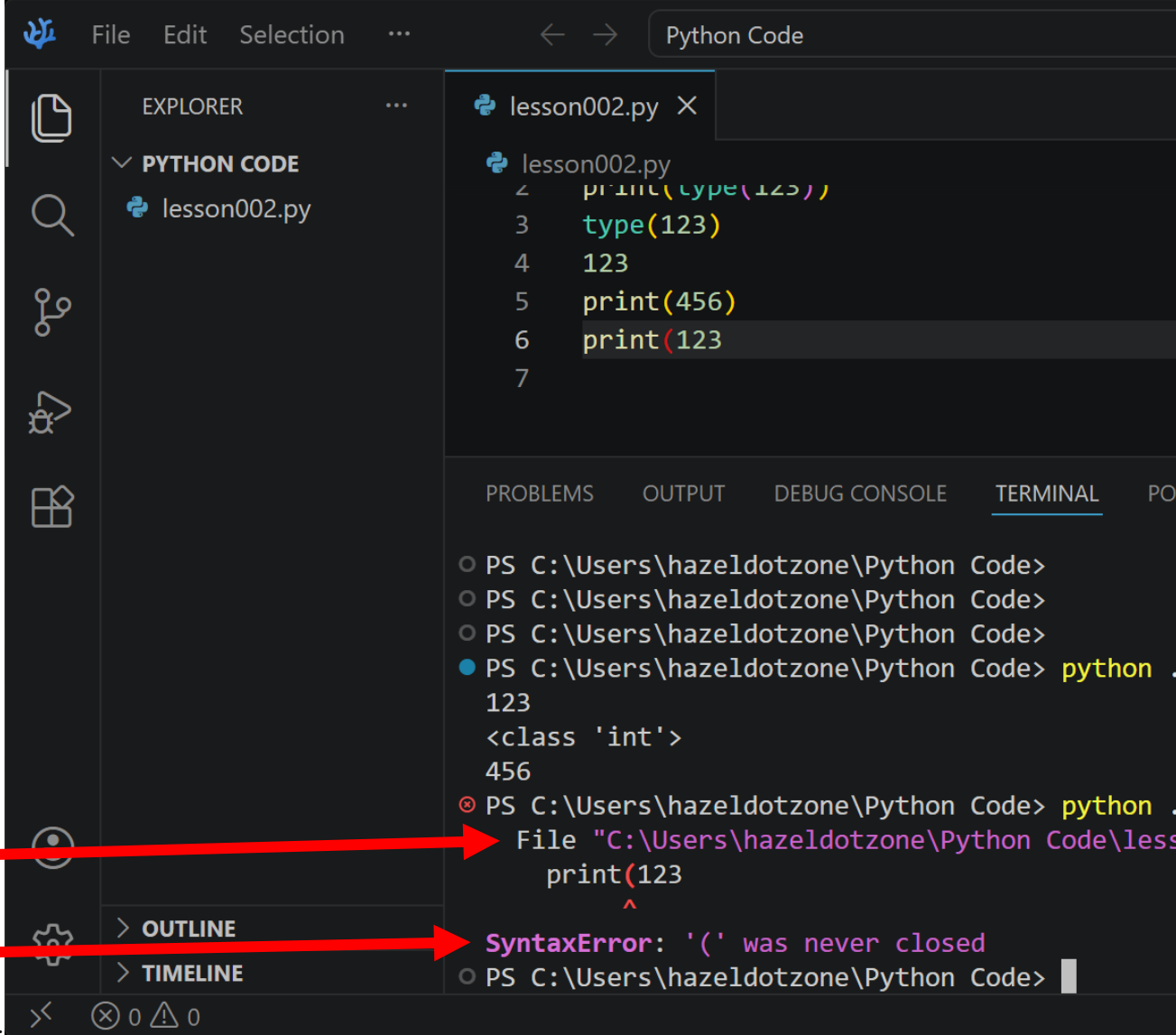
TERMINAL

```
C:\Users\hazel\dotzone\Python Code>
C:\Users\hazel\dotzone\Python Code>
C:\Users\hazel\dotzone\Python Code>
PS C:\Users\hazel\dotzone\Python Code>
PS C:\Users\hazel\dotzone\Python Code>
PS C:\Users\hazel\dotzone\Python Code>
PS C:\Users\hazel\dotzone\Python Code>
PS C:\Users\hazel\dotzone\Python Code>
PS C:\Users\hazel\dotzone\Python Code>
PS C:\Users\hazel\dotzone\Python Code> python .\lesso
123
<class 'int'>
456
PS C:\Users\hazel\dotzone\Python Code> |
```

