

# Getting Productive with Python and Visual Studio Code

---



# Development Environment construction

## Prerequisites

- Python installation
- Visual Studio Code (VS code) installation
- VS code for python extension



# Python installation

- <https://www.python.org/>
- Downloads⇒Python 3.10.6

The screenshot shows the Python.org homepage with a navigation bar at the top. The 'Downloads' tab is highlighted with a red oval. A large red arrow points from the 'Downloads' tab to the 'Python 3.10.6' download button on the right. The 'Downloads' section includes links for All releases, Source code, Windows, macOS, Other Platforms, License, and Alternative Implementations. On the right, there is a note about Python 3.9+ compatibility and a link to view the full list of downloads.

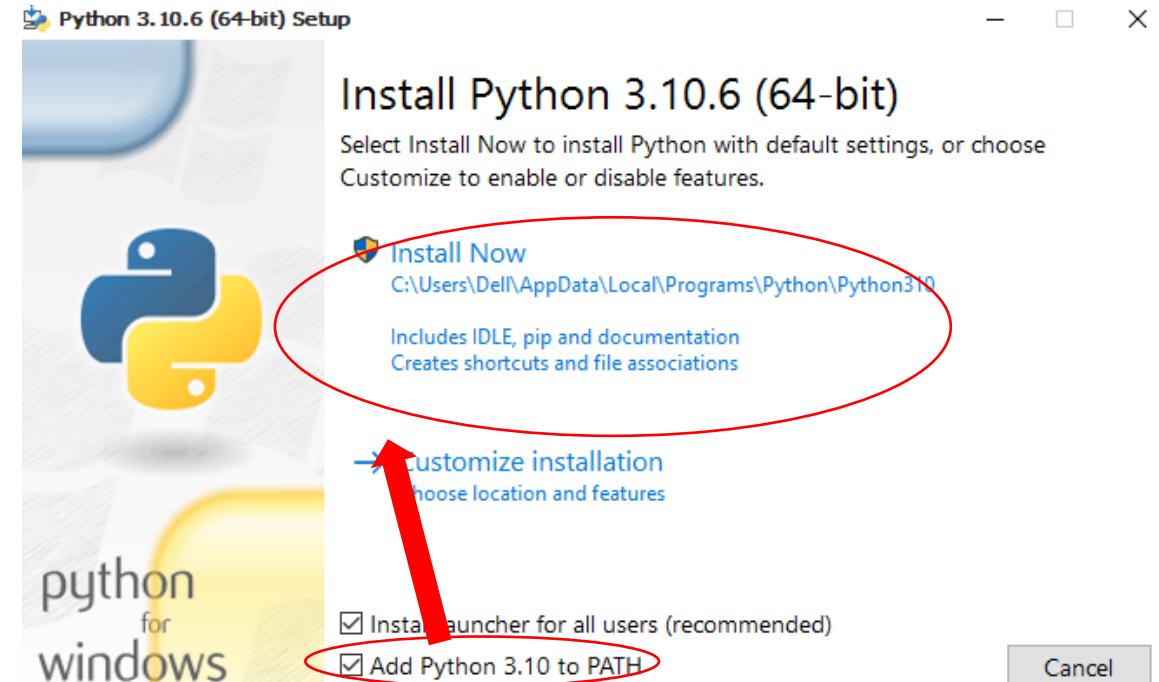
- Run the downloaded installer



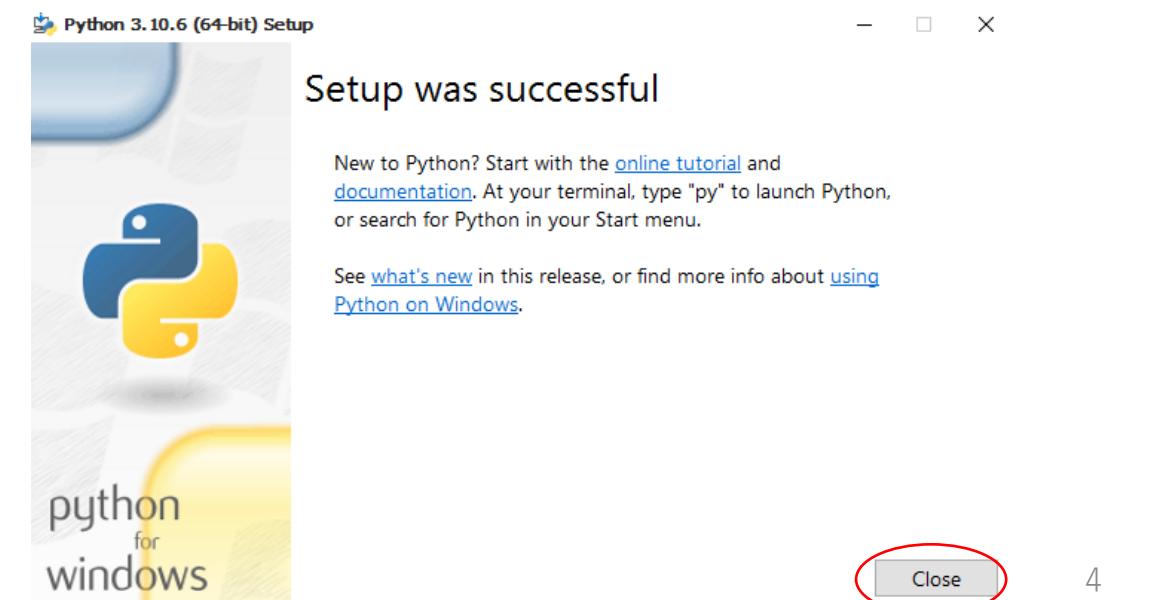


# Python installation

- Check on “Add Python 3.10 to PATH”
- Click “Install Now”



- Click “Close”



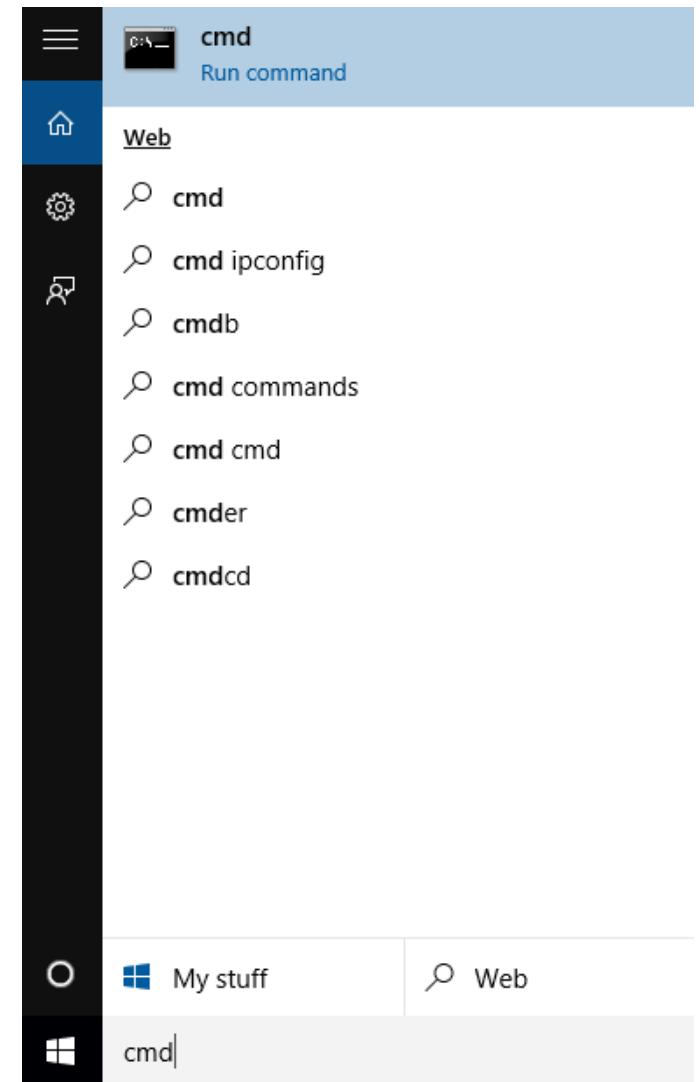


# Python installation (Confirmation)

- After installation, enter “cmd” in the start menu
- Open command prompt
- Enter “python –V” to check python version

