Software Engineering Lab #1

13th January 2022

Visual Studio Code as a Python Integrated Development Environment

Prerequisites

Note: Software packages required for this lab session have been installed on every PC in the lab ECC-706 already. In case you want to install these packages on your own machine, follow the instructions as follows.

Python Installation

- Download Python (the current version is 3.9.1) from http://www.python.org/
- For Windows platform, simply run the Python installer and follow the setup instruction.

Qt for Python Installation

- Download Qt for Python from https://doc.qt.io/qtforpython/gettingstarted.html by following the instruction here. Initially you need to get a file, for example, PySide6-6.14.0-6.14.0-cp36.cp37.cp38.cp39-none-win_amd64.whl.
- For Windows platform, download .whl for the 64 bit version and run the downloaded file using pip install command on DOS console window by issue the command in DOS "**python –m pip install filename.whl**". Make sure you refer to the directory properly for running python.exe (For example C:\Users\visit\AppData\Local\Programs\Python\Python37> is the directory of python.exe)
- The command to use, for example

  C:\Users\visit\AppData\Local\Programs\Python\Python37>python -m pip install pyside6

Microsoft Visual Studio Code Installation

- Download Microsoft Code from https://code.visualstudio.com/.
- From the Visual Studio Code installer, select the Python or Data Science workload to add Python support to Visual Studio.

Outline

1. Overview

The purpose of this lab is to learn how **to install and set up an integrated development environment (IDE) to run Python programs**. An IDE is an application for developing and debugging software applications. Here we use the **Visual Studio Code** as our Python integrated development environment.

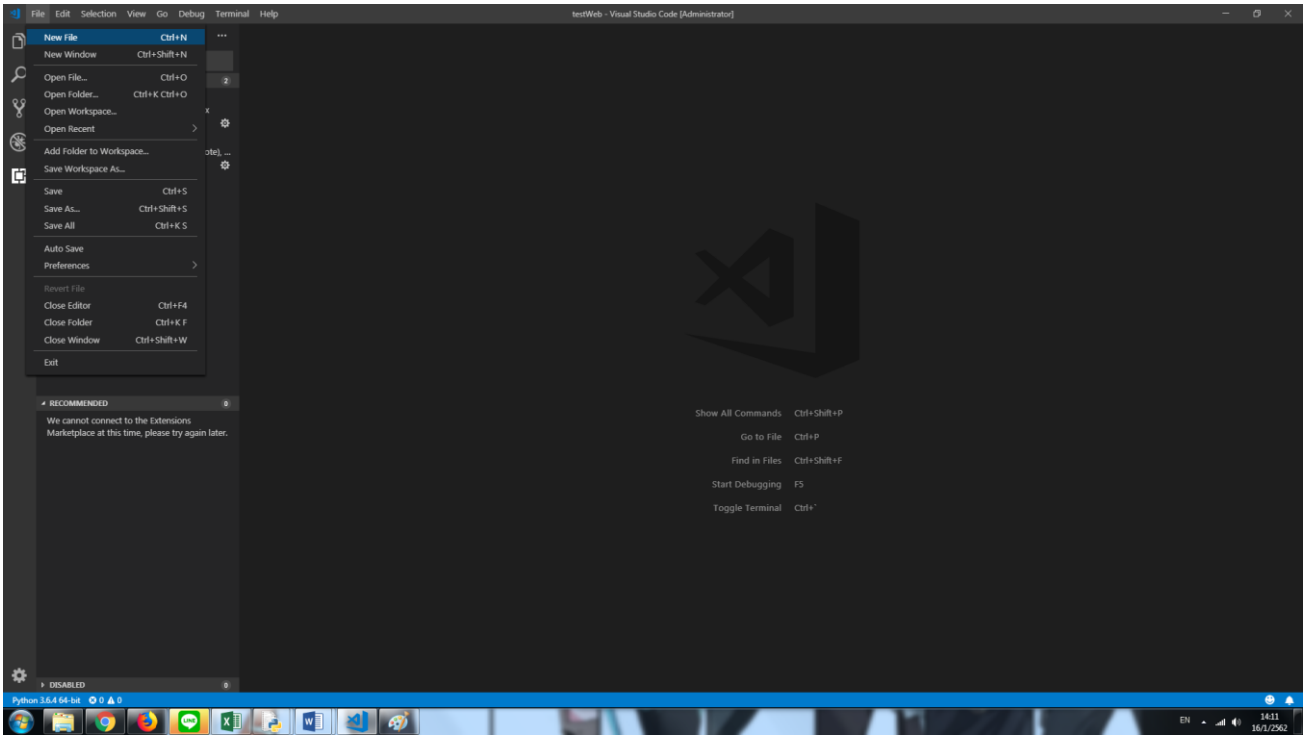
2. Objective

1) Understand the purpose of using an **integrated development environment (IDE).**
2) Learn how to install the Visual Studio IDE together with the Python plug-in.
3) Learn how to develop a simple Python application using Visual Studio Code.
4) Learn how to import existing Python code into a project.
5) Learn how to debug a Python application using Visual Studio Code.

