Table of Contents

Lab 1: Introduction to Python

Learning Outcomes

In this lab, you will:

- Install Python 3 and learn the basics of creating scripts
- Learn how to interact with the Python interpreter
- Learn how to use `print()` to print messages to standard output
- Learn about storing data in variables
- Learn how to print the contents of variables
- Learn how to get user input using `input()` and store it into variables

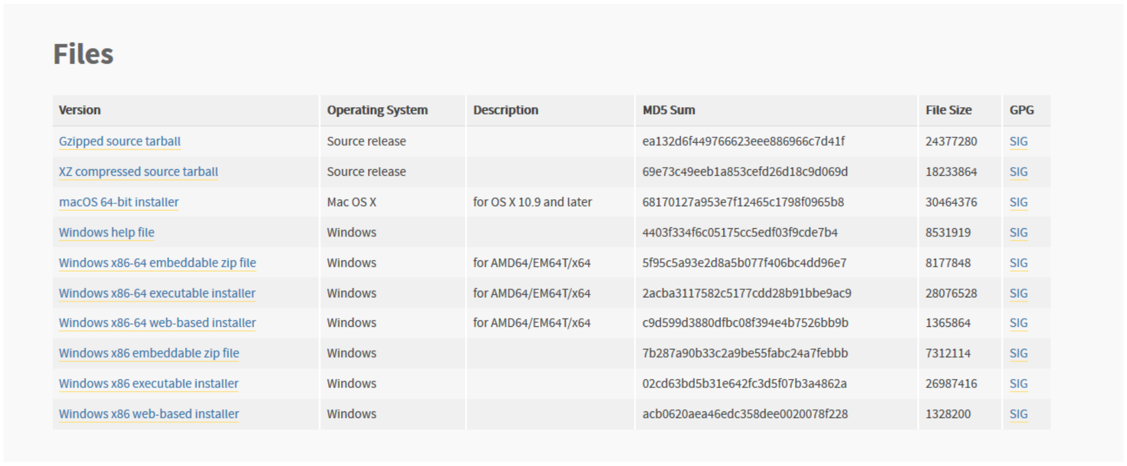
Introduction

Python is a programming language which has gradually become one of the most popular languages in the world. It's not hard to see why: the strengths of Python lie in its accessibility and its large standard library. This means that non-programmers often choose Python as their first language to learn. Its *syntax* (that is, the keywords and symbols used to define how code works) will remind you of plain English, and allow you to focus on grasping some of the fundamental concepts of programming rather than on punctuation.

1.1 Installing Python 3 on Windows

Python 3 was released in 2008. One important thing to note is that there are large differences between Python 2 and Python 3, so we will need to make sure that we aren't using the older version of Python.

1. Go to Python Downloads and select the latest version of Python. At the time of writing, this is **Python 3.8**.
2. You should see a list of possible files to download. Select **Windows x86-64 executable installer**.

