

Getting Started with Python

This handout covers how to install **Anaconda Distribution** and run Python code. By installing Anaconda, you get the core **Python interpreter**; you get **Spyder**, which is an integrated development environment (IDE) for Python coding; and you get a vast collection of packages that extend the functionality of core Python. In particular, we will be using the **matplotlib** and **numpy** packages later in this course to produce charts.

If you have problems with installation, please go to **office hours** or **post on Piazza** for help.

Downloading the Installer

NOTE: The installer we'll be downloading is larger than the average file, because it contains Python, associated packages, a code editor, and some other toys.

Visit the [download page](#) for Anaconda Distribution. The page should detect your operating system and present the appropriate **Download** link near the top. The current version is **Anaconda3 2022.10**, and it comes with **Python 3.9**. **This class is taught using Python 3.6 or higher**. Do not download any version with an older Python.

Individual Edition is now

ANACONDA DISTRIBUTION

The world's most popular open-source Python distribution platform



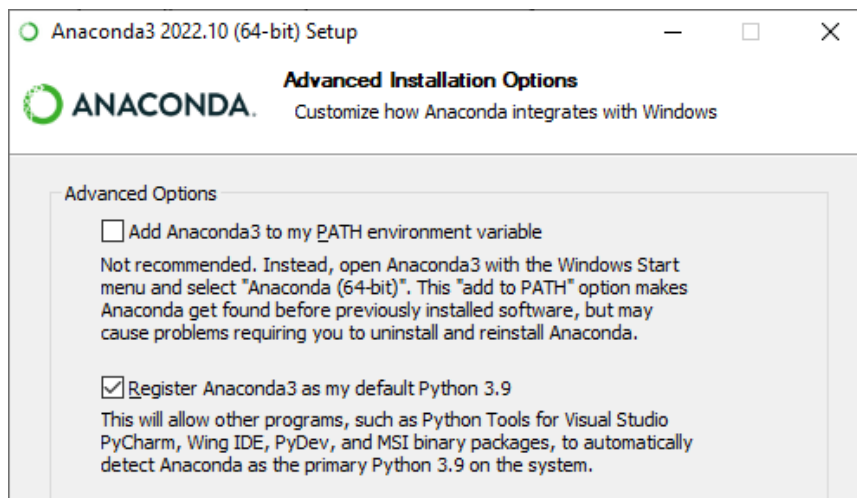
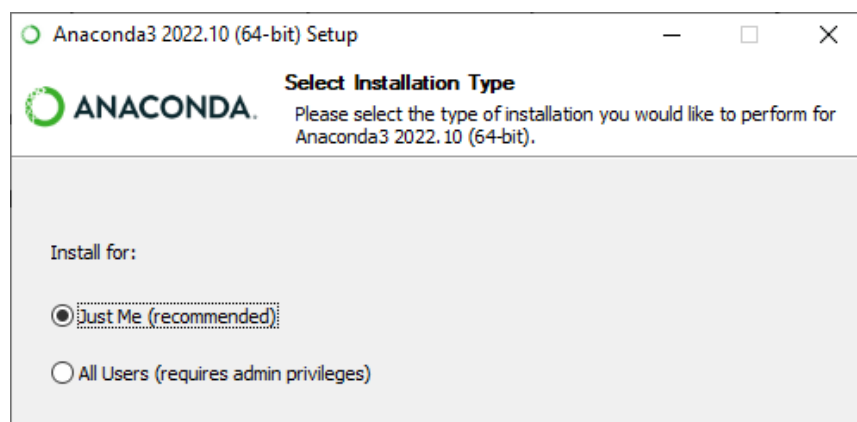
If you need an installer for a different operating system, scroll to the bottom, and select the desired version. We recommend the graphical versions when possible.

Installing Anaconda

An Anaconda installation is meant to be self-contained. Typically, this means it installs somewhere in your **local home folder**, and should not require Administrator or root privileges. It also shouldn't affect other Python installs on your system, if you have any.