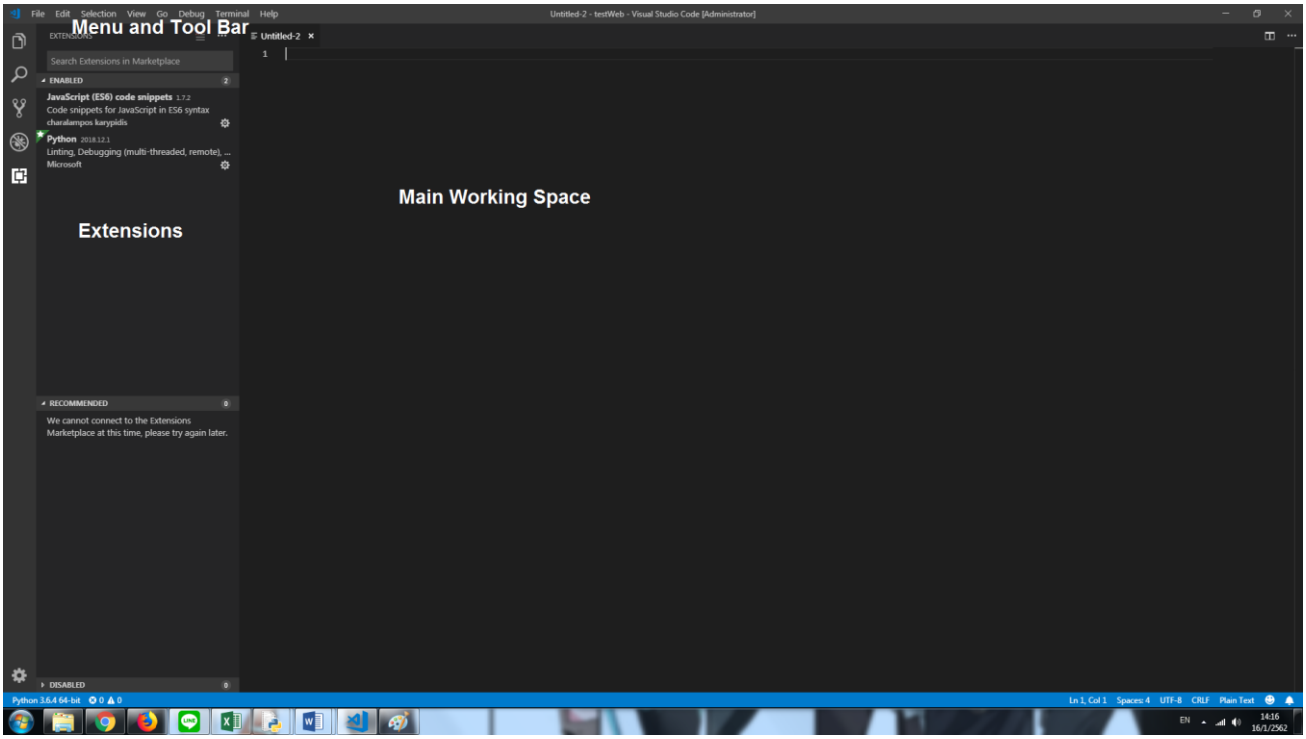
3. Create a simple application

Your first Python program in Visual Studio Code

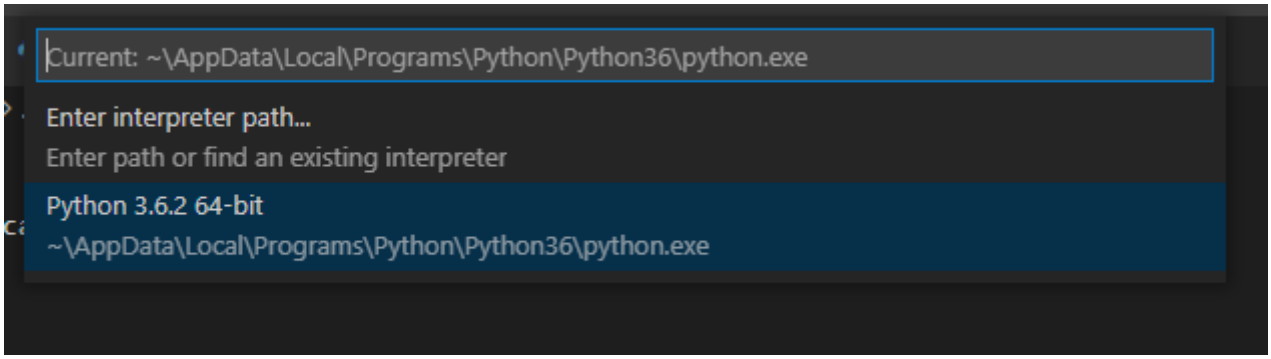
1. Select File -> New File.



2. IDE will create “Untitled” file for edit your program.



In case that, the machine has more than one version of Python interpreter. You can select which version to use in the IDE by clicking on the bottom left corner of the VS Code Window, that display version of Python for the Project.