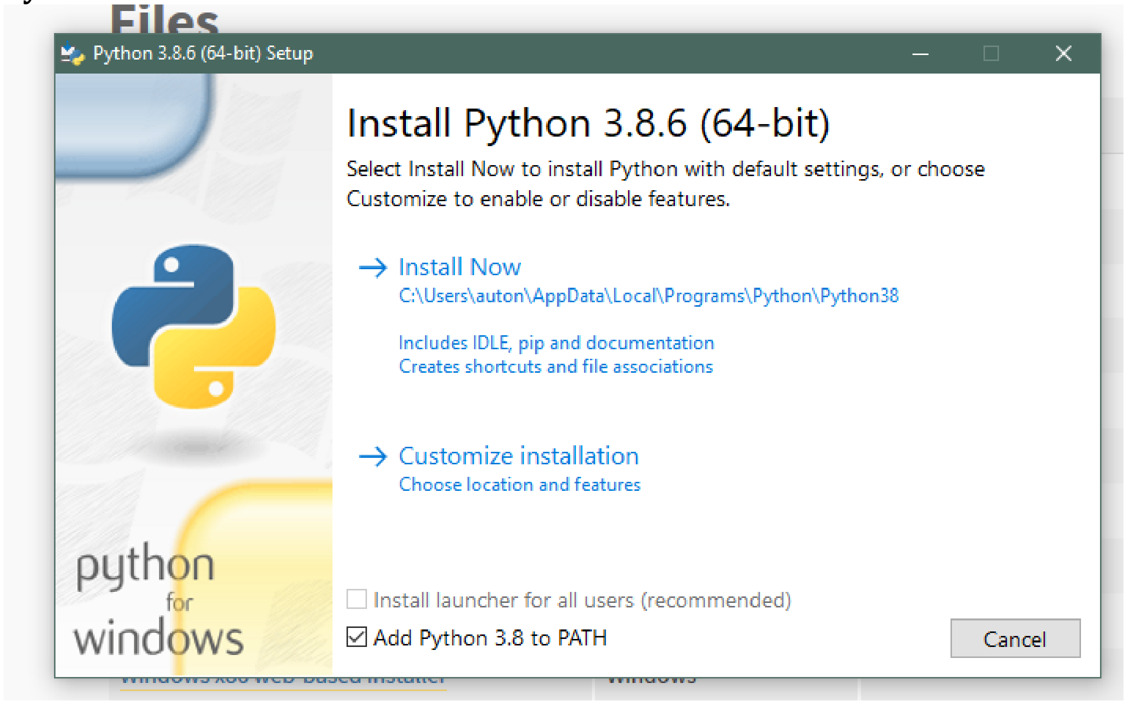


Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		ea132d6f449766623eee886966c7d41f	24377280	SIG
XZ compressed source tarball	Source release		69e73c49eeb1a853cefd26d18c9d069d	18233864	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	68170127a953e7f12465c1798f0965b8	30464376	SIG
Windows help file	Windows		4403f334f6c05175cc5edf03f9cde7b4	8531919	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	5f95c5a93e2d8a5b077f406bc4dd96e7	8177848	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	2acba3117582c5177cdd28b91bbe9ac9	28076528	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	c9d599d3880dfbc08f394e4b7526bb9b	1365864	SIG
Windows x86 embeddable zip file	Windows		7b287a90b33c2a9be55fabc24a7febbb	7312114	SIG
Windows x86 executable installer	Windows		02cd63bd5b31e642fc3d5f07b3a4862a	26987416	SIG
Windows x86 web-based installer	Windows		acb0620aea46edc358dee0020078f228	1328200	SIG

3. Accept default values. This should install a tool called *pip* and a text editor called *Idle*. You are free to use a different a different text editor if you wish. Also **check "Add**

Python 3.8 to PATH".


Python 3.8.6 (64-bit) Setup

Install Python 3.8.6 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

→ Install Now
C:\Users\auton\AppData\Local\Programs\Python\Python38

Includes IDLE, pip and documentation
Creates shortcuts and file associations

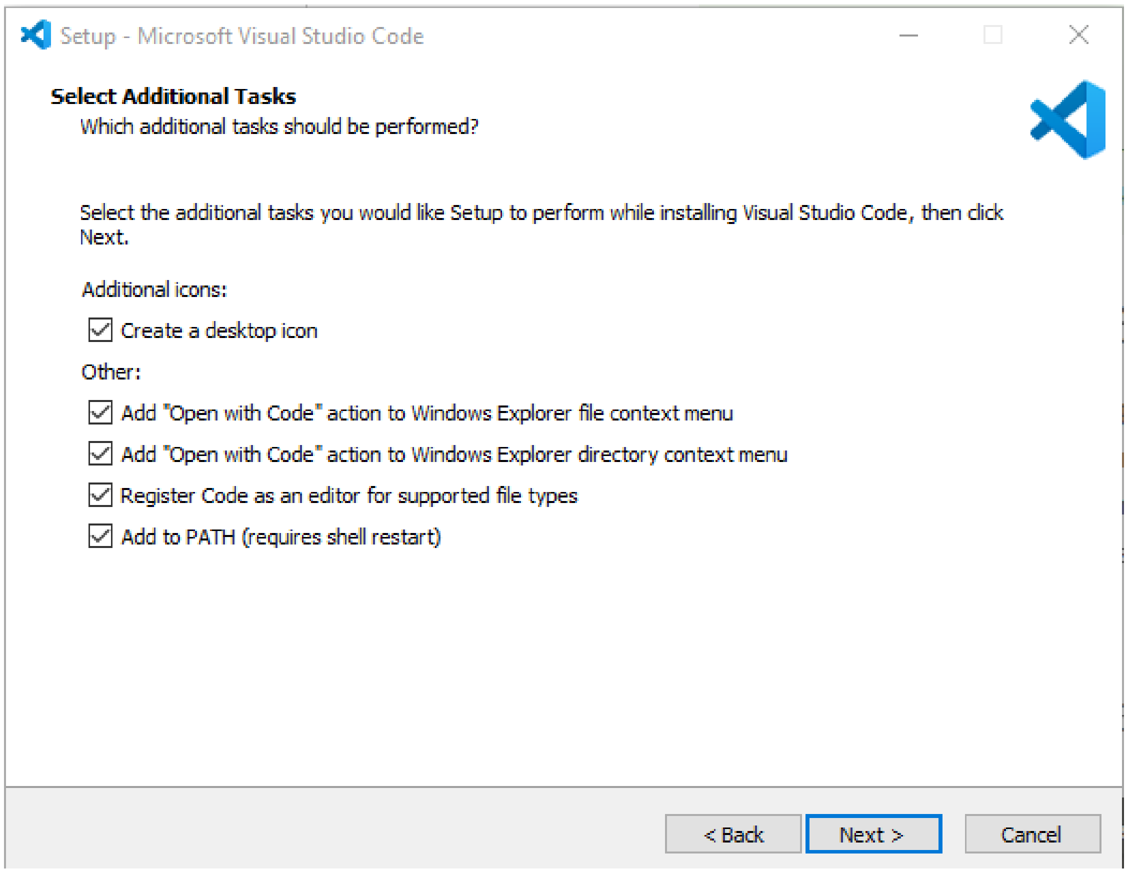
→ Customize installation
Choose location and features

