

Master VS Code for Python  
like a pro with these time-  
saving tips and shortcuts!"

Data  
Science  
Essentials

# VS code Python Tricks and Tips-Part2 (using Microsoft CoPilot)



<https://www.linkedin.com/in/sidharthmahotra/>



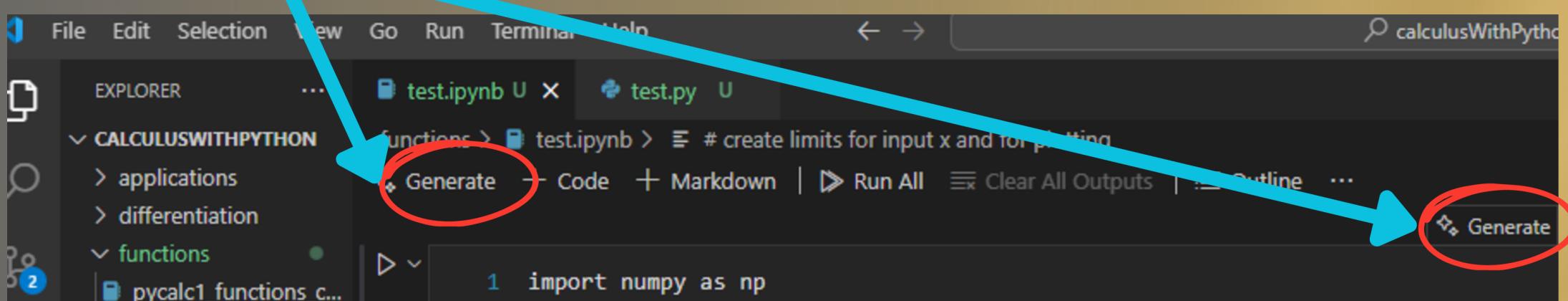
# VS code Python Tricks and Tips



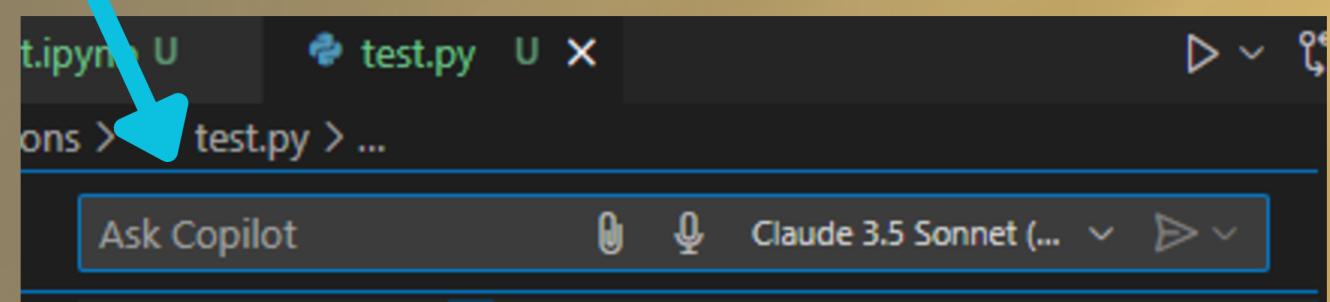
## Accessing Copilot



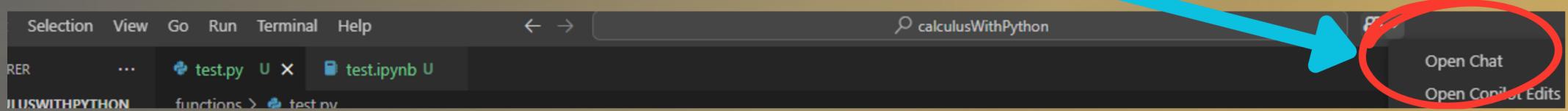
Jupyter notebook -> Select  
Generate



Python Script-> Type CTRL + I



Python Script/Jupyter notebook-> Click Open Chat