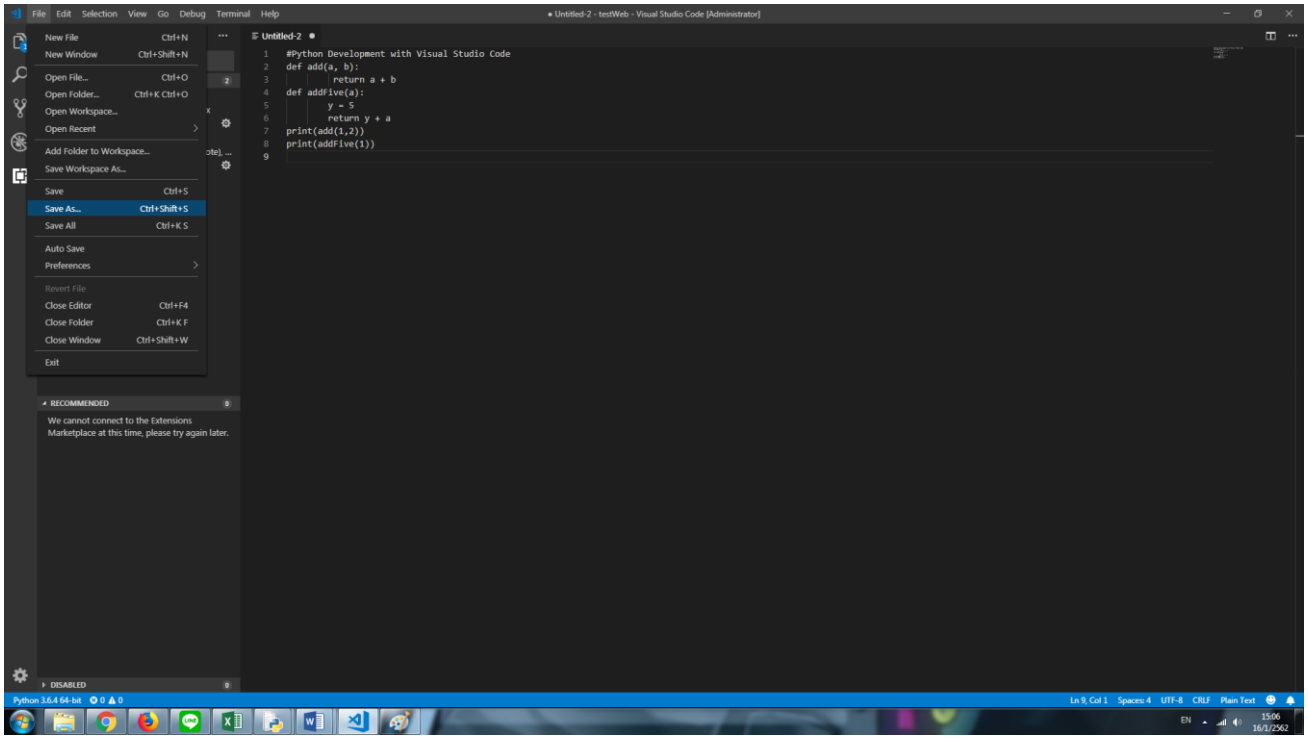


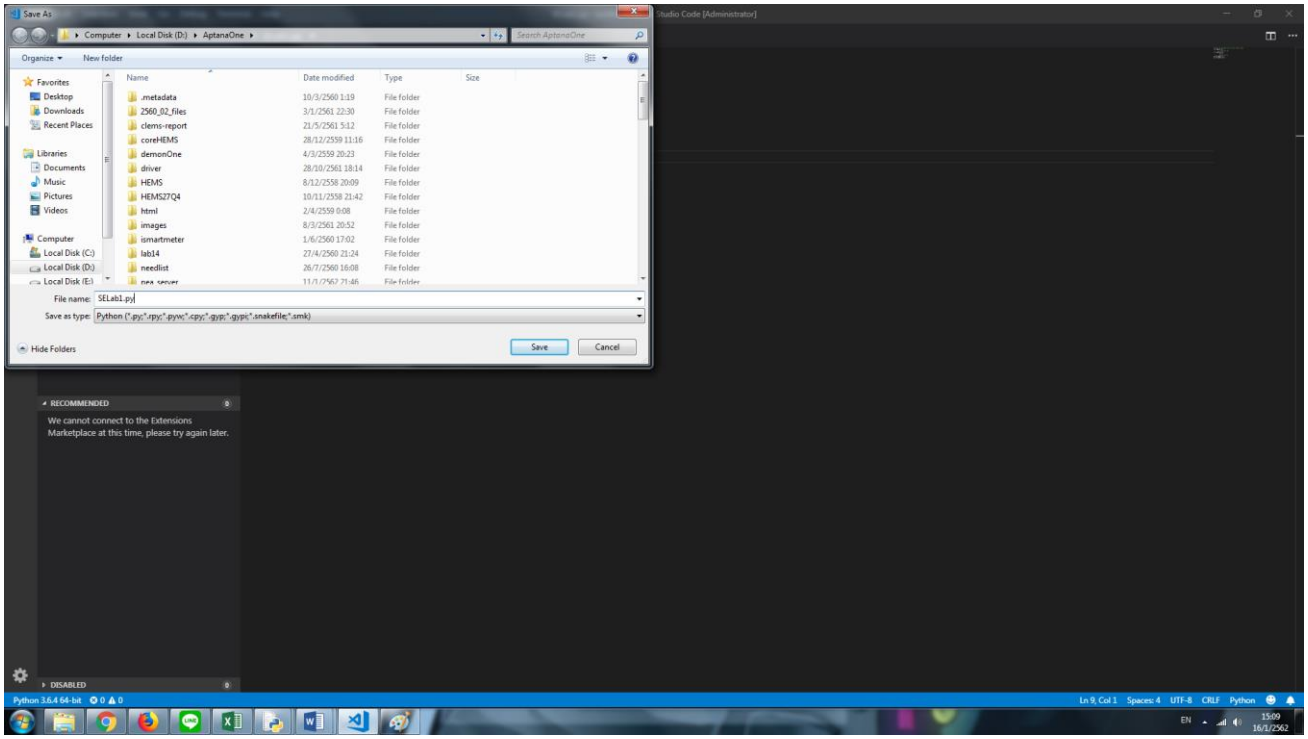
It will then display the list of python (interpreter) installed into the system, select one that suits your project.