☐ Install launcher for all users (recommended)
☑ Add Python 3.8 to PATH

Installing VS Code on Windows

IDLE is a great way to get familiar with Python, but as we proceed in the course we are going to find that IDLE is too limited. VS Code is an open source text editor with a wide array of extensions that benefit programmers. Follow the steps to install VS Code below:

1. Go to VS Code's Download Page and select the version appropriate to your OS. For Windows, choose the 64-bit version and choose *System Installer* if you are using your own machine and have admin privileges.
2. Run the installer and accept the defaults. You may also want to select options to make VS Code work as an editor for its supported file types.


Setup - Microsoft Visual Studio Code

Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

Additional icons:

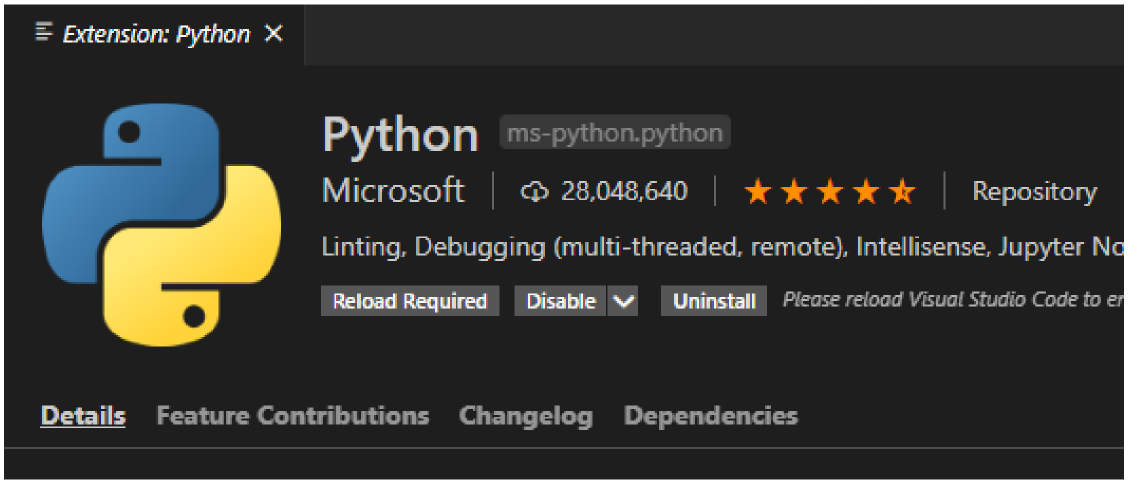
- [✓] Create a desktop icon

Other:

- [✓] Add "Open with Code" action to Windows Explorer file context menu
- [✓] Add "Open with Code" action to Windows Explorer directory context menu
- [✓] Register Code as an editor for supported file types
- [✓] Add to PATH (requires shell restart)

399e321b0ba596363a26783cbe76ee40.png

3. Run VS Code for the first time.
4. We will now install the Python extension for VS Code. Along the left side, you will see several icons. Select the bottom icon (Extensions) or press `Ctrl+Shift+x`. Enter “Python” into the search bar and install the first extension you see.


