VS Code Debugging Feature

What is Debugging?

Debugging is the process of detecting and removing bugs (errors or unintended behavior) from any kind of code. The process is by no means restricted to data science, which merely adopted it from software engineering. And we can use Debugging to learn how a set of code works/what the codes do in detail (flow of the program/the codes executed).

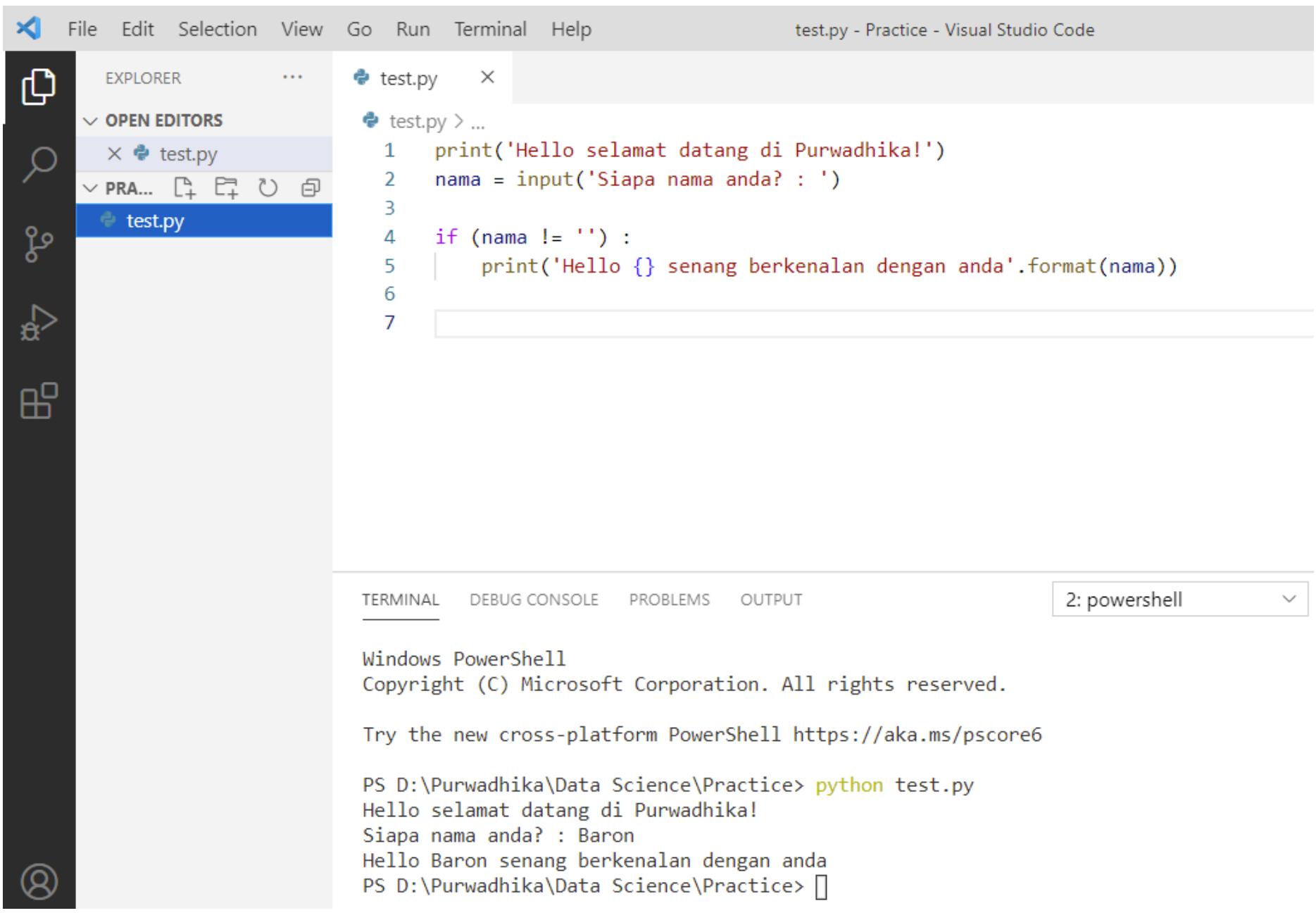
Debugging in VS Code

Let's dive right into the debugging. we will use this code for the demonstration.

```
print('Hello selamat datang di Purwadhika!')
nama = input('Siapa nama anda? : ')

if (nama != '') :
    print('Hello {} senang berkenalan dengan anda'.format(nama))
```