3. Type the following Python codes in Main Working space area.

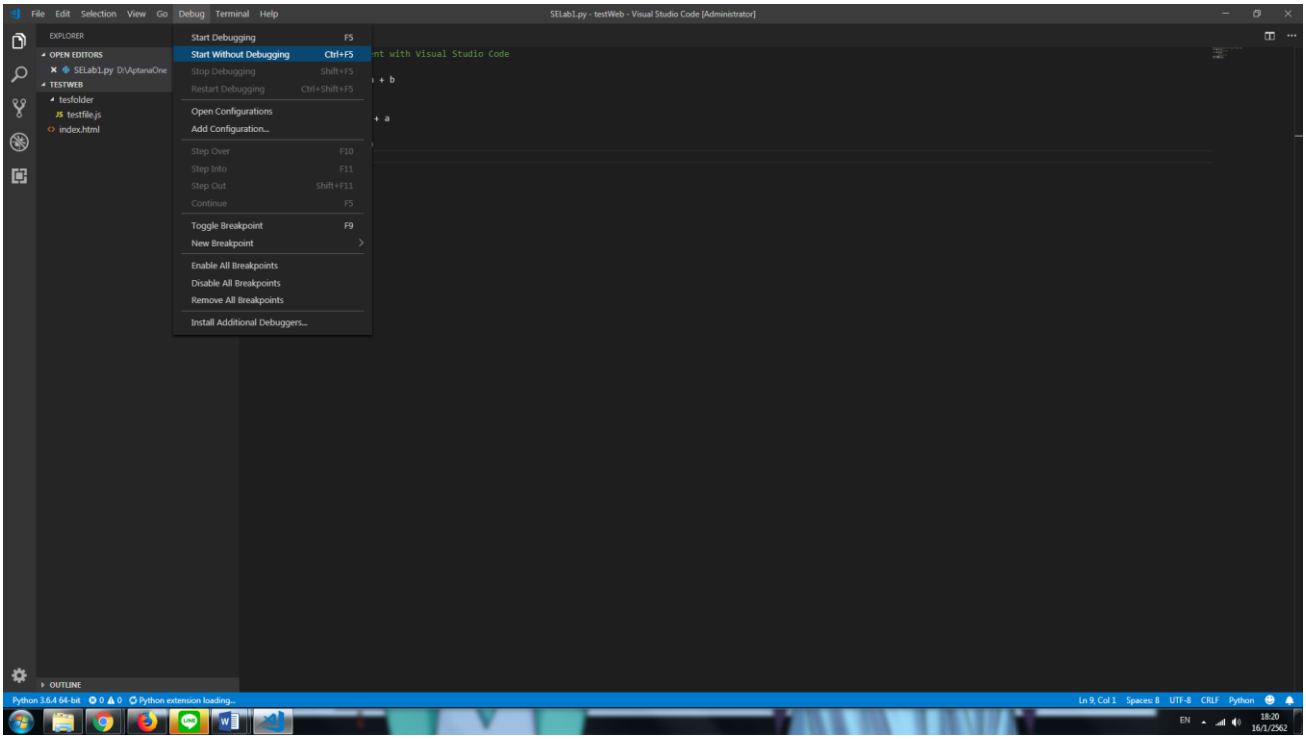
```
1  #Python Development with Visual Studio Code
2  def add(a, b):
3      return a + b