OUTLINE
TIMELINE



We ONLY have
Written output



We may still have errors, though

Location

Error

Now the location information
In our error message is more important
Since it will tell us what line
Python thinks the fault is on

```
Python Code  
lesson002.py X  
lesson002.py  
2 print(type(123))  
3 type(123)  
4 123  
5 print(456)  
6 print(123  
7  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\hazeldotzone\Python Code>  
PS C:\Users\hazeldotzone\Python Code>  
PS C:\Users\hazeldotzone\Python Code>  
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py  
123  
<class 'int'>  
456  
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py  
File "C:\Users\hazeldotzone\Python Code\lesson002.py", line 6  
print(123  
    ^  
SyntaxError: '(' was never closed  
PS C:\Users\hazeldotzone\Python Code> █  
Ln 6, Col 10 Sp
```

We may still have errors, though

Location

Error

In this example, we forgot the)
Python helpfully tells us about it

```
Python Code  
lesson002.py X  
lesson002.py  
1 print(type(123))  
2  
3 type(123)  
4 123  
5 print(456)  
6 print(123  
7  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\hazeldotzone\Python Code>  
PS C:\Users\hazeldotzone\Python Code>  
PS C:\Users\hazeldotzone\Python Code>  
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py  
123  
<class 'int'>  
456  
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py  
File "C:\Users\hazeldotzone\Python Code\lesson002.py", line 6  
print(123  
    ^  
SyntaxError: '(' was never closed  
PS C:\Users\hazeldotzone\Python Code> █
```