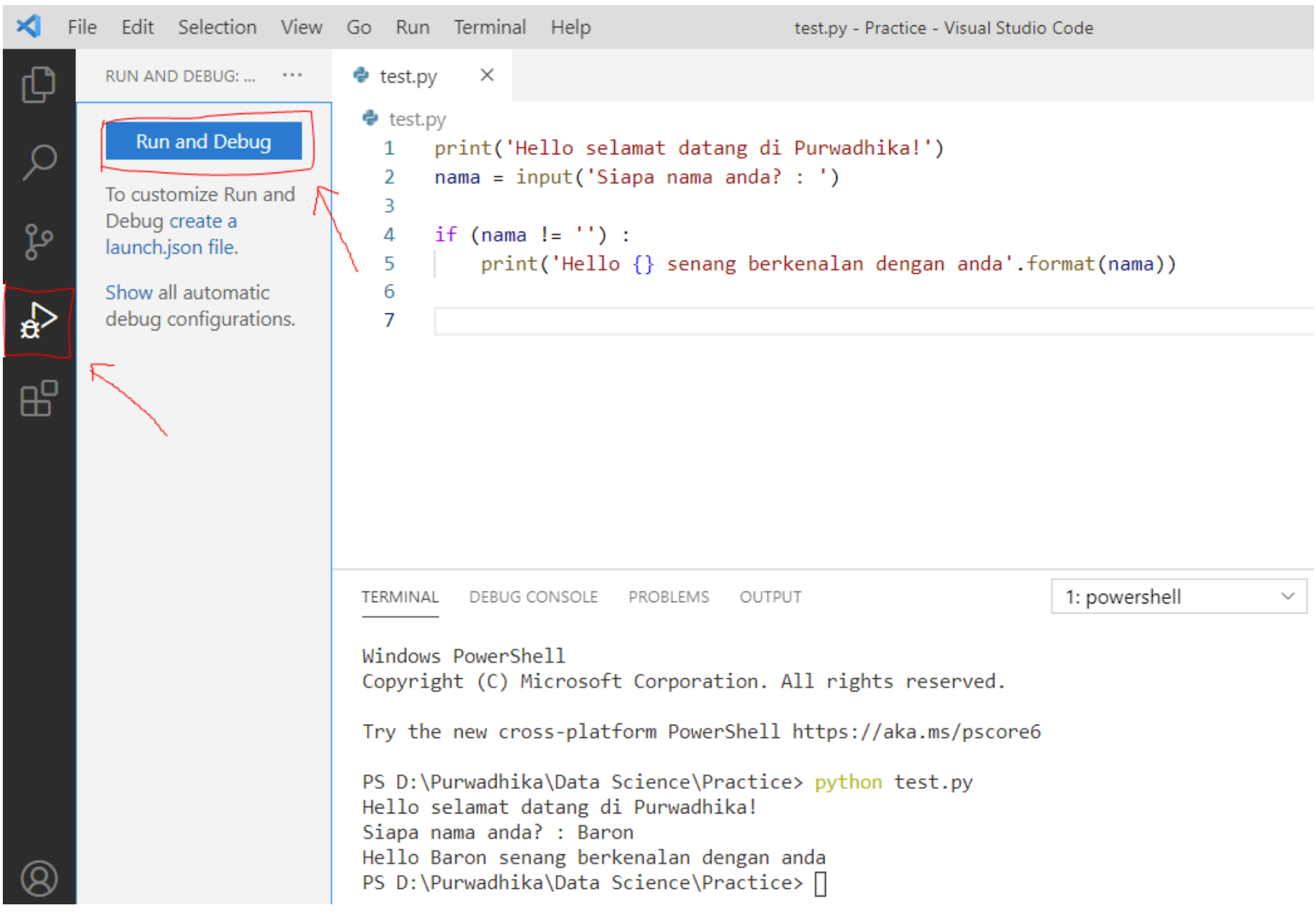
As we know, we can just run this code by simply type "python filename.py" in Terminal. The following image is the example of running the code using python keyword in vs code terminal.