4  def addFive(a):
5      y = 5
6      return y + a
7  print(add(1,2))
8  print(addFive(1))
```

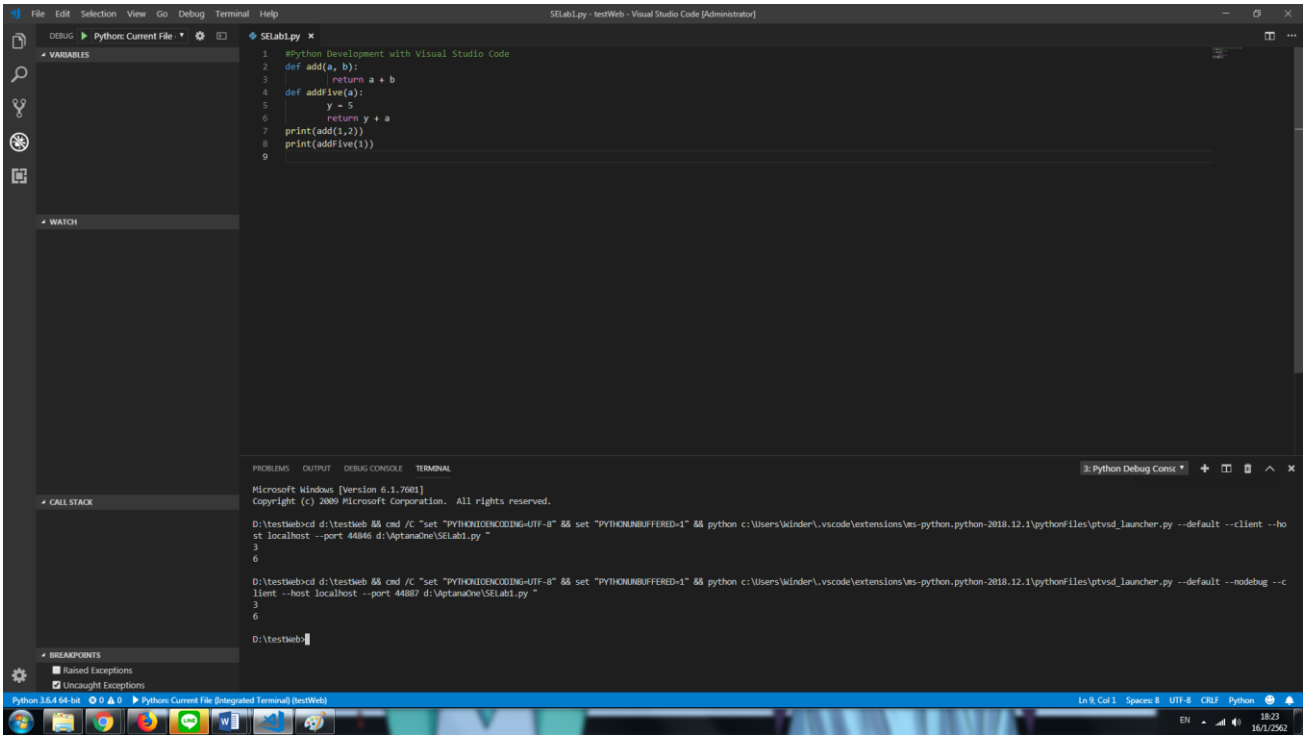
4. Click File >>> Save As…. Save File with name “SELab1.py”