We may still have errors, though

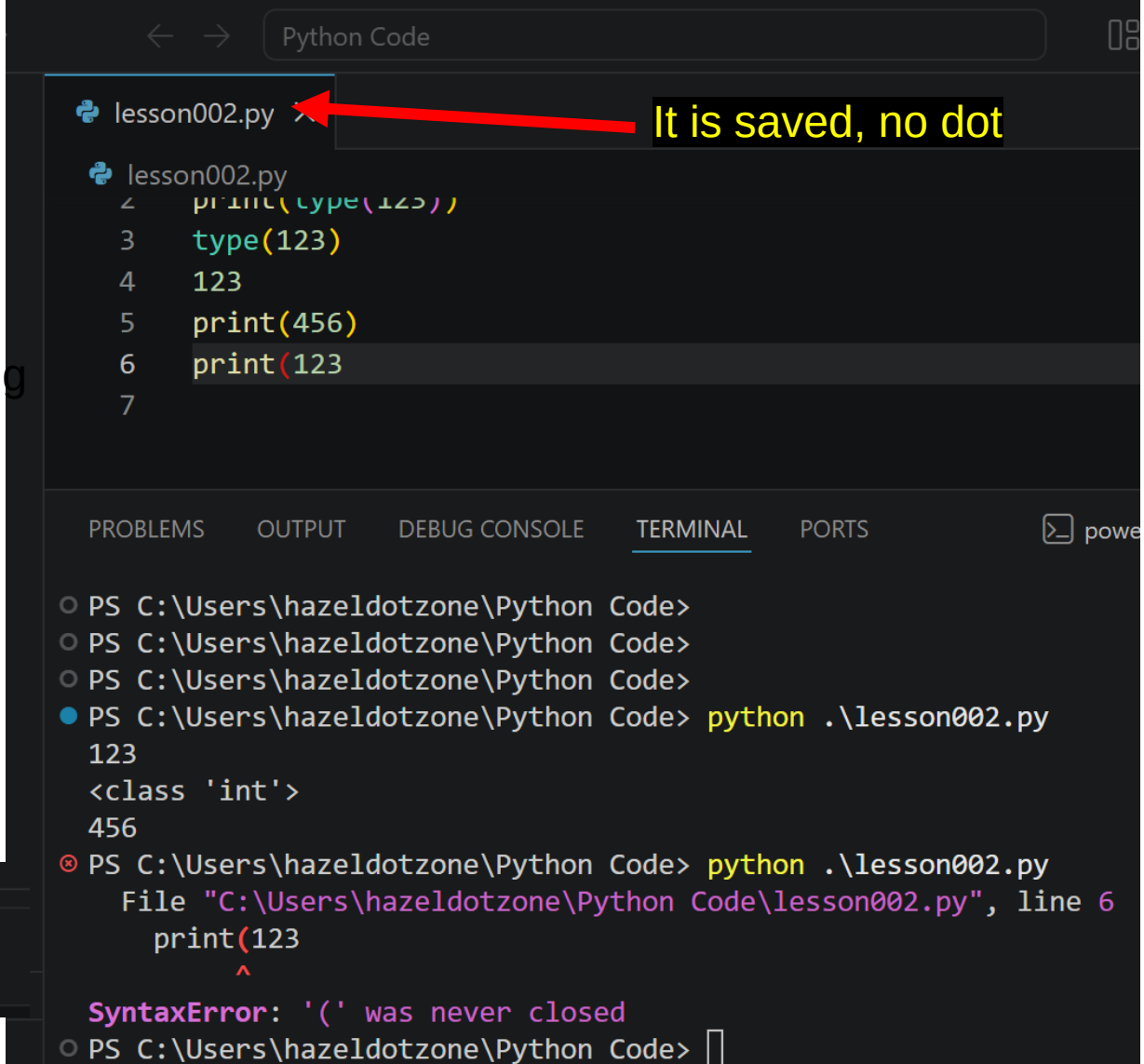
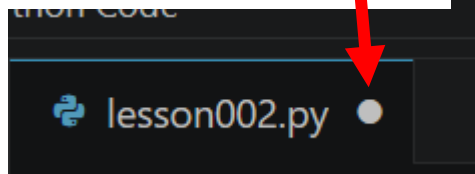
Location

Error

Fixed the code but Python is still giving
The same error message?

We forgot to save!