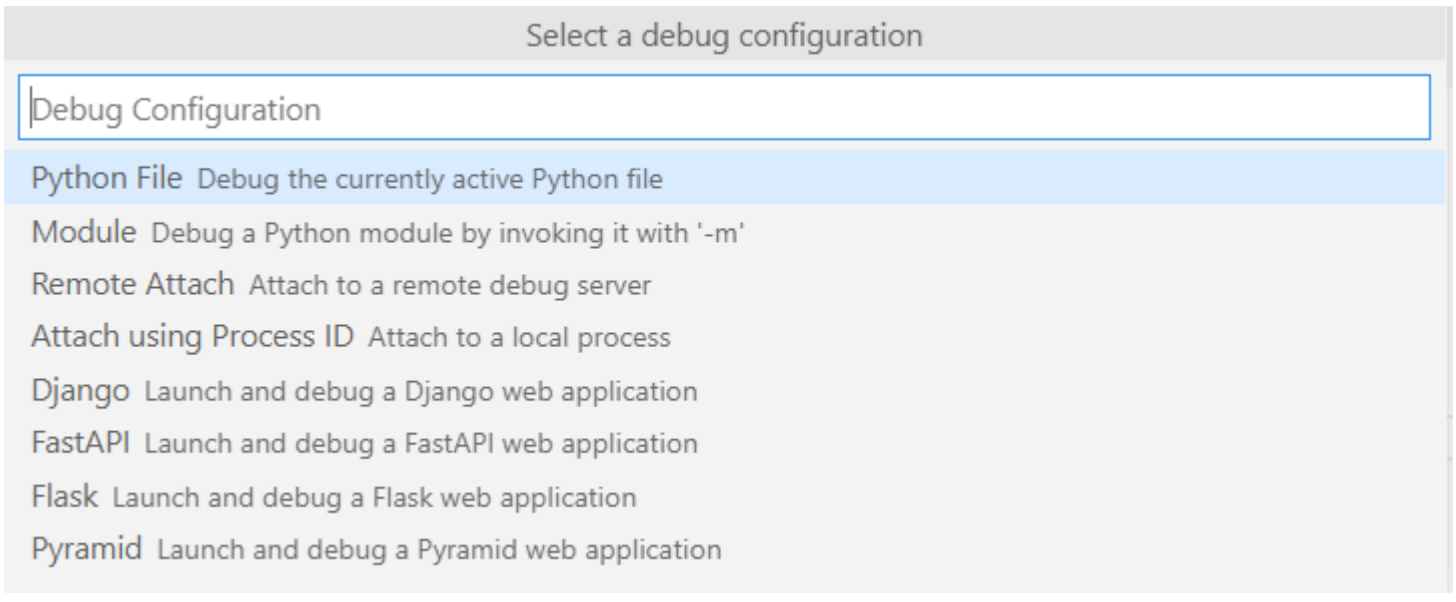
Now that we know our code is executed, we can go ahead and run the debugger to test the code and the execution flow as well. You can start the debugger in three ways.