# VS code Python Tricks and Tips

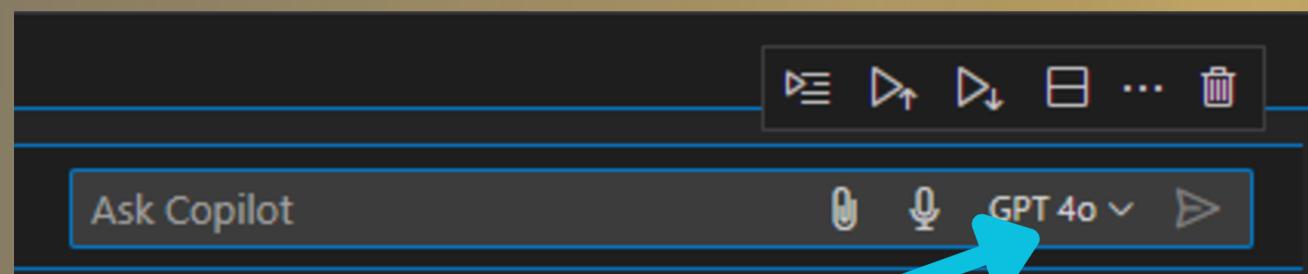


## Using Copilot



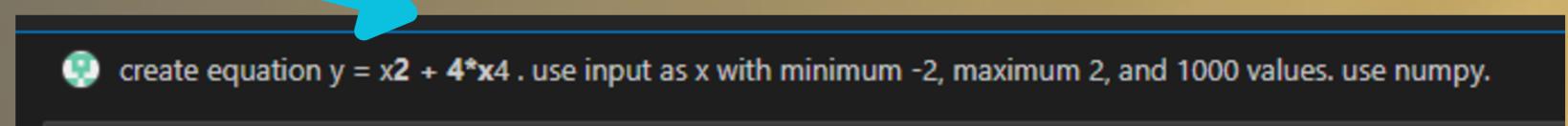
## Jupyter notebook

Click **Generate**



Pro Tip: Choose Model : OpenAi GPT 4o or Claude 3.5 Sonnet

Write Prompt for task at hand ->  
Press Play/Right Arrow button



Accept the code , Discard it or  
rewrite the prompt

```
1 import numpy as np
2
3 x = np.linspace(-2, 2, 1000)
4 y = x**2 + 4*x**4
5
6 # Display the first few values of x and y
7 x[:5], y[:5]
```