Need to
save



It is saved, no dot

Make sure you're always looking at

- The output of your last run
- From after you saved

Output of our previous run

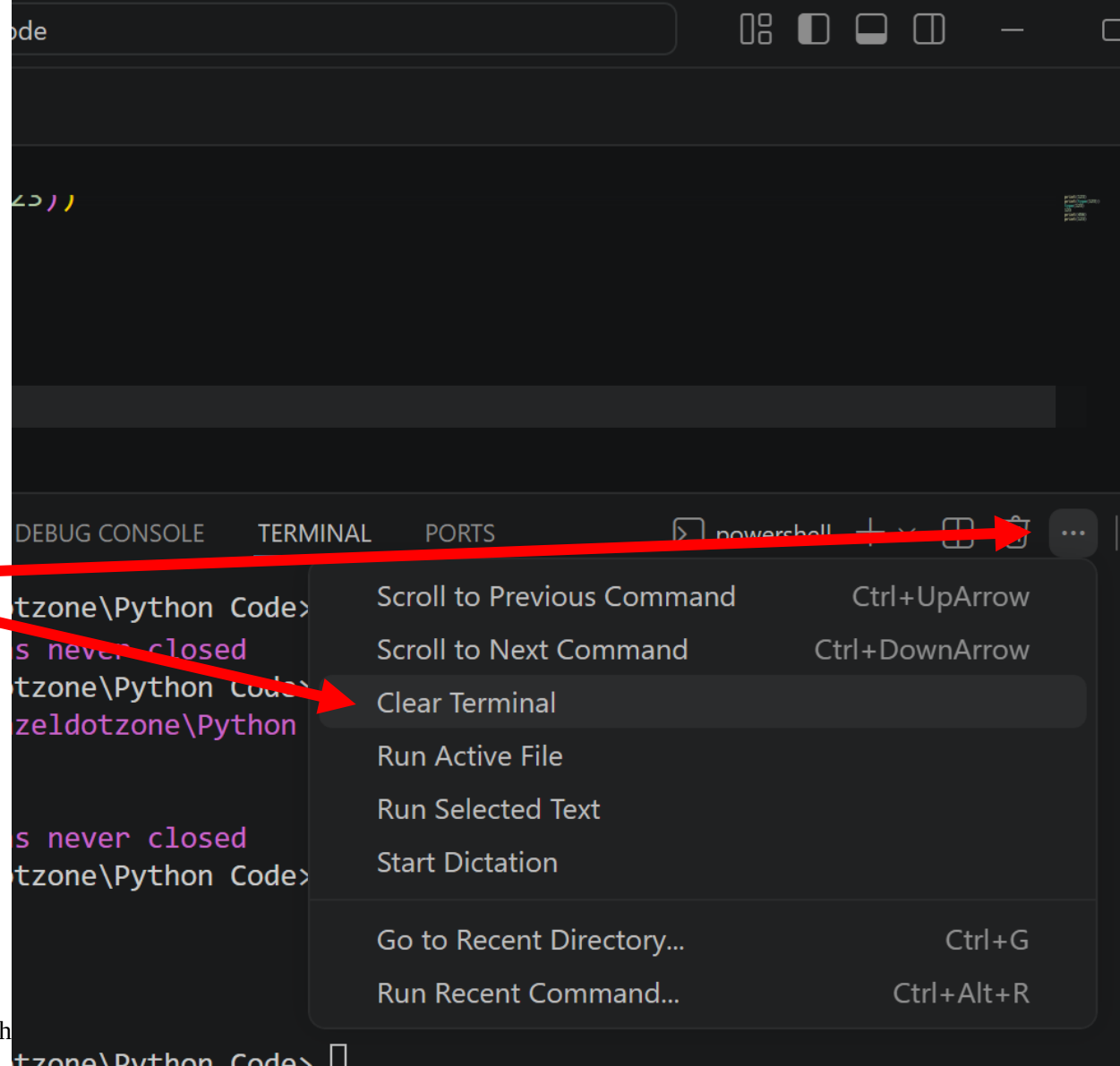
Output of our last run

```
lesson002.py X
lesson002.py
1 print(type(123))
2
3 type(123)
4 123
5 print(456)
6 print(123)
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\hazeldotzone\Python Code> python .\les
SyntaxError: '(' was never closed
⊗ PS C:\Users\hazeldotzone\Python Code> python .\les
File "C:\Users\hazeldotzone\Python Code\lesson00
print(123
^
SyntaxError: '(' was never closed
● PS C:\Users\hazeldotzone\Python Code> python .\les
123
<class 'int'>
456
123
○ PS C:\Users\hazeldotzone\Python Code>
```

We can always clear our Terminal if it gets too Cluttered!



Terminal Pro Tips

Since we are running the same command a lot of times, typing it out each time is annoying.

Instead, we can press the up arrow key to recall the previous command.
(Works in PowerShell and bash)

```
● PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
<class 'int'>
456
123
○ PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
```