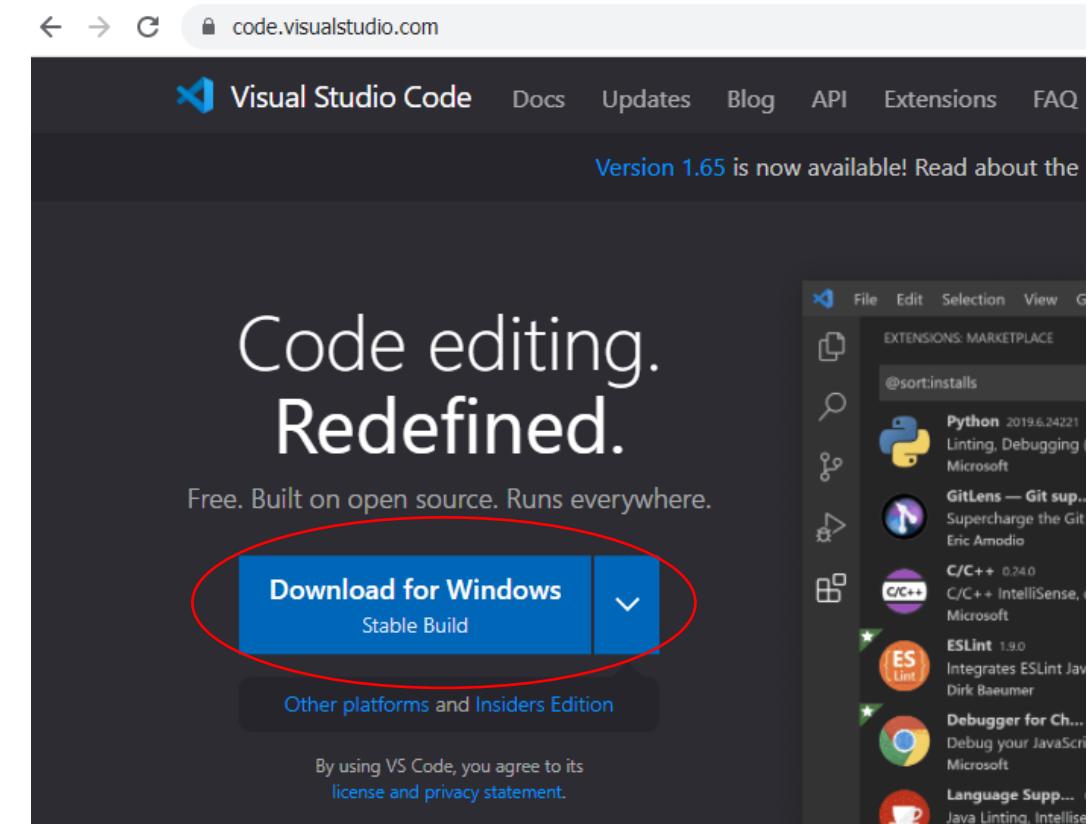
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Dell>python -V
Python 3.10.6
C:\Users\Dell>
```



# Visual Studio Code Installation

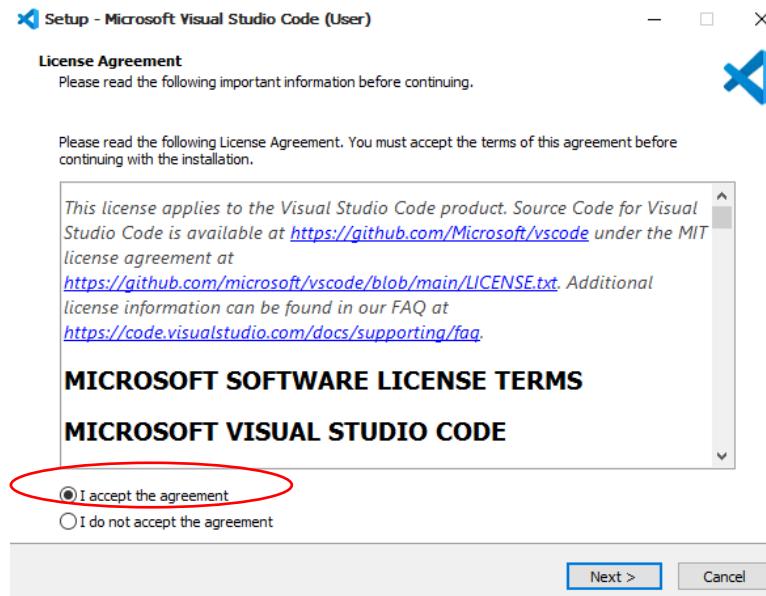
- <https://code.visualstudio.com/>
- Click the Download for Windows
- Run the installer



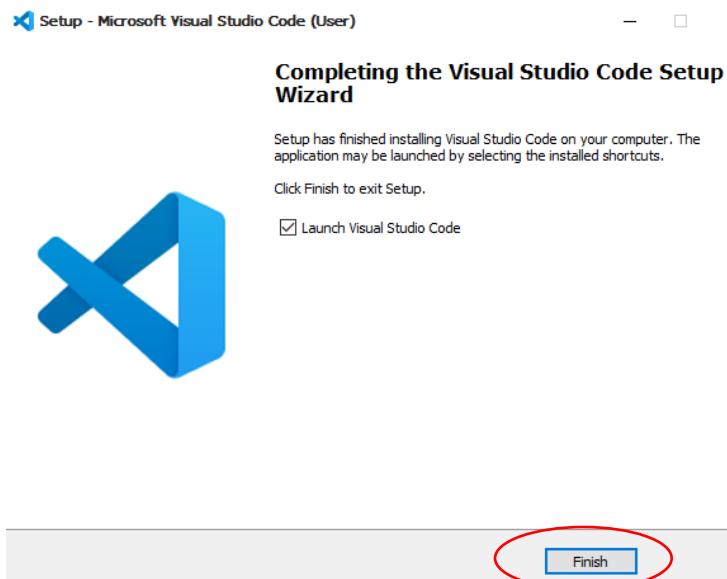