Siddharth Mahotra

# VS code Python Tricks and Tips

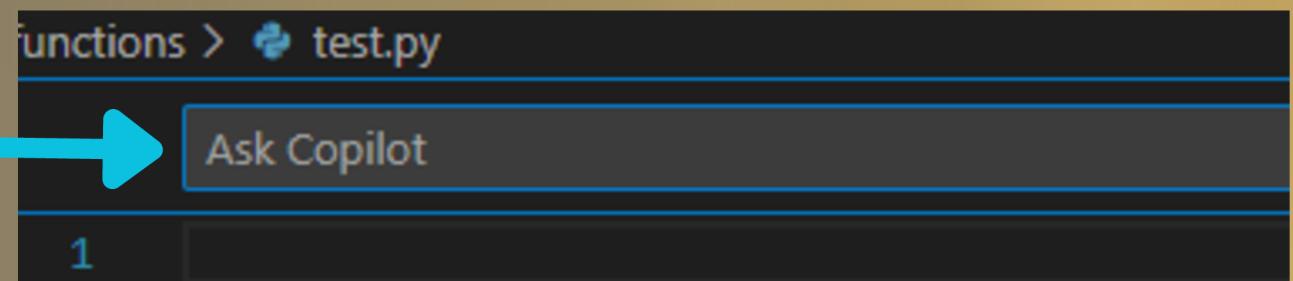


## Using Copilot

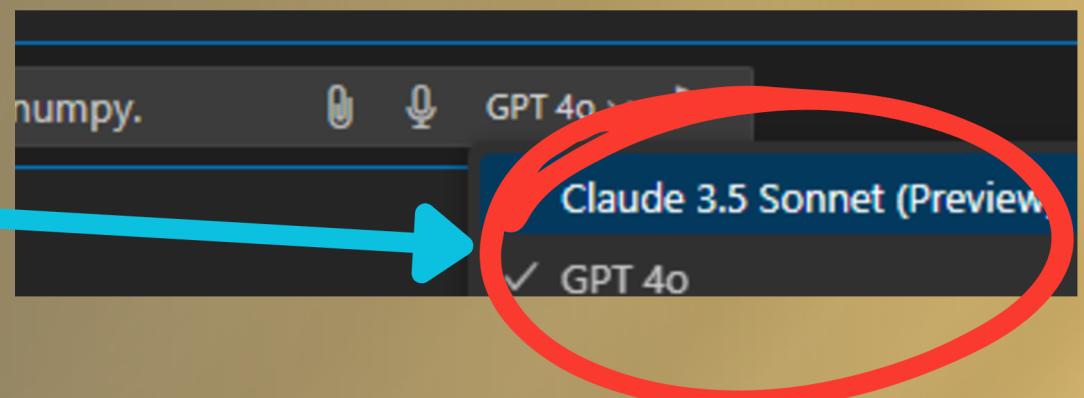


### Python Script

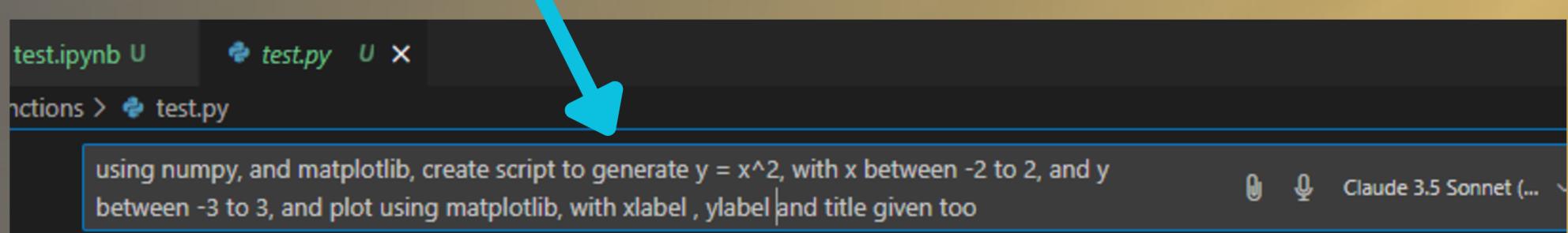
Press Ctrl + I -> opens Copilot



Pro Tip: Choose Model : OpenAi GPT 4o or Claude 3.5 Sonnet



Write Prompt for task at hand ->  
Press Play/Right Arrow button





Sidharth Mahotra

# VS code Python Tricks and Tips

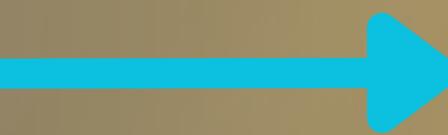


## Using Copilot



## Python Script

Prompt result: Generated code



```
test.ipynb U test.py U X
functions > test.py > ...
1 import numpy as np
2
3 import matplotlib.pyplot as plt
4
5 # Generate x values from -2 to 2
6 x = np.linspace(-2, 2, 100)
7
8 # Calculate y values using x^2
9 y = x**2
10
11 # Create the plot
12 plt.figure(figsize=(8, 6))
13 plt.plot(x, y, 'b-', label='y = x^2')
14
15 # Set axis limits
16 plt.xlim(-2, 2)
17 plt.ylim(-3, 3)
18
19 # Add labels and title
20 plt.xlabel('x')
21 plt.ylabel('y')
22 plt.title('Quadratic Function: y = x^2')
23
24 # Add grid and legend
25 plt.grid(True)
26 plt.legend()
27
28 # Display the plot
29 plt.show()
```