Extension: Python

Python ms-python.python

Microsoft | 28,048,640 | ★★★★★ | Repository

Linting, Debugging (multi-threaded, remote), Intellisense, Jupyter No

Please reload Visual Studio Code to er

6788a48bc7eea0622dad445bcb83361c.png

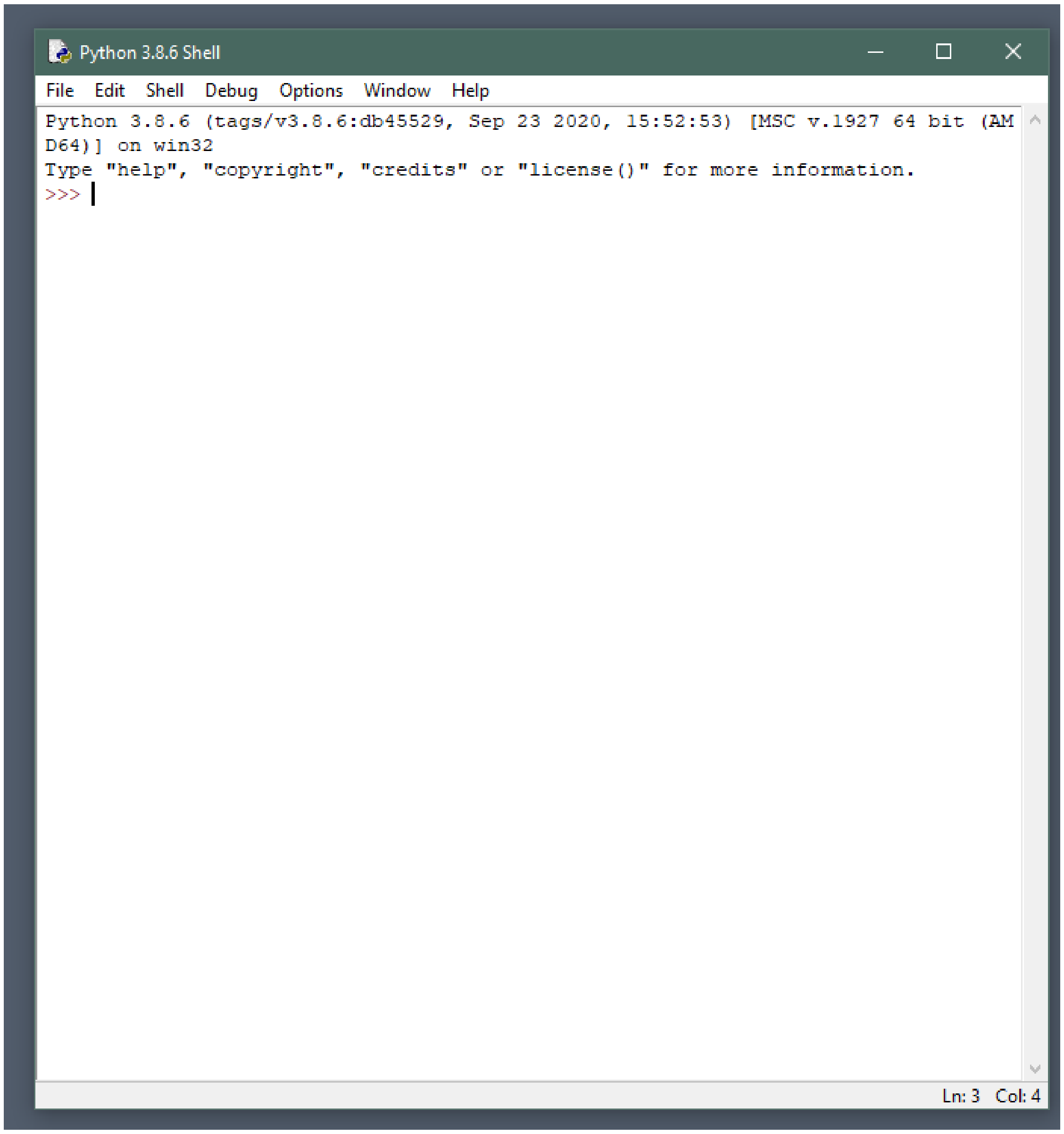
5. Follow prompts to reload VS Code when the extension is installed.

Other Preparation

It's also recommended that you create a directory for your course files. This can be named whatever you like, but you should probably also create a "Lab 1" directory inside of it to keep things organized. Organization is very important for programmers 😅.

1.2 Working With Python For The First Time

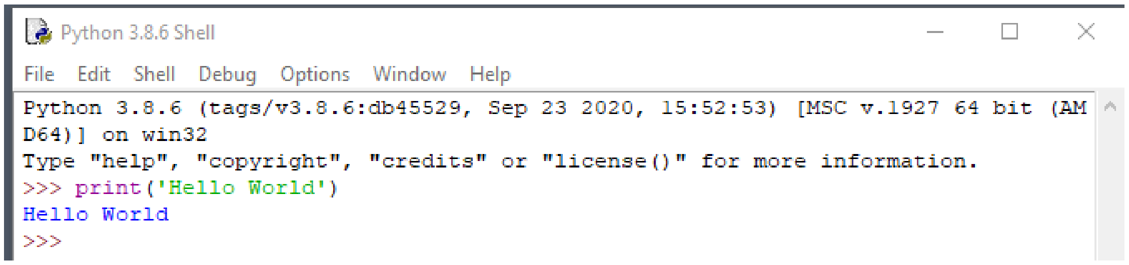
1. We will use Python's builtin text editor called *IDLE*. Run that now.



```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
```

9061b75aeab1f50c1377b94e5bd1f58b.png

2. You will see a prompt >>>> waiting for your input. Type `print('Hello World')` and press Enter. Congratulations, you have printed your first *string*!



```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('Hello World')
Hello World
>>>
```

1490e0ad99ed17743a35df6922c26011.png

What is a string?