# Visual Studio Code Installation

- Choose “I accept the agreement”

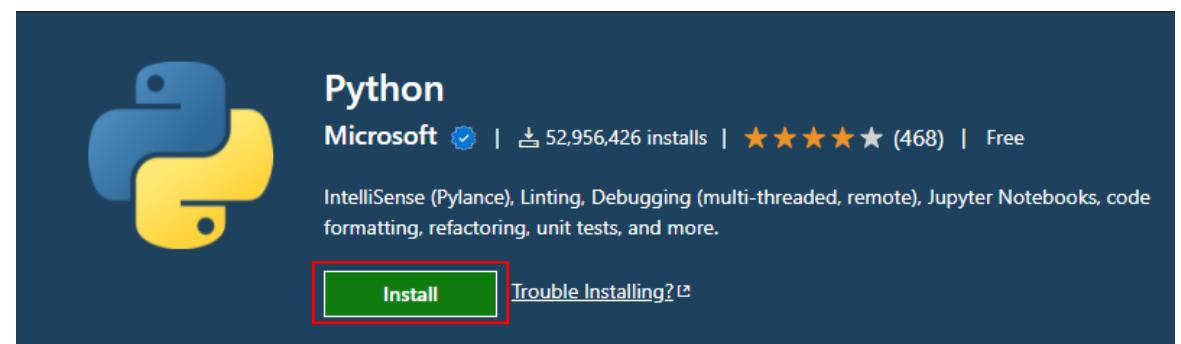
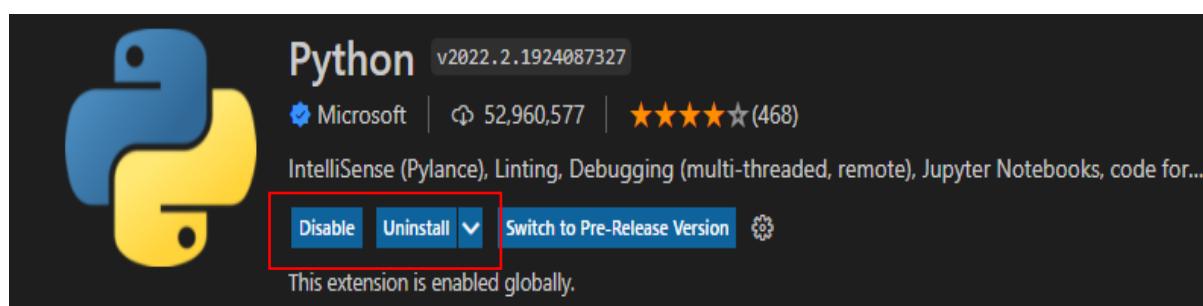
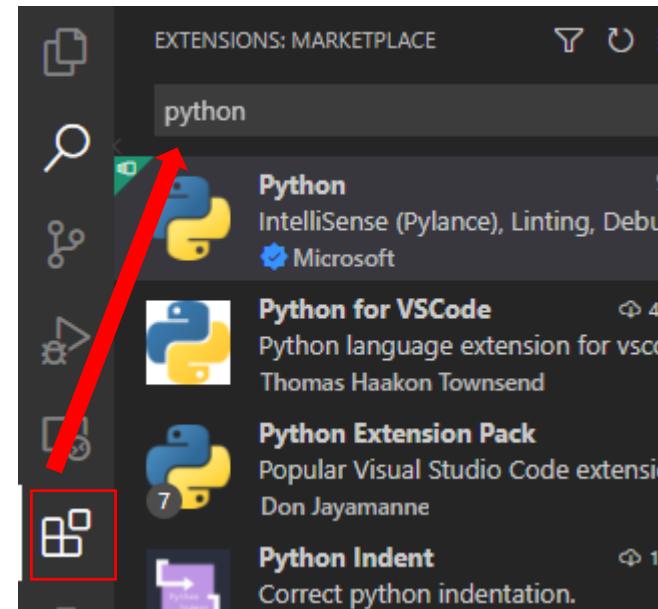


- Click “Next” button to continue installation



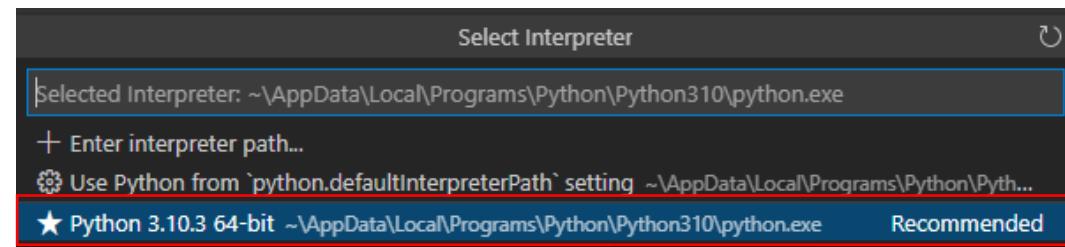
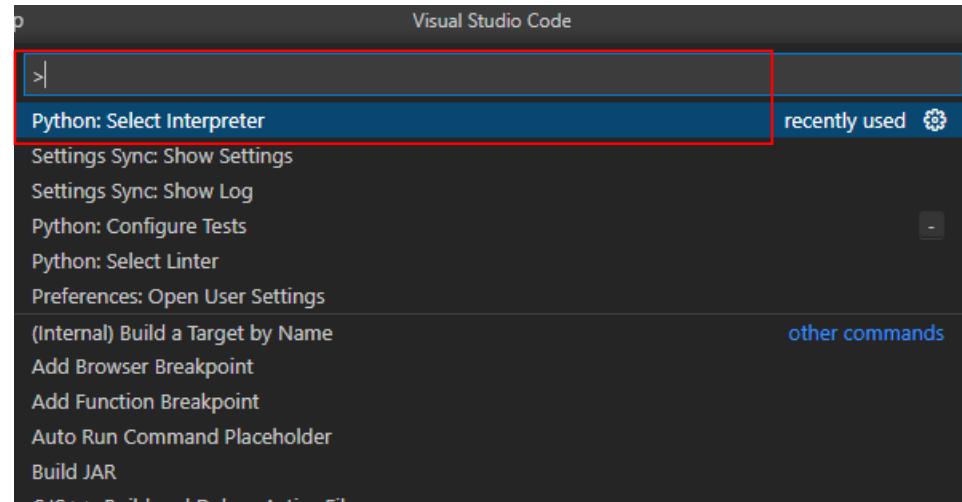
# Python Extensions (python installation)

- Open the VS code IDE