Sidharth Mahotra

# VS code Python Tricks and Tips

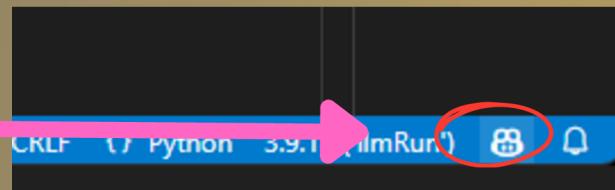


## Using Copilot

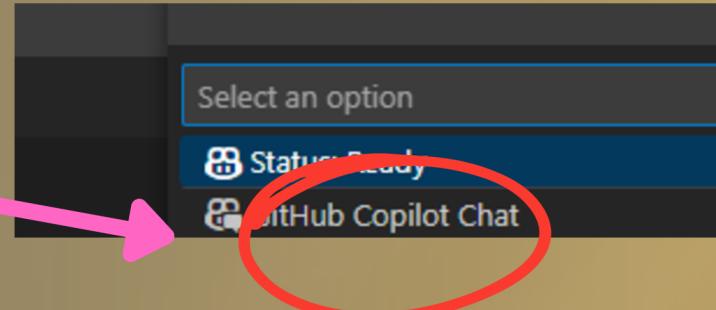


### Python Script

Enabling chat again in script ->  
Press icon on lower right corner :  
**Show Copilot Status Menu**



Choose : **GitHub Copilot Chat**



The screenshot shows a Python script named `test.py` in the editor. In the bottom right corner, there is a "Ask Copilot" window. The window has a "Copilot is powered by AI, so mistakes are possible. Review output carefully before use." message. It also contains a list of commands:

- ∅ or type # to attach context
- @ to chat with extensions
- Type / to use commands
- /fix the problems in my code
- /tests add unit tests for my code
- /execute show the selected code works

A pink arrow points from the "GitHub Copilot Chat" menu option in the previous screenshot to the "Ask Copilot" window in this screenshot.

Prompt in the right panel Chat Window : Get results

Can be used for general programming or any other questions too





Sidharth Mahotra

# VS code Python Tricks and Tips



## Using Copilot

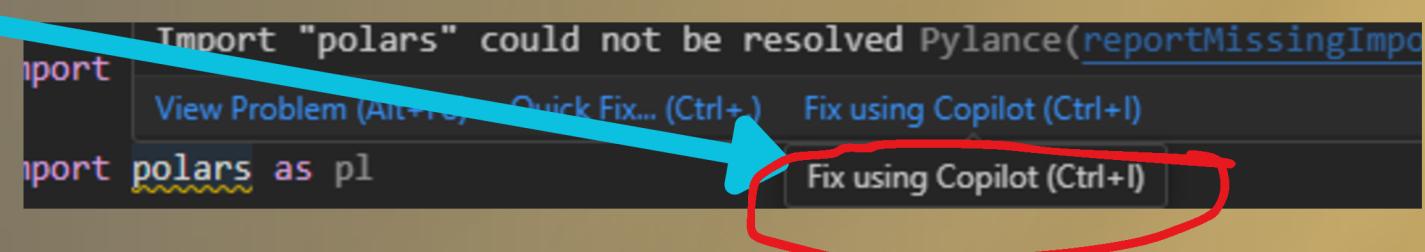


## Fix Code errors

Hover cursor over **Yellow Wavy underlined code**

```
test.ipynb U X test.py U
functions > test.ipynb > import numpy as np
Generate + Code + Markdown | ▶
1 import numpy as np
2
3 import polars as pl
```

Check the message and choose -  
> **Fix using Copilot**



Review the response from Copilot  
( **Highlighted in green**)

/fix Import "polars" could not be resolved

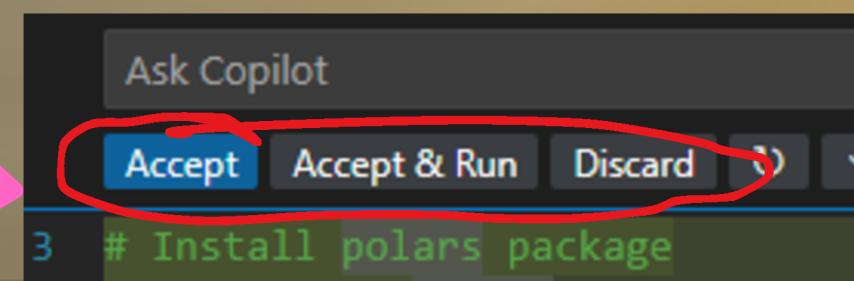
To fix the "Import 'polars' could not be resolved" error, you need to install the `polars` package. You can do this using the `%pip install` magic command in a new cell.