A string is a series of characters entered from a keyboard. In this case, our string is **Hello World**. Strings can also contain numbers and special characters such as (&%$#). You can't do math with strings, but you can print them to a screen. The string is our first *data type*. There are others that we will introduce further on in the course.

For now, remember that strings need to be inside either single quotes ('') or double quotes ("").Use `print()` again and enter another sentence of your choice.

3. `print()` is a built-in *function* used to print output to the screen. Programmers use a particular vocabulary to talk about functions: the things we put inside the parentheses are called *arguments*. They modify what the function will do. In the case of `print()`, we expect only one argument, and that argument should be a string.

1.3 Working With User Input

1. Now in Idle, enter the following line and press Enter:

```
x = input('Enter your name: ')
```

2. You should see that the program has paused and is waiting for input. Please enter your name.
3. `input()` is another example of a function. Again, you can enter a string as an argument and it will be used to prompt the user for input. `input()` will also *return* something. Whatever is returned by a function, needs to be stored inside a *variable*. We have named our variable x, but this isn't a very good name. Let's change it.
4. Use the **up arrow key** (↑) to cycle through the commands you have already entered. Select the `x = input('Enter your name: ')` command and replace x with `user_name`.
5. When we name variables, we want our names to be descriptive in order to reduce confusion. Programming is confusing enough!
6. Variables will store any important data while our program is running. We can use the variable name to retrieve that stored information now. Enter the following:

```
print(user_name)
```

7. Notice that Python will replace the variable name (`user_name`) with whatever you typed in.
8. In addition, we can combine two strings using the plus sign (+). Enter the following:

```
print("Hello" + user_name)
```

You will probably see something similar to this:

```
HelloEric
```

9. Remember that spaces are also considered to be part of our string. In order to make this output a little better, put a space inside your *double quotes*.
10. Additionally, let's make the output a little more friendly by ending the string with an exclamation mark. After `user_name`, put another plus sign, then an exclamation mark enclosed in quotes. It does not matter if you use double quotes or single quotes. Your new `print` statement should look something like this:

```
print("Hello " + user_name + '!')
```

...which should give you the following output:

```
Hello Eric!
```

I've combined double quotes with single quotes. This doesn't matter, as long as every *opening quote* has a matching *closing quote*. (eg. Entering `print('Hi")` will not work).

Note: When naming things in Python (including variables), keep in mind the following rules:

1. Use letters, numbers and underscores (_).
2. Variable names can't start with a number.
3. Variable names are case sensitive. *Use only lower case letters*.
4. You can't use spaces, and you can't use special characters.
5. You will notice that some special keywords (such as `print`) are reserved for the Python programming language and cannot be used.

1.4 Creating Our First Python Script

So far, we have been interacting with the *Python Interpreter*, which is a way of saying that we are entering code, and Python is translating each line of code into instructions. This is useful because we get immediate feedback from the interpreter: if there is a problem with our line of code, we will know immediately.

The next step will be to enter our code into a file, so that we can save our instructions and have Python repeat them whenever we need. We often call these files programs or *scripts*.

By script we simply mean a short computer program meant to perform a simple task quickly. The Python Interpreter is still translating your code, but this time your code has been saved in a file, and will be executed like any other program on your system.

1. From Idle, click on the menu item `File` and select `New File`.
2. Now, enter the following lines, similar to what you entered into your interpreter just a moment ago.

```
'''
Name: Your name here.
Student ID: Your nine-digit seneca ID.
Description: This is our first program.
'''

print("Hello World!")
user_name = input("Please enter your name: ")  # asks user for input
print("Hello " + user_name + "!")  # prints user name in a friendly way.
```

3. Save your file inside your "Lab 1" directory. The file should be named `lab1a.py`.

Note: the `.py` is called a *file suffix*, and is used to identify what kind of file this is. This isn't like a normal text file, so it shouldn't be named with a `.txt`! Identify this as a Python script by ensuring that `.py` is at the end.

---

4. Now at the top of the Idle window, click on `Run` -> 'Run Module'. Make sure that your first script is running successfully and doing what you expect it to do.
5. Notice that the lines between `'''` didn't affect anything, nor did the words coming after the `#`. These are examples of *comments*. Comments are used to document code in various ways.