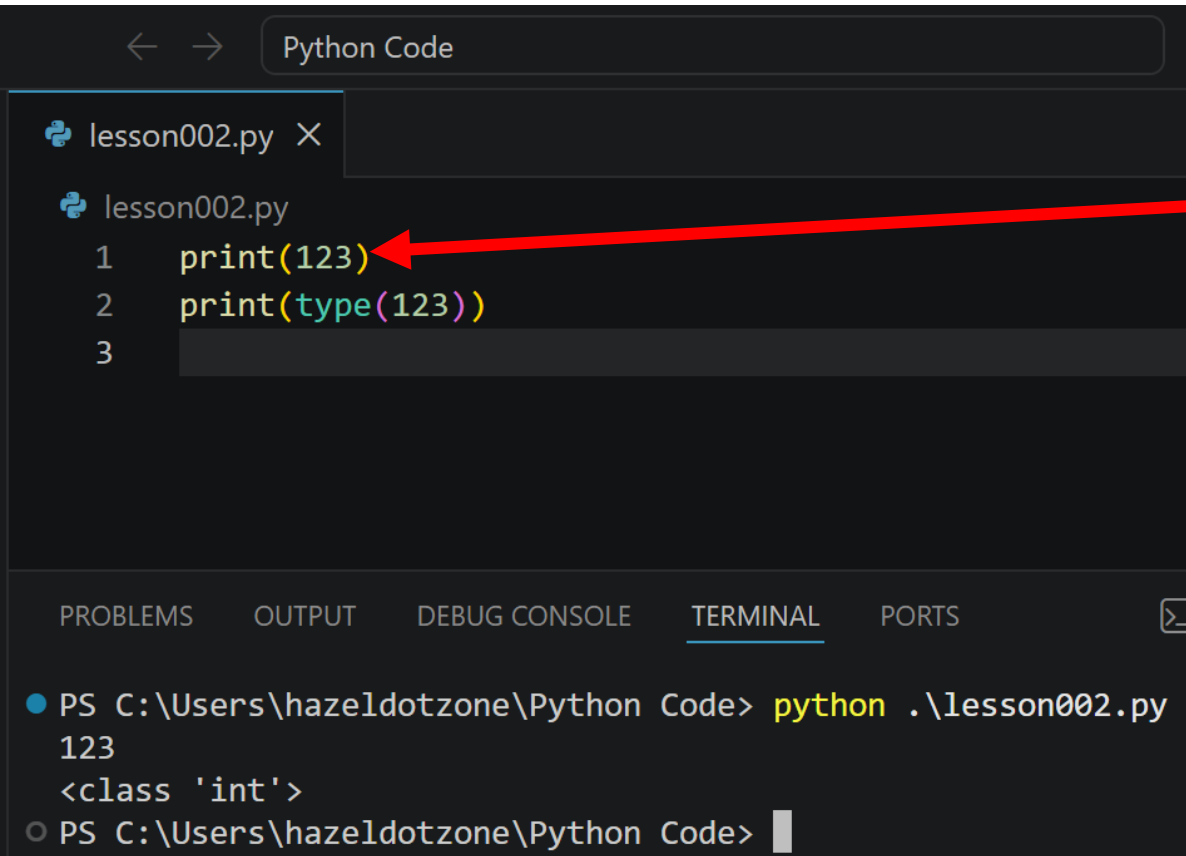
Codium Pro Tips

Since we are running the same command a lot of times, typing it out each time is annoying.

Instead, we can press the up arrow key to recall the previous command.
(Works in PowerShell and bash)

We can also use ctrl shift + to increase font size in all of VSCodium
ctrl shift - to decrease font size

More Literals



The screenshot shows a Python IDE window titled "Python Code" with a file named "lesson002.py". The code in the editor is:

```
1 print(123)
2 print(type(123))
3
```

The terminal at the bottom shows the execution of the script:

```
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
<class 'int'>
```

int Literal

More Literals

```
Python Code  
lesson002.py X  
lesson002.py  
1 print(123)  
2 print(type(123))  
3 print("hello")  
4 print(type("hello"))  
5  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py  
123  
<class 'int'>  
hello  
<class 'str'>  
PS C:\Users\hazeldotzone\Python Code>
```

int Literal

str Literal

Type of "hello" is printed

(of course since we're using the print function it's written output, REPL Isn't running!)