Ask Copilot

Accept Accept & Run Discard

```
3 # Install polars package
4 %pip install polars
```

Accept or Discard the solution





Siddharth Mahotra

# VS code Python Tricks and Tips



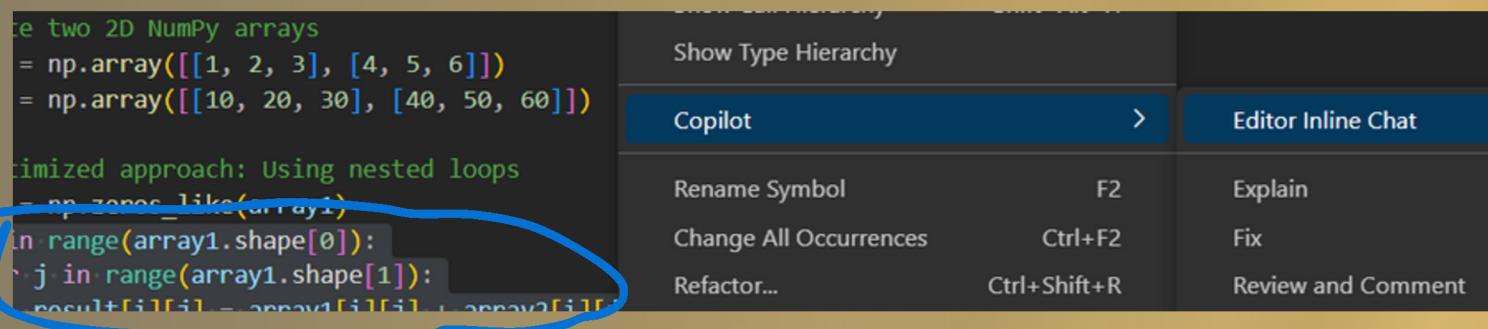
## Using Copilot



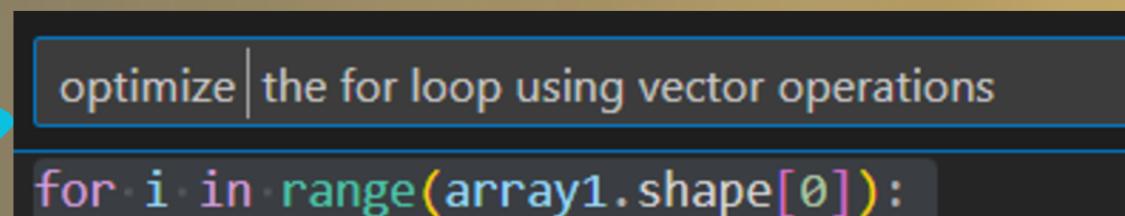
## Optimize code

Unoptimized portion of the code

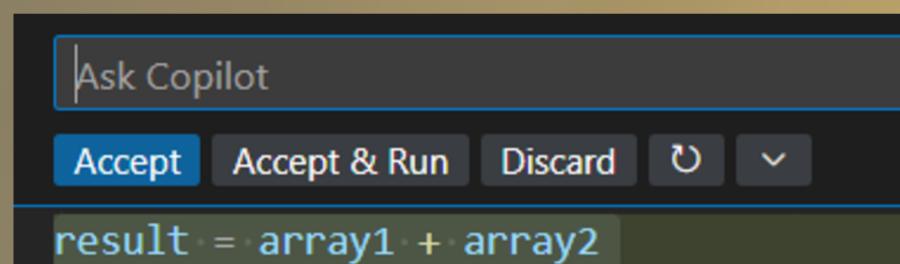
```
1 # Create two 2D NumPy arrays
2 array1 = np.array([[1, 2, 3], [4, 5, 6]])
3 array2 = np.array([[10, 20, 30], [40, 50, 60]])
4
5 # Unoptimized approach: Using nested loops
6 result = np.zeros_like(array1)
7 for i in range(array1.shape[0]):
8     for j in range(array1.shape[1]):
9         result[i][j] = array1[i][j] + array2[i][j]
10
```