- See EXTENSION MARKETPLACE for installing extensions
- Enter "python" in the search box
- Click Install
- After installation, check the installed extensions.



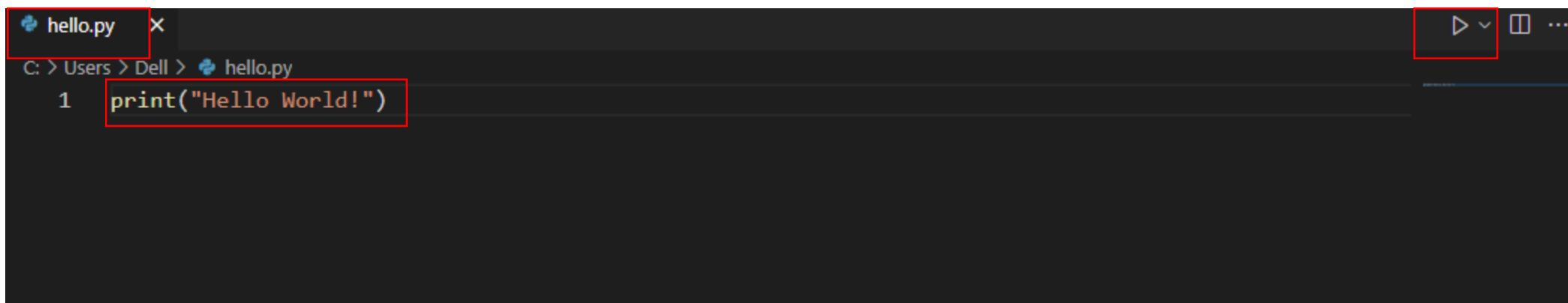
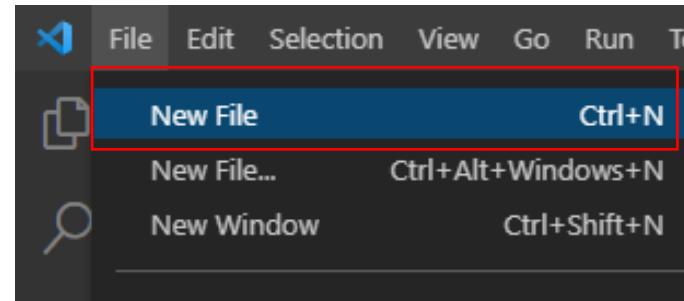
# Python Interpreter for VS code

- To run Python code and get Python IntelliSense, you must tell VS Code which interpreter to use.