Believe it or not, but being able to write good comments is a crucial skill for programmers. Often we return to old code that we have written, and it can be difficult to understand what we wrote and why. Comments are a way to describe *why we have written the code that we have*. For this course, you will be graded partially on the quality of your code comments.

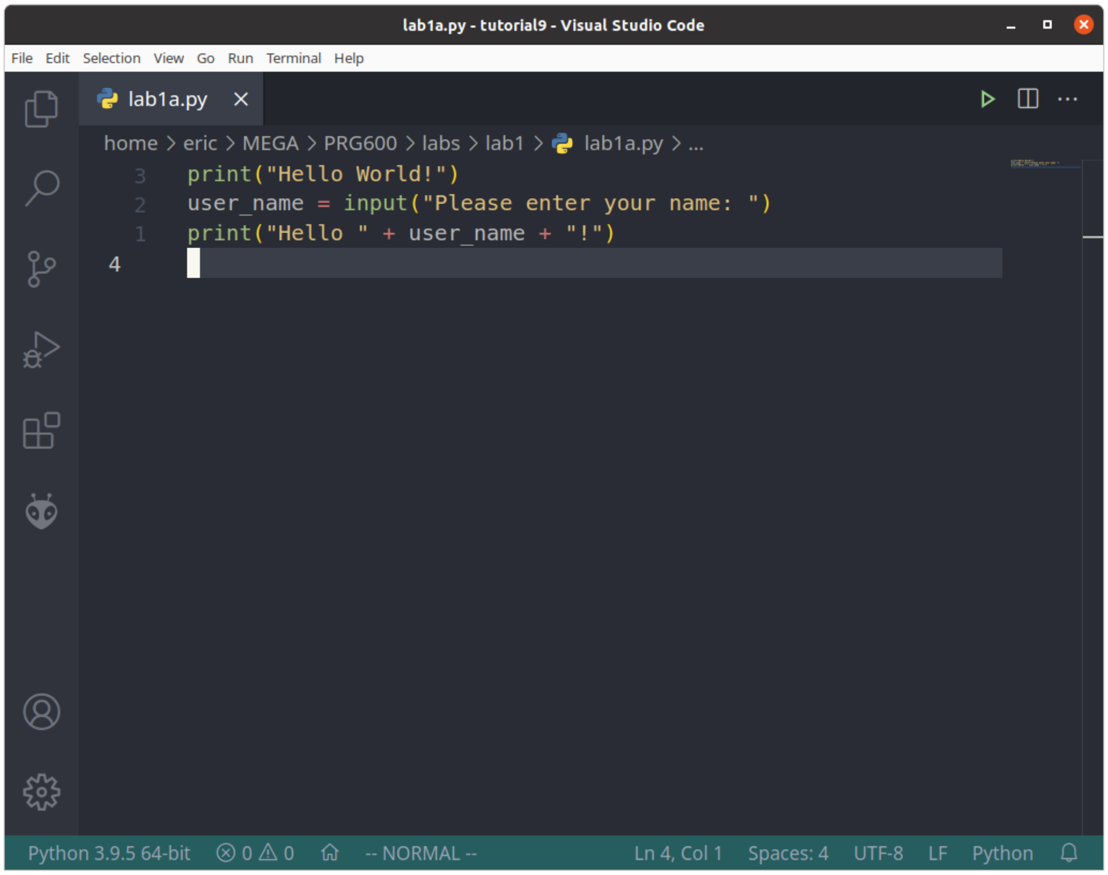
- **Docstrings** are usually paragraphs that might describe how the use the script. In this course, you will be asked to put docstrings with your name and student ID in **each** script you submit.
- **In-line comments** are short sentences that explain *why* you are performing the instruction on this line. You will **also be asked to explain your code using these comments**. Explaining your approach is an important part of things like **whiteboard job interviews**.

1.5 Opening a Script in VSCode

All of our Python work will be inside these `.py` files. We will soon be writing scripts of such complexity that we will require better tools than IDLE can provide. So let's make sure that our first script runs inside VSCode.

1. If you haven't already, start up **VSCode**.

2. Use the “Open File…” dialog to find your `lab1a.py` and open it. You should see a screen similar to this:


lab1a.py - tutorial9 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
lab1a.py
home > eric > MEGA > PRG600 > labs > lab1 > lab1a.py > ...