Enter prompt for Copilot to  
optimize the code



Accept or Discard the solution



Accepted solution that matches  
unoptimized solution

```
1 # Create two 2D NumPy arrays
2 array1 = np.array([[1, 2, 3], [4, 5, 6]])
3 array2 = np.array([[10, 20, 30], [40, 50, 60]])
4
5 # Optimized approach: Using Copilot suggestion
6 result = np.zeros_like(array1)
7 result = array1 + array2
8
9 print(result)
✓ 0.0s
...
[[11 22 33]
 [41 55 66]]
```



Sidharth Mahotra

# VS code Python Tricks and Tips



## Using Copilot

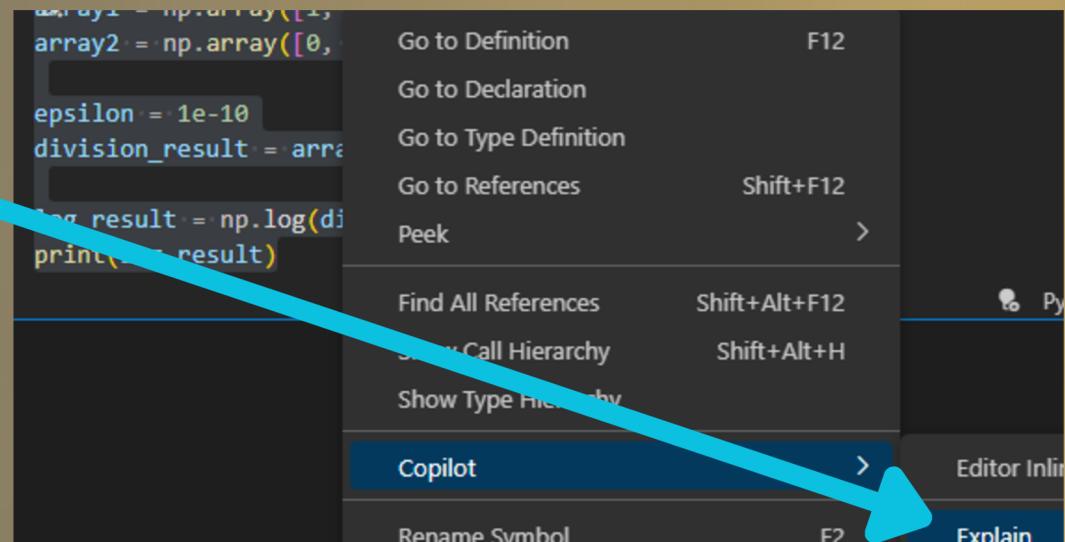


## Understand code

Code snippet without comments

```
1
2 array1 = np.array([1, 2, 3, 4])
3 array2 = np.array([0, 2, 0, 4])
4
5 epsilon = 1e-10
6 division_result = array1 / (array2 + epsilon)
7
8 log_result = np.log(division_result)
9 print(log_result)
```

Select code to be optimized ->  
Right click -> Copilot -> Explain



The 'Explain' output panel displays the following analysis:

In the provided code, two NumPy arrays, `array1` and `array2`, are created using the `np.array` function. `array1` contains the elements `[1, 2, 3, 4]`, and `array2` contains the elements `[0, 2, 0, 4]`. These arrays are used to perform element-wise operations.

To avoid division by zero errors, a small constant `epsilon` with a value of `1e-10` is added to `array2`. This ensures that no element in `array2` is zero, preventing any division by zero issues. The division operation `array1 / (array2 + epsilon)` is then performed element-wise, resulting in a new array `division_result` where each element of `array1` is divided by the corresponding element of `array2` plus `epsilon`.

Next, the natural logarithm of each element in `division_result` is calculated using the `np.log` function, resulting in the `log_result` array. Finally, the `print` function is used to output the `log_result` array to the console. This sequence of operations demonstrates how to safely perform element-wise division and logarithmic transformation on NumPy arrays while avoiding division by zero.