- Command Palette (Ctrl+Shift+P), then type **Python: Select Interpreter**
- Select the Recommended Interpreter



# Example : Hello World

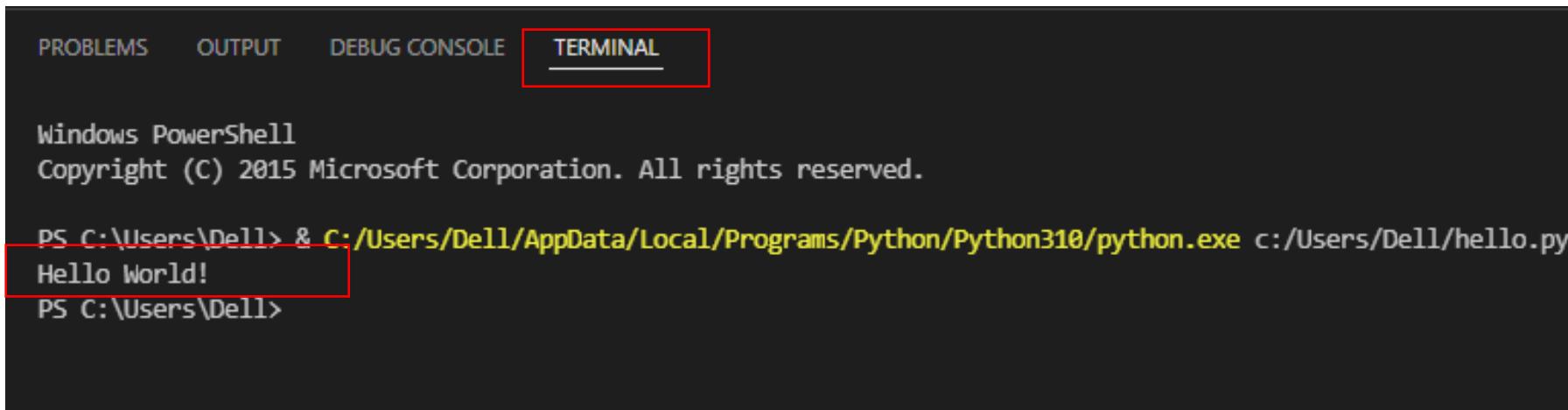
- Click on new file
- Save as “Hello.py”
  - `print("Hello World!")`
- Press the run button on the top right to run python code