- Select **Start Debugging** from the **Run** menu
- Selecting the **Run icon** from the left pane and click on **Run and Debug**
- Pressing **F5** on the keyboard

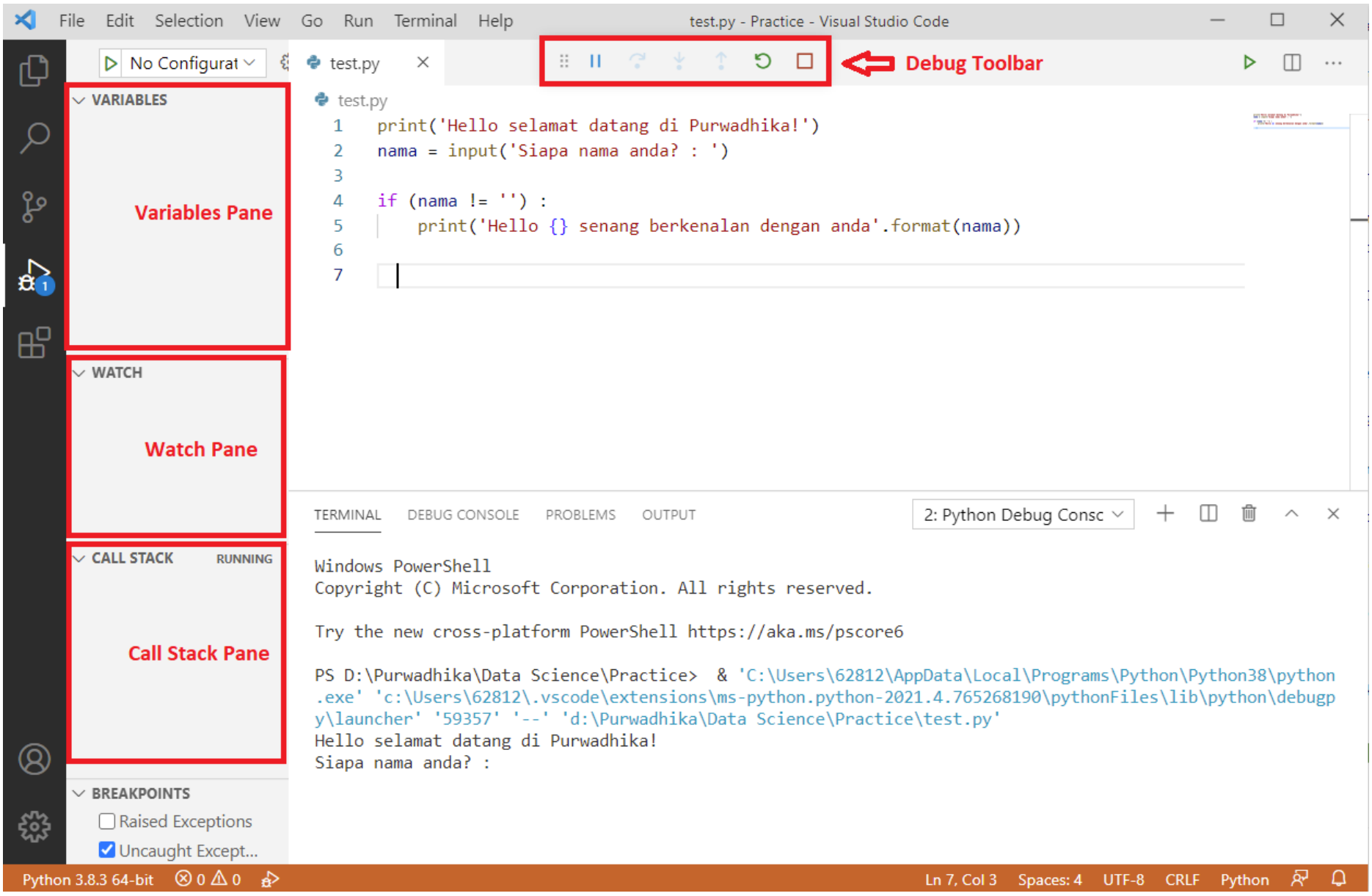
I would like to go with the second option and hit **Run and Debug**.



As soon as you hit the **Run and Debug** button, a popup will appear in VS Code which will prompt you to choose the **Debug Configuration** that you would like to use. Let us go ahead with the **Python File** option. You can select other debug configurations based on the application that you are working with.



As you select the debug configuration in Visual Studio Code, you can see that the Terminal window opens up from the bottom and the debugger starts executing your code.