Review explanation in workspace



# VS code Python Tricks and Tips



## Using Copilot



## Add inline comments

Code without comments

```
6 result = np.zeros_like(array1)
7 result = array1 + array2
8
9 print(result)
```

At end of code line -> Press Tab  
-> comment shown in Gray ->  
Tab again -> Accept in Green

```
## Unoptimized approach. Using nested loops
result = np.zeros_like(array1) # Create an array of zeros with the same shape as array1
result = array1 + array2
print(result)
```

add comments for rest of the code

```
5 result = np.zeros_like(array1) # Create an array of zeros with the same shape as array1
6 result = array1 + array2      # Add the two arrays element-wise
7
8
9 print(result) # Output: [[11 22 33]]
```



Siddharth Mahotra

# VS code Python Tricks and Tips



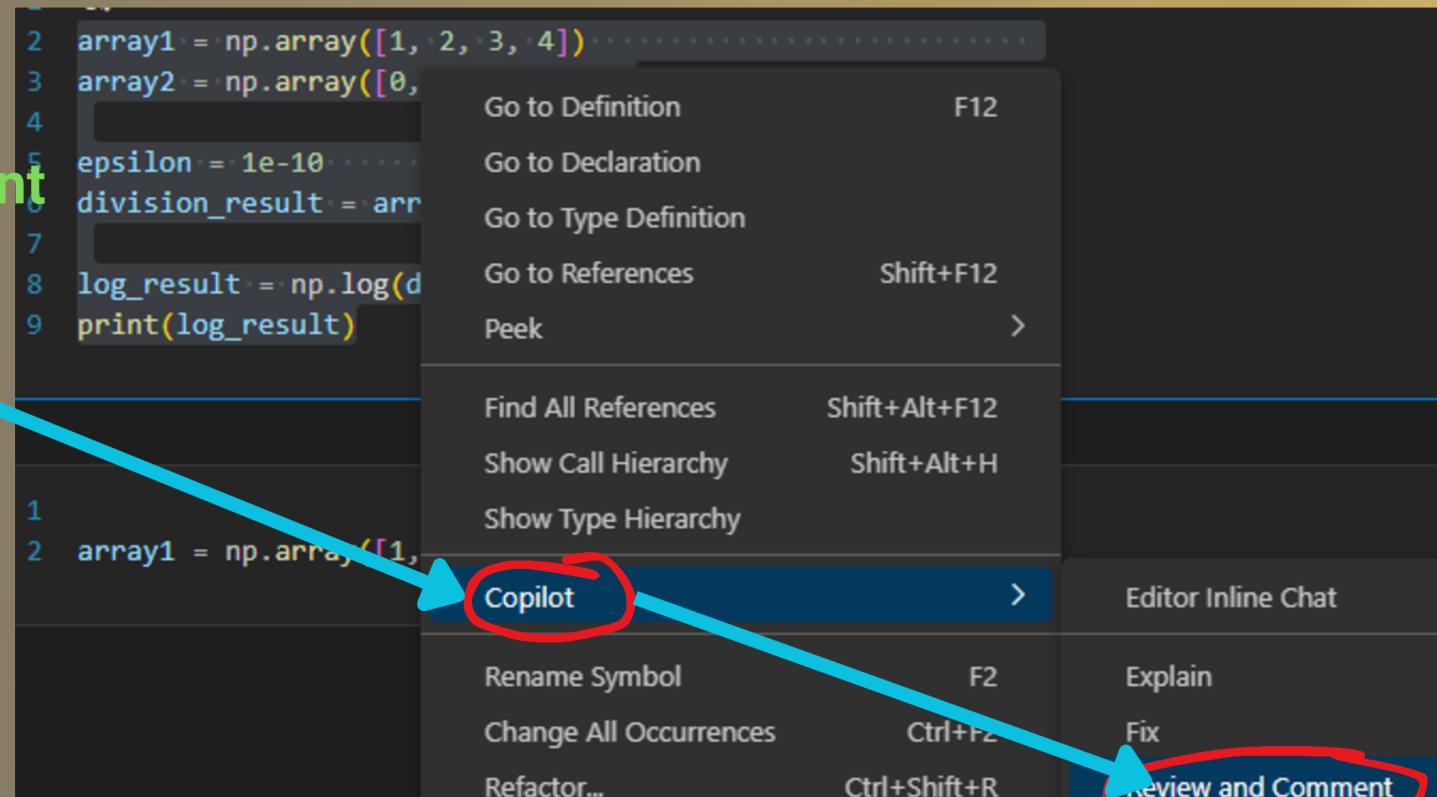