More Literals

```
Python Code  
lesson002.py X  
lesson002.py  
1 print(123)  
2 print(type(123))  
3 print("hello")  
4 print(type("hello"))  
5  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py  
123  
<class 'int'>  
hello  
<class 'str'>  
PS C:\Users\hazeldotzone\Python Code>
```

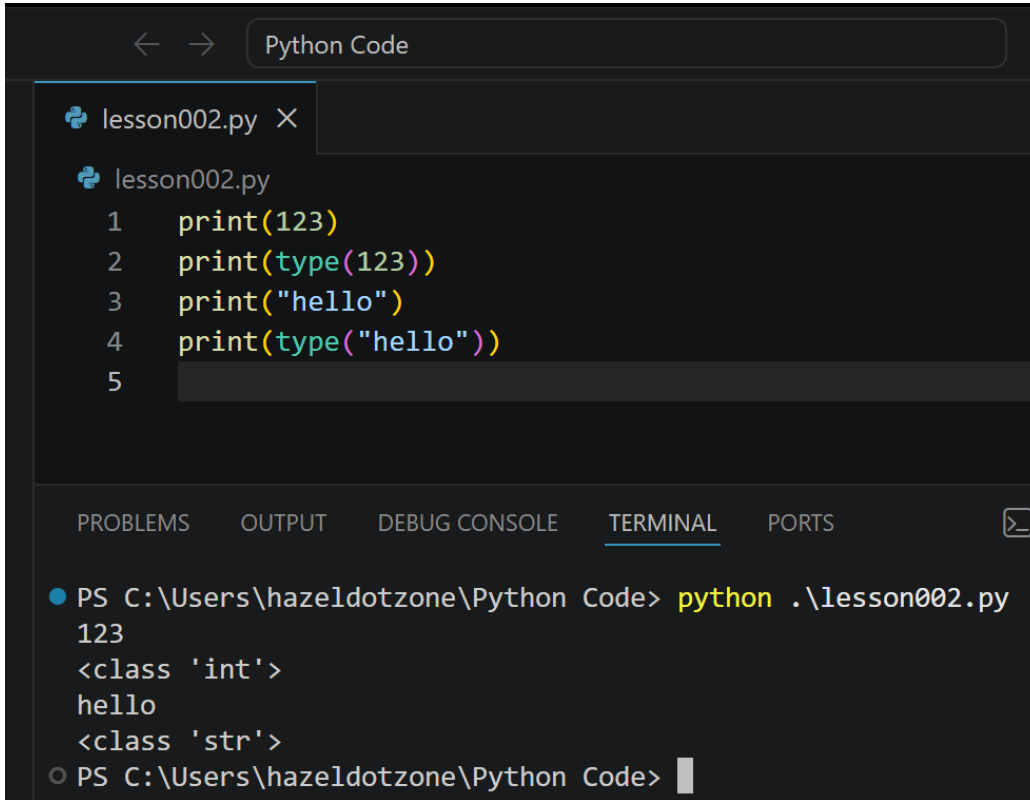
int Literal

str Literal

Type of "hello" is printed

Addition

- We can add `int`



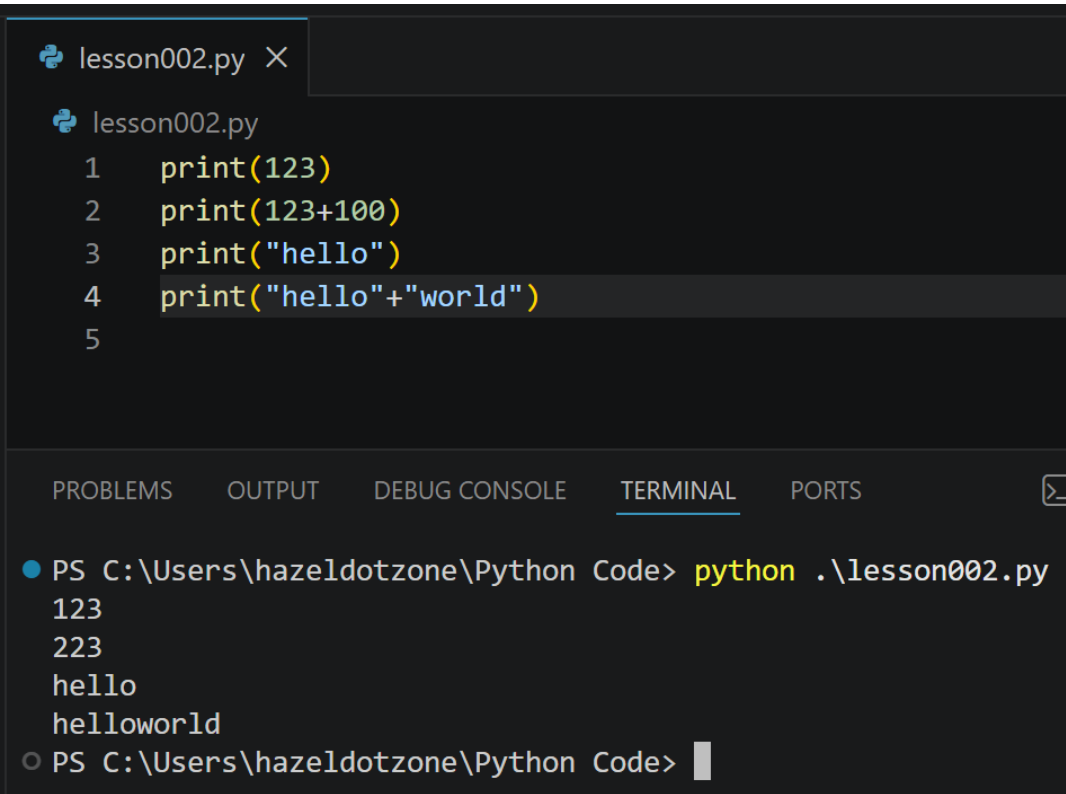
The screenshot shows a Python IDE window titled "Python Code" with a file named "lesson002.py". The code in the editor is as follows:

```
1 print(123)
2 print(type(123))
3 print("hello")
4 print(type("hello"))
5
```

Below the code editor, the "TERMINAL" tab is active, showing the execution of the script:

```
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
<class 'int'>
hello
<class 'str'>
```

Addition



```