5. Click at Debug and select run Start without Debugging.

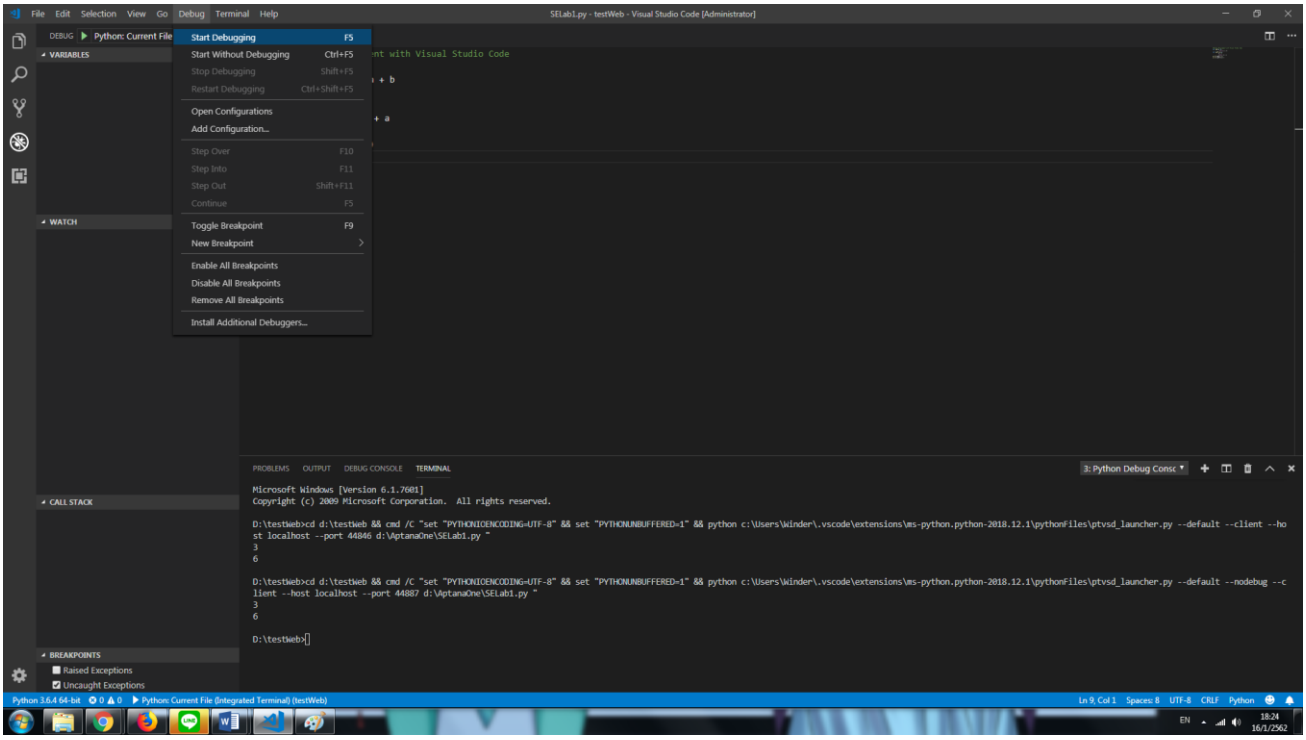




6. Congratulations! You have created your first Python module and run it!

4. Debug and test your application

1. Start the debugger by selecting Debug, then Start Debugging.



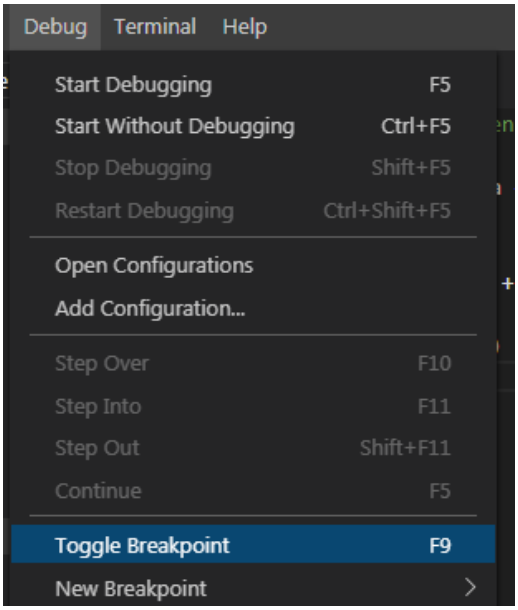
2. To debug with breakpoints, right-click in the source code where you want to add a breakpoint and then add a **breakpoint**.

- By adding some breakpoints, you can test the code during debugging.

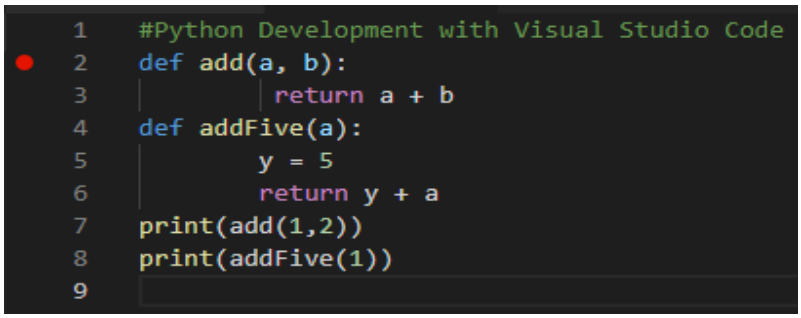
- You can add breakpoints by choosing Debug, Toggle Breakpoint on the menu bar or by clicking in the left margin of the editor next to the line of code where you want a breakpoint to be added.

To add breakpoints

1. Select the following line: **def add(a, b):**
2. Add a breakpoint from the menu by selecting **Debug**, then **Toggle Breakpoint**.



A red dot appears in front of the line of code in the editor window.



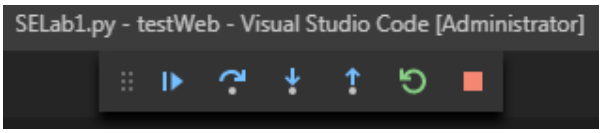
3. Choose the F9 key to add a breakpoint, and then choose the F5 key to start debugging.
4. At the bottom of the IDE, the Autos, Locals, and Watch windows are docked together on the left side, and the Call Stack, Breakpoints, Command, Immediate, and Output windows are docked together on the right side.
5. On the menu bar, choose Debug, Step Out.
6. Choose the F5 key to continue debugging. When the message box appears, choose the OK button on the message box to close it.
7. Choose the SHIFT + F5 keys to stop debugging.
8. On the menu bar, choose Debug, disable all breakpoints.

Once you've broken into the debugger already, you can step through your code one statement by statement. The step commands include **Step Into, Step Over, and Step Out**, please see Table 1 below for more details.

Table 1. Debugging Key bindings

Command	Description
F11	**Step Into** will execute the next statement and stop. If the next statement is a call to a function, the debugger will stop at the first line of the function being called. Check the Debug, Step Into menu to find the keyboard shortcut to use to step (typically F11).
F10	**Step Over** will execute the next statement, but if it is a call to a function then the entire function will be executed. This allows you to easily skip functions when you are not interested in debugging them. Check the Debug, Step Over menu to find the keyboard shortcut (typically F10).
Shift + F11	**Step Out** will execute until the end of the current function. It is useful when there is nothing else interesting in the current function. Check the Debug, Step Out menu to find the keyboard shortcut (typically Shift+F11).

- If you want to continue running, press F5. Your program will not break until it reaches the next breakpoint. Note that when you Step Over or Step Out, if the running code hits a breakpoint it will break again, even if it has not reached the end of the function.