```
hello.py
C: > Users > Dell > hello.py
1 print("Hello World!")
```

# Example : Hello World

- Hello World! appears in the terminal below .
- Congratulations, you have created your first Python program.



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are four tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with the TERMINAL tab being the active one and highlighted with a red border. Below the tabs, the terminal displays the following text:

```
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\Dell> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Dell/hello.py
Hello World!
PS C:\Users\Dell>
```

The command `& C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Dell/hello.py` and its output `Hello World!` are highlighted with a red rectangular box.

# Research Participation

- Download the folders in the following link.

## Python\_CWP\_Libraries

[https://drive.google.com/file/d/1qi9aNMrZrkfPZFOz3caL53sRlpwNdWkj/view?usp=share\\_link](https://drive.google.com/file/d/1qi9aNMrZrkfPZFOz3caL53sRlpwNdWkj/view?usp=share_link)

## Python\_CWP\_UI

[https://drive.google.com/file/d/1wh7I7GDw-y5K4fcadymwp8QW6AaYLiXw/view?usp=share\\_link](https://drive.google.com/file/d/1wh7I7GDw-y5K4fcadymwp8QW6AaYLiXw/view?usp=share_link)

- You can see in **details** how to solve the code writing problem in next slides.

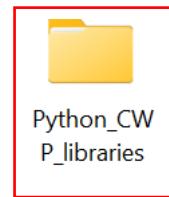
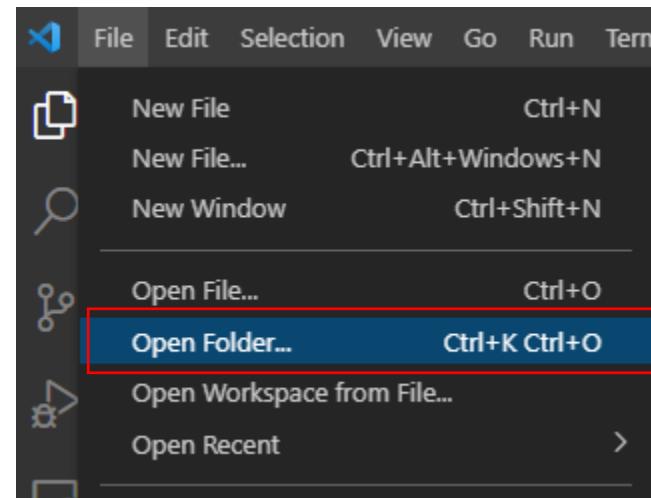
# 1. Download the folder in the following link

[Python\_CWP\_Libraries]

[https://drive.google.com/file/d/1qi9aNMrkPZFOz3caL53sRlpwNdWkj/view?usp=share\\_link](https://drive.google.com/file/d/1qi9aNMrkPZFOz3caL53sRlpwNdWkj/view?usp=share_link)

# 2. Open the downloaded folder in VS code.

- In each folder, there are **source code, test code** python files.



A screenshot of the VS Code interface showing the file structure in the Explorer panel. A folder named 'p1\_Length' is selected and highlighted with a red box. It contains two files: 'Length.py' and 'LengthTest.py'. The main editor pane shows the content of 'Length.py', which contains a single line of code: 'class length: pass'. The status bar at the bottom indicates the file is 15 lines long.