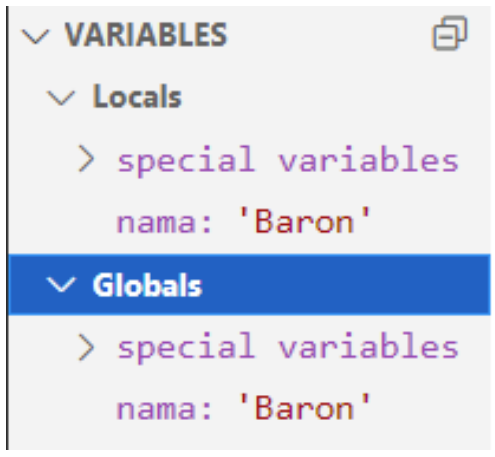


Let me quickly explain the various areas that you can see on the screen.

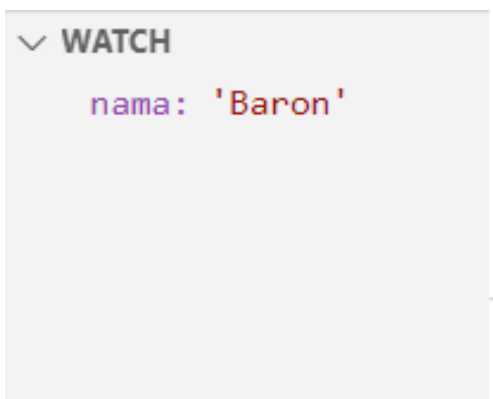
Debug Toolbar – This is the main toolbar that allows you to navigate along with your code as you try to debug it. There are six actions on this toolbar which are as follows:

- Continue/Pause (F5) – Allows you to pause or continue the debugging process
- Step Over (F10) – Allows you to move over to the next line of code
- Step Into (F11) – Allows you to enter inside a different method during debugging
- Step Out (Shift+F11) – Allows you to move to the parent stack
- Restart (Ctrl+Shift+F5) – Restarts the debugging session from the beginning
- Stop (Shift+F5) – Stops the debugging session

Variables Pane – Using the variables pane you can easily inspect the data elements within your program. When you start debugging a lot of system-defined variables are initiated along with the user-defined variables. During the debugging session, you can verify the values of each of those variables from this pane.



Watch Pane – Sometimes you may write a program with hundreds of variables within it. It is not possible to monitor the values of all those variables from the Variables pane as mentioned above. In such a case, you might want to monitor only one or two variables of your choice leaving the worry about the rest. You can add those variables to your watch list, and you can easily monitor the status and the values for those particular variables within this pane.



Call Stack Pane – This is helpful when your code has a lot of inner methods and you navigate deep inside a stack and then you might lose track of your stack. When there is any error in your program you can easily know from which stack is the error has occurred and then debug it accordingly.

Breakpoints – This is an important concept while debugging code in any language. You may notice that while you initially started the debugger session, the terminal just executed the script and the debugging session stopped automatically. What if you want to stop the debugging session at some particular line and monitor the variables closely? In such a case, the breakpoints come to our rescue. You can attach a breakpoint to any line by just click on the left of the line number. A red dot will appear which means that the breakpoint has been defined for that line. The next time when you debug, you can see that the program stops the execution at the line and waits for your command. You can now use the Debug Toolbar to navigate along with your code.

```
1  print('Hello selamat datang di Purwadhika!')
2  nama = input('Siapa nama anda? : ')
3
4  if (nama != '') :
5      print('Hello {} senang berkenalan dengan anda'.format(nama))
6
7
```

Conclusion

We have talked about the various topics on how to debug Python scripts in Visual Studio Code. As already discussed, Visual Studio Code is by far one of the simplest, yet powerful tools from Microsoft and used by many developers for the development of their applications. While writing complex applications, it often becomes a pain point to identify bugs in the existing system. In order to understand the complete flow and remove those bugs, debugging helps us to identify those pain points and understand them with more details about the execution flow of the application. Debugging should be considered as one of the core parts of any application development.