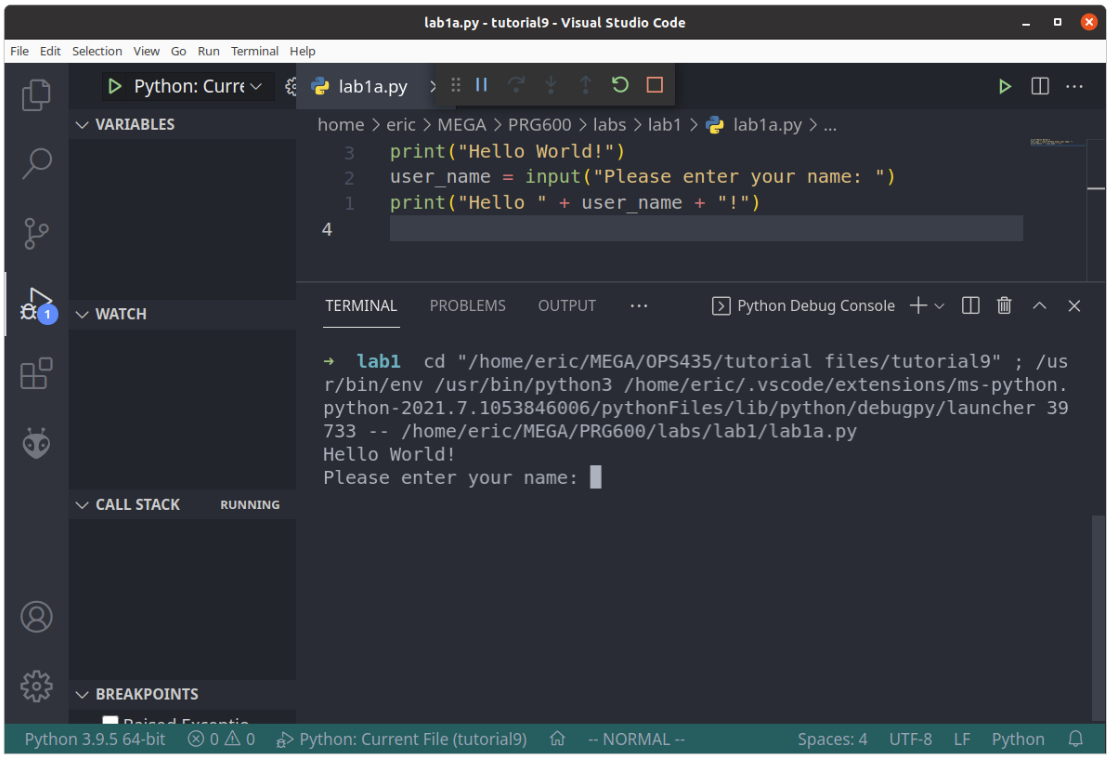
```
print("Hello World!")
user_name = input("Please enter your name: ")
print("Hello " + user_name + "!")

```

Python 3.9.5 64-bit ⓧ 0 ⚠ 0 -- NORMAL -- Ln 4, Col 1 Spaces: 4 UTF-8 LF Python

3. On the left side, find the “Run and Debug” button or press `Ctrl + Shift + D`. Click on the "play button to launch your program.

4. You may be asked to select a configuration. Make sure to select ‘Python’ from the list. If all is well, you should see a screen like this:


lab1a.py - tutorial9 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Python: Curre
VARIABLES
WATCH
CALL STACK RUNNING
BREAKPOINTS
TERMINAL PROBLEMS OUTPUT
Python Debug Console
home > eric > MEGA > PRG600 > labs > lab1 > lab1a.py > ...
Python 3.9.5 64-bit ⊗ 0 ⚠ 0 Python: Current File (tutorial9) -- NORMAL -- Spaces: 4 UTF-8 LF Python

```python
print("Hello World!")
user_name = input("Please enter your name: ")
print("Hello " + user_name + "!")
```

```
➜  lab1  cd "/home/eric/MEGA/OPS435/tutorial files/tutorial9" ; /us
r/bin/env /usr/bin/python3 /home/eric/.vscode/extensions/ms-python.
python-2021.7.1053846006/pythonFiles/lib/python/debugpy/launcher 39
733 -- /home/eric/MEGA/PRG600/labs/lab1/lab1a.py
Hello World!
Please enter your name: 
```