```
class length:
    pass
```

# Source Code

```
p1_Length > Length.py > ...
1  class length:
2
3      pass
4
5      # using len()
6      def str_length1(x):
7          result = len(x)
8          return result
9
10     # without using len()
11     def str_length2(string):
12         string_length = 0
13         for i in string:
14             string_length += 1
15         return (string_length)
```

# Test Code

```
p1_Length > LengthTest.py > main
1  # A library function of length
2  import unittest,sys,os
3  import inspect
4  from Length import length
5
6  class Test_StrLength(unittest.TestCase):
7      # use len()
8      def test_str_length1(self):
9          source_code = inspect.getsource(length.str_length1)
10         self.assertEqual(length.str_length1('Hello'), 5)
11         self.assertIn("len()", source_code)
12
13     # without use len()
14     def test_str_length2(self):
15         source_code = inspect.getsource(length.str_length2)
16         self.assertEqual(length.str_length2('Hello'), 5)
17         self.assertNotIn("len()", source_code)
18
```

```
# Output the result file
def main(out=sys.stderr, verbosity=2):
    loader = unittest.TestLoader()
    suite = loader.loadTestsFromModule(sys.modules['__name__'])
    unittest.TextTestRunner(out, verbosity=verbosity).run(suite)

if __name__ == '__main__':
    path1 = os.getcwd()
    path2 = "results"
    completeName = os.path.join(path1, path2, "p1.result")
    with open(completeName, 'w') as f:
        main(f)