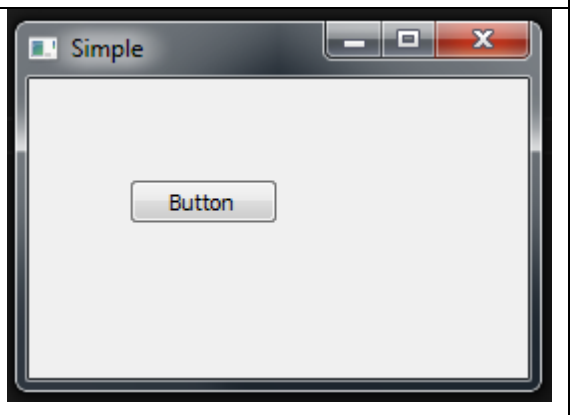


You can always use the GUI in the menu to do debugging. The following displays the key bindings for the debug buttons.

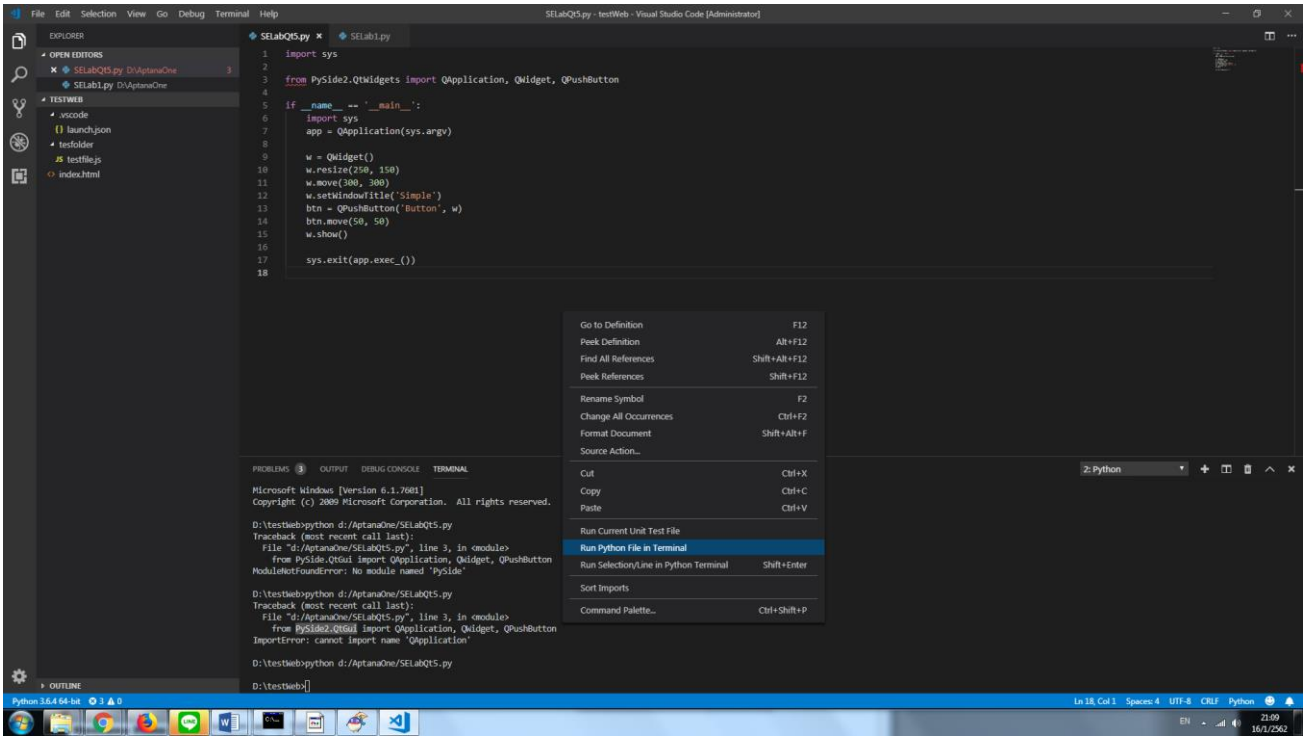
5. Create a simple application

Program 1.1: Your first PTVS program

Lines of Code	Source Code	Output

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
	`import sys`
	`from PySide6.QtWidgets import`
	`QApplication, QWidget, QPushButton`
	
	`if __name__ == '__main__':`
	`  import sys`
	`  app = QApplication(sys.argv)`
	
	`  w = QWidget()`
	`  w.resize(250, 150)`
	`  w.move(300, 300)`
	`  w.setWindowTitle('Simple')`
	`  btn = QPushButton('Button', w)`
	`  btn.move(50, 50)`
	`  w.show()`
	`  sys.exit(app.exec_())`
		

1. Create New Project and type the following source code in Main windows space area as follows.



2. Run this program by right–clicking at Run Python file in Terminal or press F5. The Result will display as follows.