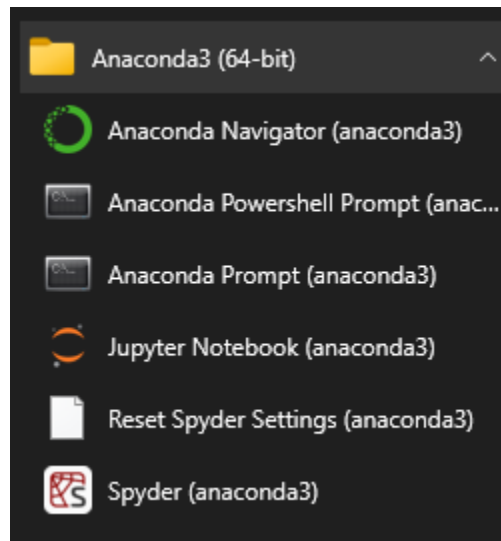
NOTE: Detailed install/uninstall instructions can be found on [Anaconda's documentation website](#).

After downloading the graphical installer, double click the .exe (Windows) or .pkg (Mac) file, and accept all the default options. (The defaults on Windows are shown below.) The entire process should take just a few minutes.

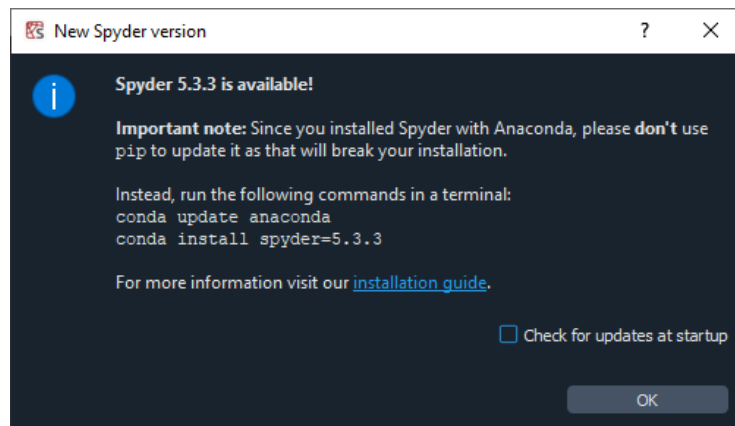


Starting Spyder

Spyder stands for **Scientific Python Development Environment**. Being an [integrated development environment](#), it provides a useful interface for writing, running, and debugging code. After installing Anaconda, Spyder should be accessible from your operating system's launcher (Windows start menu, screenshot below, or the macOS launchpad).



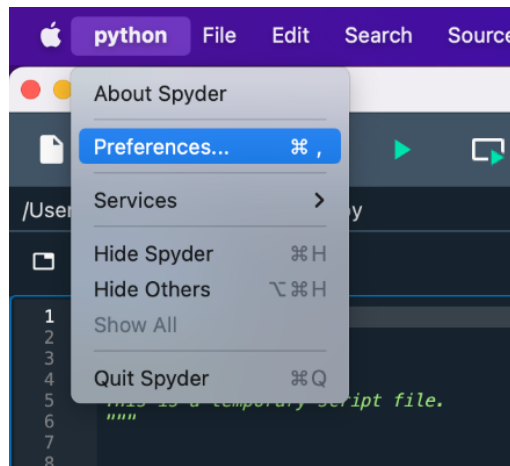
The first time you open Spyder, it may take a few moments to initialize, and then its window will appear. Spyder may prompt you to update, but we suggest you ignore the message.