lesson002.py X
lesson002.py
1 print(123)
2 print(123+100)
3 print("hello")
4 print("hello"+"world")
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
223
hello
helloworld
PS C:\Users\hazeldotzone\Python Code>
```

- We can add `int`
- We can add `str`?

Syntax Review

```
4 print("hello"+"world")
```

identifier

start of
arguments
delimiter

string
literal

binary
addition
operator

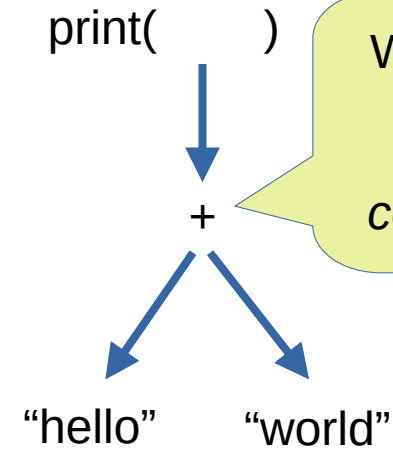
string
literal

end of
arguments
delimiter

Addition – Diagram Review

```
lesson002.py X
lesson002.py
1 print(123)
2 print(123+100)
3 print("hello")
4 print("hello"+"world")
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
223
hello
helloworld
PS C:\Users\hazeldotzone\Python Code>
```



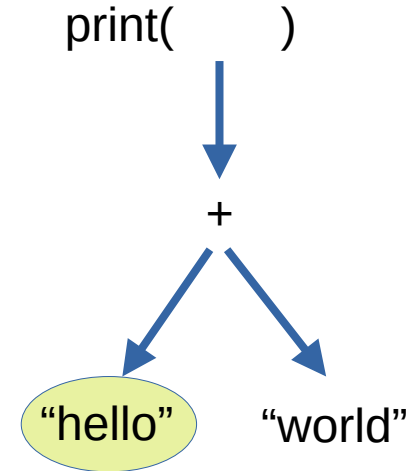
When applied to strings I perform *concatenation*

It's still the same operator, but what it does depends on what goes into it!