vscode2

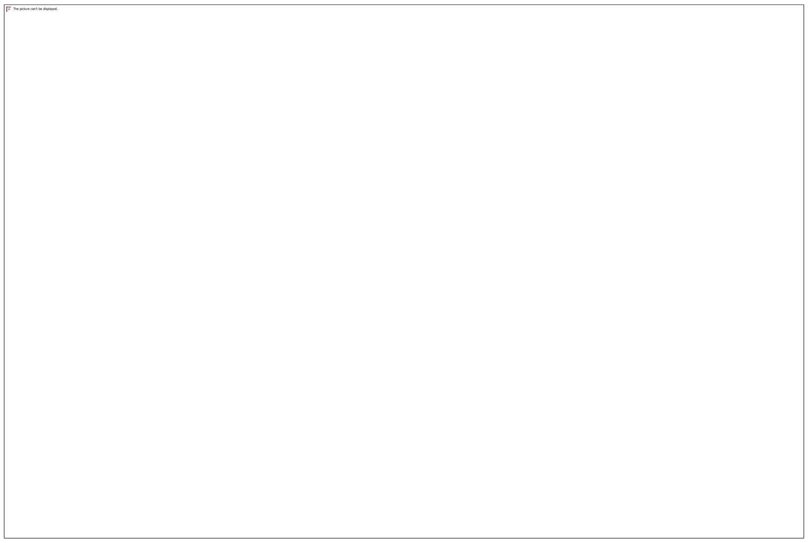
5. VSCode allows us to run our code inside of a *terminal*, which is another way of saying that we are able to interact with Python. You will be learning how to working with terminals and command line interfaces in other courses, but additionally it will be our primary way of interacting with Python.

Note: If you are *not* seeing the Python script at this point, contact your professor. We are going to be using these tools for all of the next labs and assignments, so if there's an issue it's better that we resolve it sooner rather than later!

1.6 TASK: Using A Check Script

For this lab, you can use a check script that will verify that your files are printing the correct output. Here are the steps for using the check script:

1. Download the file **check-lab1.py** from here.
2. Copy check-lab1.py into your Lab 1 directory, the **same directory** as where you have put lab1a.py.
3. Start VSCode if it isn't already open. From "File", select "Open Folder…". Select the Lab 1 directory.
4. If you get a message about this directory being open in "Restricted Mode", be sure to select "Trust the authors of this directory".



A restricted mode dialog

1. Double-click on the check script from the file manager in the left panel. Make sure that this file has *focus* (that is, it is the active document in VSCode).
2. Run this file the same way you ran your lab1a.py file.
3. In the bottom terminal, you will get messages telling you if there were errors in lab1a.py. Go back and verify that the output of lab1a.py matches *exactly* what's shown in the lab.
4. You will also see that the check script has failed because lab1b.py cannot be found. This is your next task to complete the lab.

1.7 TASK: Your Next Script

You will now be asked to create a new script. Your script must fulfill the following requirements:

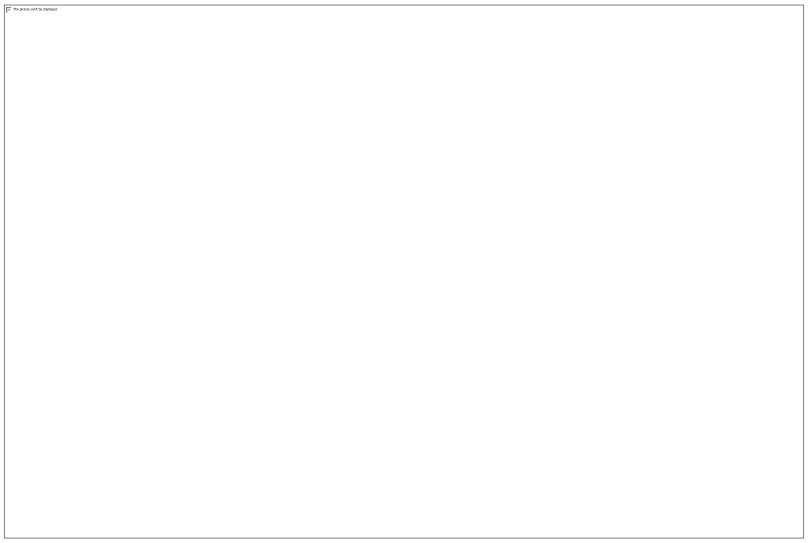
- ☐ Your script is named `lab1b.py`.
- ☐ Your script contains a docstring (`'''`) with your name and student ID.
- ☐ Your script will use a combination of `input()` and `print()` statements to re-create the dialog below. Your script will also run without errors.

Example Dialog

```
Welcome to PRG600!
Enter your name: Eric
Enter your student ID: 123456789
Enter your age: 41
Enter your favorite food: pizza
Eric is 41 years old and loves to eat pizza!
Good luck with the course!
```

- Once you have completed this script, save it and run the check script again. Make sure that the check script has focus when you click "Run and Debug"!

- Use output from the check script to verify that your script is producing the correct output. If everything is successful, you should see the terminal print two “OK” messages.



An example of a successful check

- Once you get no warnings or errors, find the file that the check script has created. It is called `check-lab1-output.txt`.
- Submit this file along with your two scripts.

Deliverables

To complete the lab, you will need to submit the following to Blackboard:

- ☐ lab1a.py
- ☐ lab1b.py
- ☐ check-lab1-output.txt

Congratulations! You have completed the first lab.

Glossary

- Interpreter
- Script
- Variable
- Argument
- Function
- String