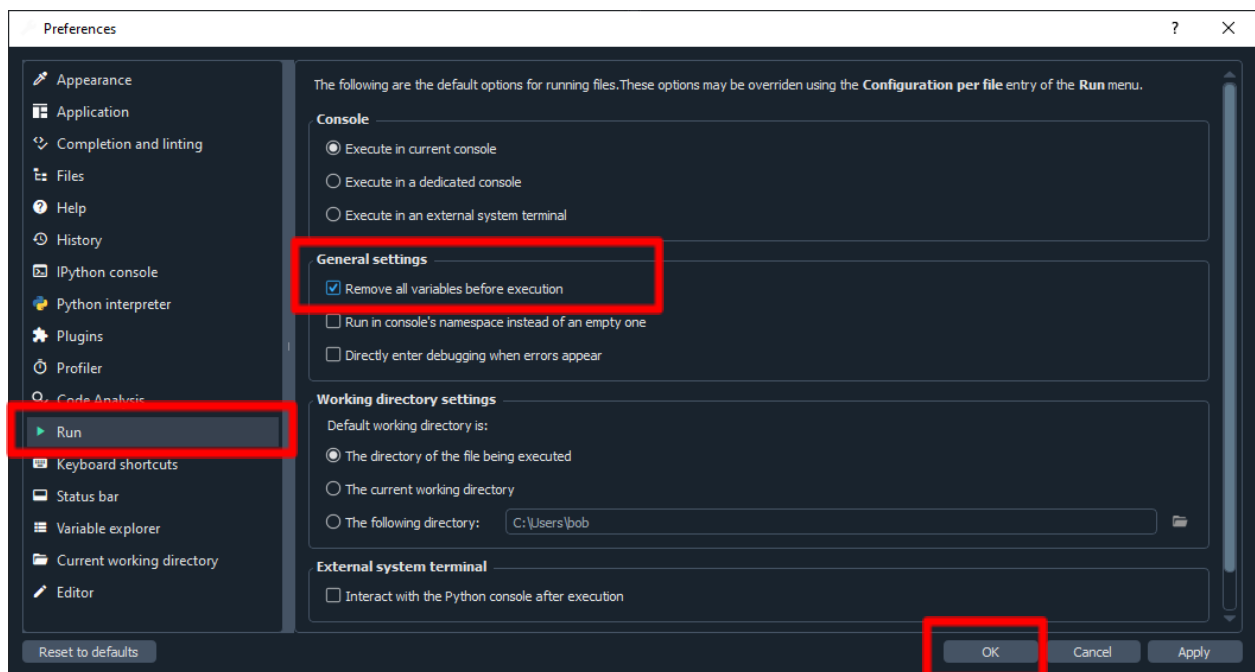
If you can't find Spyder in your launcher, you can also open **Anaconda Navigator** (wait for it to initialize), and then launch Spyder from within it.

Configuring Spyder

Before using Spyder, we recommend configuring it to clear variables before execution. This will help prevent future debugging issues due to retaining old data between executions of your code. This feature can be found in the preferences, accessed by **python** → **Preferences** on Mac, or **Tools** → **Preferences** on Windows and Linux.



Once in Preferences, choose the “Run” category, find the “General settings” group, and check “Remove all variables before execution”. Click “OK” when finished.