## Using Copilot



## Review and Comment

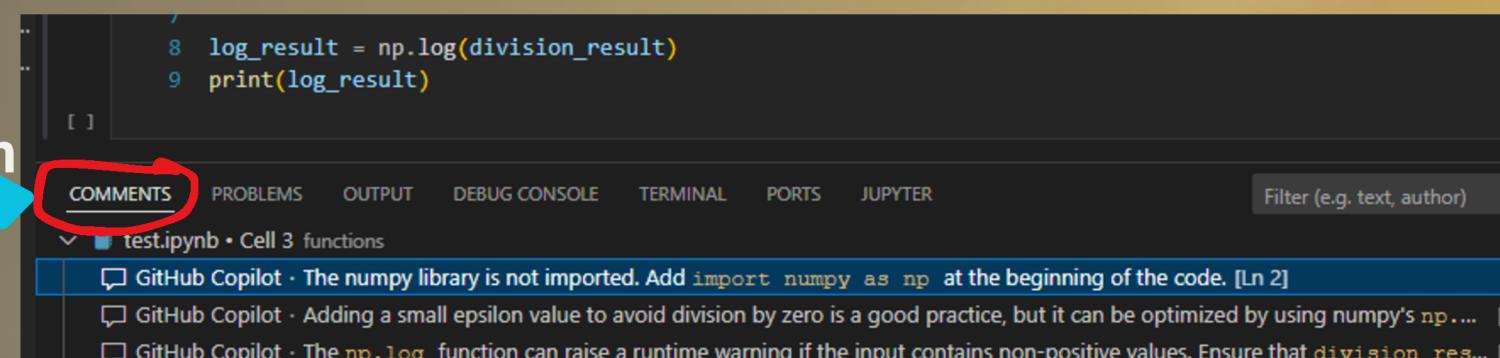
Code without comments

```
1
2 array1 = np.array([1, 2, 3, 4])
3 array2 = np.array([0, 2, 0, 4])
4
5 epsilon = 1e-10
6 division_result = array1 / (array2 + epsilon)
7
8 log_result = np.log(division_result)
9 print(log_result)
```



Select -> Right click ->  
Copilot -> Review and Comment

Choose **Comments** at bottom  
pane -> Review suggestions





Sidharth Mahotra

# VS code Python Tricks and Tips

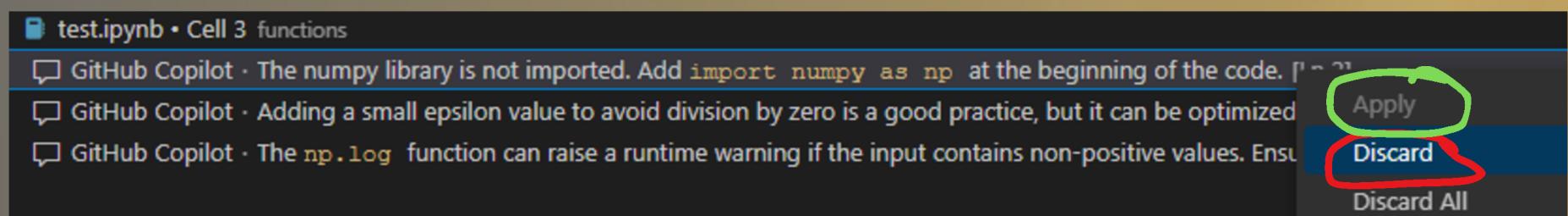


## Using Copilot



## Review and Comment - cntd

**Discard comment ( if not relevant) or apply the change if needed**



Choose Comments at bottom  
pane -> Review suggestions



Sidharth Mahotra

If you  
**find this**  
helpful, please  
like and share  
it