# run the unittest
if __name__ == '__main__':
    unittest.main()
```

# Output the result file and # run the unittest are not considerable  
in writing source code

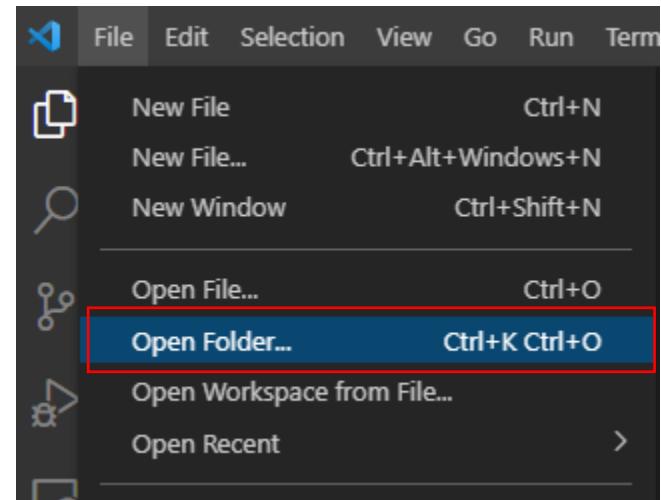
1. Download the folders in the following link

[Python\_CWP\_UI]

[https://drive.google.com/file/d/1wh7I7GDwy5K4fcadymwp8QW6AaYLiXw/view?usp=share\\_link](https://drive.google.com/file/d/1wh7I7GDwy5K4fcadymwp8QW6AaYLiXw/view?usp=share_link)

2. Open the downloaded folder in VS code.

- In each folder, there are **source code, test code** python files, **output.png** and reference note.



A screenshot of the VS Code interface showing the 'EXPLORER' view with the 'PYTHON\_CWP\_UI' folder expanded. Inside, files like 'Label.py', 'LabelTest.py', 'note.md', and 'Output.png' are listed. A red box highlights this list. To the right, the 'LABEL.PY' tab is active, displaying the following Python code:

```
1 import tkinter as tk
2
3 class Label:
4     pass
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

# Source Code

```
p1_Label > Label.py > ...
1 import tkinter as tk
2
3 class Label:
4
5     pass
6
7     def label_program():
8
9         root = tk.Tk()
10        root.title("Hello World")
11
12        # Create a label with red text color
13        label = tk.Label(root, text="Hello, World!", fg="red")
14        label.pack()
15
16        return root, label
17
```

# Test Code

```
p1_Label > LabelTest.py > main
1 import unittest,sys,os
2 import tkinter as tk
3 from Label import Label
4
5 class Label_Program(unittest.TestCase):
6     def test_Label_Program(self):
7         root, label = Label.label_program()
8
9         # test root window properties
10        self.assertIsInstance(root, tk.Tk)
11        self.assertEqual(root.winfo_exists(), 1)
12        self.assertEqual(root.title(), "Hello World")
13
14        # test label properties
15        self.assertIsInstance(label, tk.Label)
16        self.assertEqual(label['text'], "Hello, World!")
17        self.assertEqual(label['fg'], "red")
```

- See the **Test Code** then write Source code under pass keyword.
- See the **output.png** file to easily to understand program output.
- See **note** file for Reference and Program Description.

```
# output the result file
def main(out=sys.stderr, verbosity=2):
    loader = unittest.TestLoader()
    suite = loader.loadTestsFromModule(sys.modules[__name__])
    unittest.TextTestRunner(out, verbosity=verbosity).run(suite)

if __name__ == '__main__':
    path1 = os.getcwd()
    path2 = "results"
    completeName = os.path.join(path1, path2, "p1.result")
    with open(completeName, 'w') as f:
        main(f)

# run the unittest
if __name__ == '__main__':
    unittest.main()
```

# Output the result file and # run the unittest are not considerable in writing source code

After finishing the source code, you can **check source code** by running the test code. See the results in the terminal or “results” folder.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files and folders under the 'PYTHON\_CWP\_UI' workspace. A red box highlights the 'results' folder, which contains a file named 'p1.result'. The Terminal tab at the bottom is active, showing command-line output. Another red box highlights the 'TERMINAL' tab. The status bar at the bottom indicates the current file path and the Python interpreter being used.

```
EXPLORER ... p1.result X Label.py LabelTest.py Output.png note.md  
PYTHON_CWP_UI  
p1_Label  
> __pycache__  
Label.py  
LabelTest.py  
note.md  
Output.png  
p2_Button  
p3_TextBox  
p4_Frame  
p5_Canvas  
p6_RadioButton  
p7_Scale  
p8_CheckBox  
p9_ComboBox  
p10_ListBox  
results  
p1.result  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
PS C:\Python_CWP_UI> & C:/Users/shune/AppData/Local/Microsoft/WindowsApps/python3  
.  
Ran 1 test in 0.060s  
OK
```

After completing ,

- Copy all source code files.
- Copy the ‘results’ folder.
- Send email to ...
- p1kl27uw@s.okayama-u.ac.jp