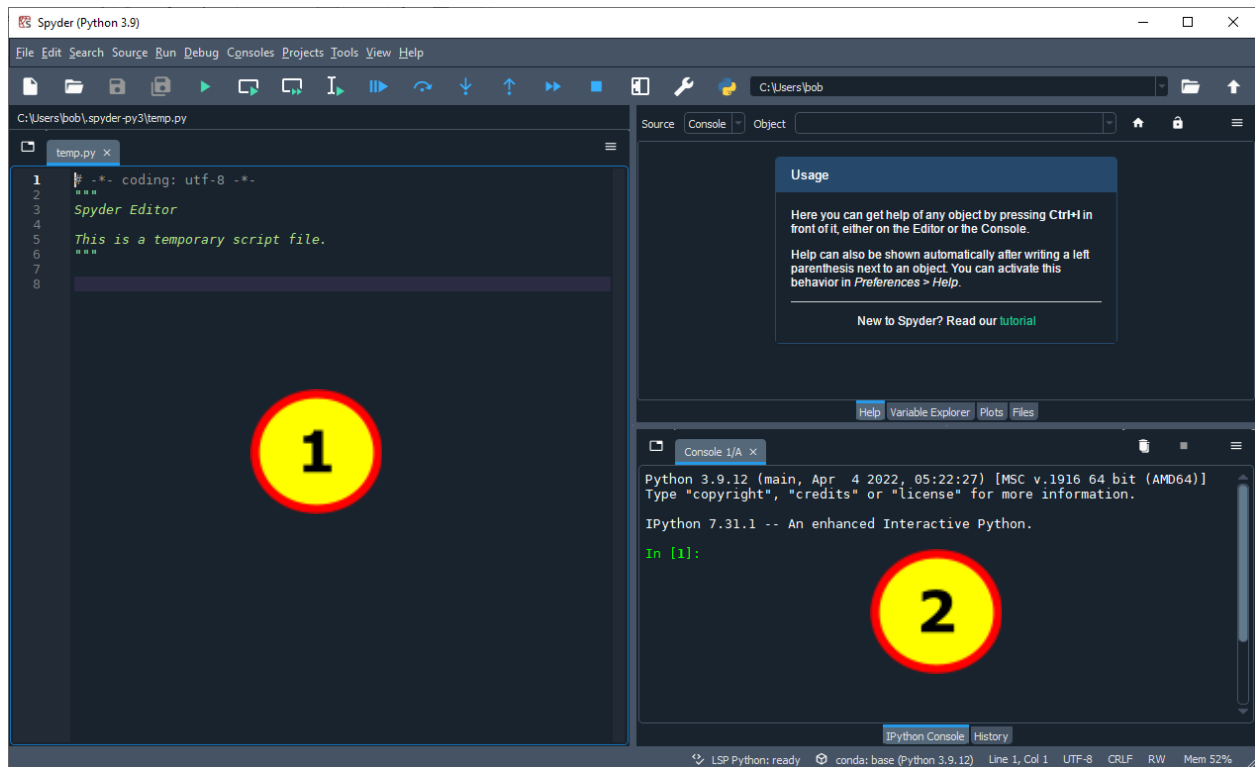


Editing and Running Code in Spyder

When you launch Spyder, you will see a window consisting of multiple panes. The two most commonly used panes are:

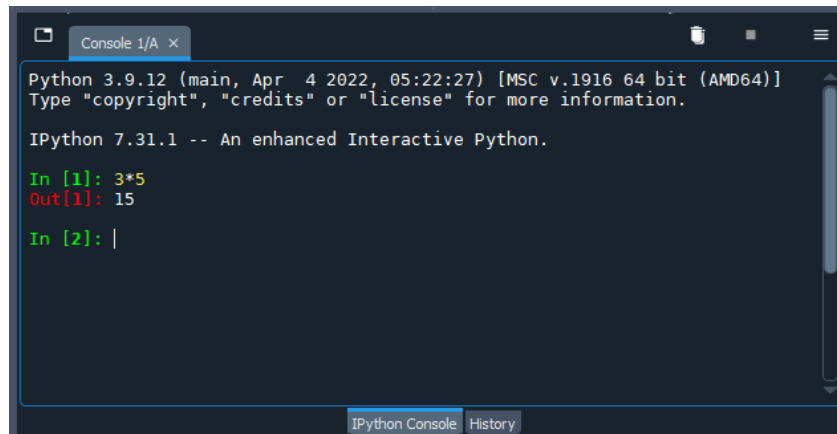
1. a **code editor** that lets you create and edit existing Python source files
2. the **IPython interpreter**, which gives you access to Python's interactive mode

During the following discussion of Spyder's features, you should try to replicate the screenshots.



Using the IPython prompt:

The IPython prompt looks something like this: “In [1]:”. You type Python code directly into this prompt, and pressing Enter executes the code fragment. Try typing the following after the prompt and pressing the Enter key: **3*5**

A screenshot of a terminal window titled 'Console 1/A'. The window shows the IPython prompt 'In [1]:' followed by the input '3*5'. The output 'Out[1]: 15' is displayed below the input. The window also shows the IPython version '7.31.1' and the Python version '3.9.12'. The prompt 'In [2]:' is visible at the bottom of the console area.

```
Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: 3*5
Out[1]: 15

In [2]: |
```

Addition (+), subtraction (-), multiplication (*), division (/), modulus/remainder (%), and exponentiation/power (**) operators are built into the Python language. This means you can use them right away. If you want to use a square root in your calculation, you can either raise something to the power of 0.5 or you can **import** the **math** module. Do not worry about what that means right now – we will cover it later during the course. Below are two examples of square root calculation:

```
In [1]: 16**0.5
Out[1]: 4.0
```

```
In [2]: import math
```

```
In [3]: math.sqrt(16)
Out[3]: 4.0
```

The math module allows you to do a number of useful operations:

```
In [4]: math.log(16, 2)
Out[4]: 4.0
```

```
In [5]: math.cos(0)
Out[5]: 1.0
```

(The exercises below are just for practice. Solutions will not be graded or collected.)

Use the IPython prompt to calculate:

1. $6 + 4 * 10$

2. $(6 + 4) * 10$

- a. Compare this to #1, and note that Python uses parentheses just like you would in normal math to determine the order of operations.