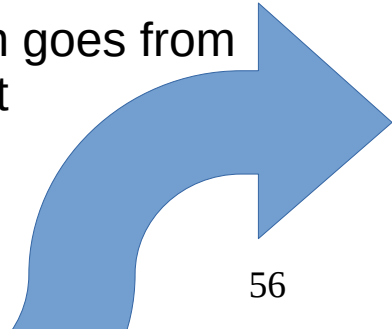
Addition – Diagram Review

```
lesson002.py X
lesson002.py
1 print(123)
2 print(123+100)
3 print("hello")
4 print("hello"+"world")
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
223
hello
helloworld
PS C:\Users\hazeldotzone\Python Code>
```



Evaluation goes from
left to right
bottom up

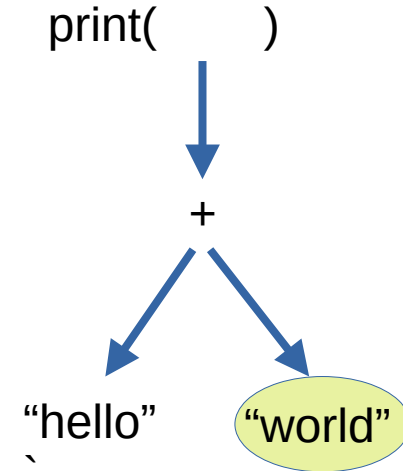


Addition – Diagram Review

```
lesson002.py X
lesson002.py
1 print(123)
2 print(123+100)
3 print("hello")
4 print("hello"+"world")
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
223
hello
helloworld
PS C:\Users\hazeldotzone\Python Code>
```

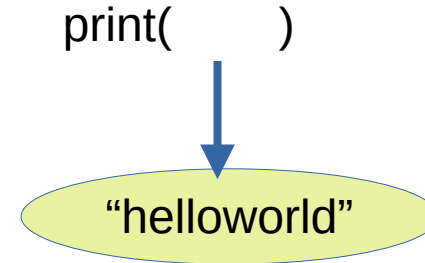


Evaluation goes from
left to right
bottom up

Addition – Diagram Review

```
lesson002.py X
lesson002.py
1 print(123)
2 print(123+100)
3 print("hello")
4 print("hello"+"world")
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
223
hello
helloworld
PS C:\Users\hazeldotzone\Python Code>
```



Evaluation goes from
left to right
bottom up

Addition – Diagram Review

```
lesson002.py X
lesson002.py
1 print(123)
2 print(123+100)
3 print("hello")
4 print("hello"+"world")
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
223
hello
helloworld
PS C:\Users\hazeldotzone\Python Code>
```

print()

Run the call
With one argument
"helloworld"

Evaluation goes from
left to right
bottom up

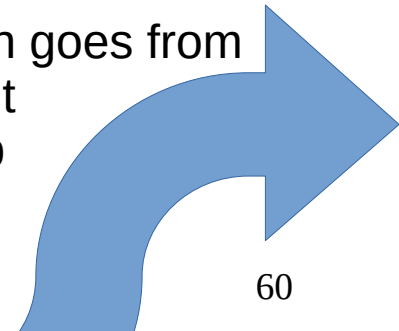
Addition – Diagram Review

None

```
lesson002.py X
lesson002.py
1 print(123)
2 print(123+100)
3 print("hello")
4 print("hello"+"world")
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
223
hello
helloworld
PS C:\Users\hazeldotzone\Python Code>
```

Evaluation goes from
left to right
bottom up



Addition – Evaluation Order

Remember...
they're
literals!

```
lesson002.py X
lesson002.py
1 print(123)
2 print(123+100)
3 print("hello")
4 print("hello"+"world")
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\hazeldotzone\Python Code> python .\lesson002.py
123
223
hello
helloworld
PS C:\Users\hazeldotzone\Python Code>
```

1) "hello" → "hello"

2) "world" → "world"

3) + → "helloworld"

4) print()

5) None