3. 23.0 to the 5th power

4. Positive root of the following equation:

$$34x^2 + 68x - 510$$

a. Recall, given $ax^2 + bx + c$, then $x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$.

5. `import math`

`math.cos(3.4)**2 + math.sin(3.4)**2`

Accepting user input:

In Pset 1, you will be using Python's `input()` function to prompt for user input. To verify that it works, enter the following line at the IPython prompt. It should repeat the question back at you, and once you type in an answer (and hit Enter), it will print that again as the output.

```
In [1]: input("What do you want for your birthday? ")
```

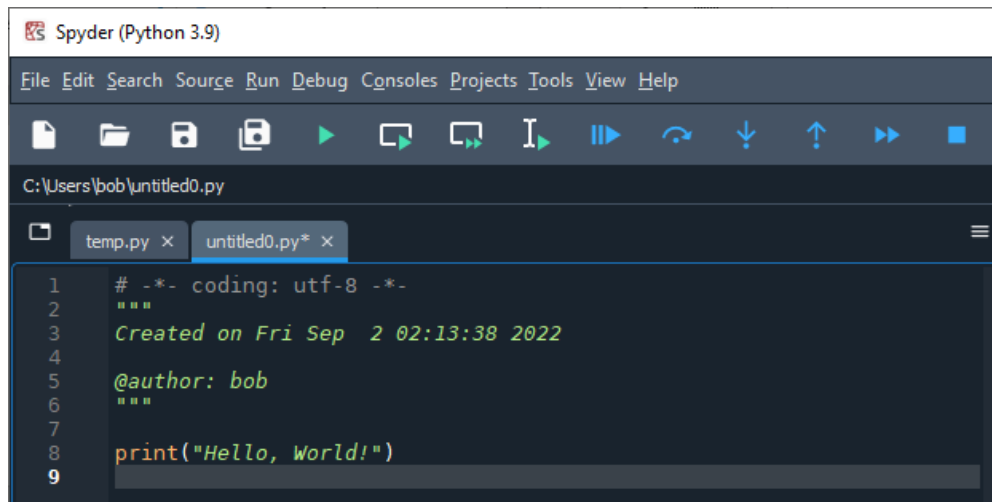
```
What do you want for your birthday? I would like a hat.
```

```
Out[1]: 'I would like a hat.'
```

To create, save, and run a file:

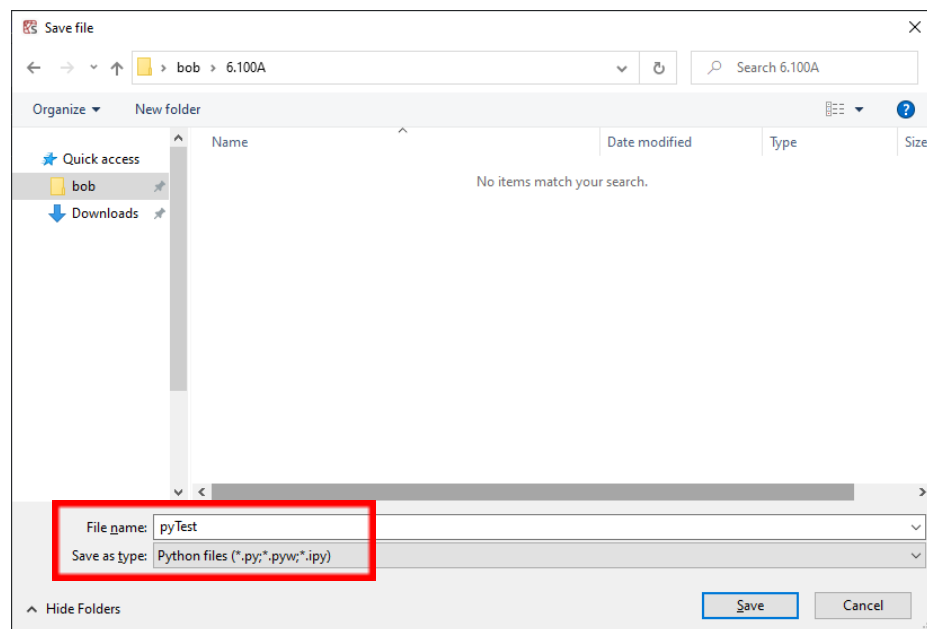
Creating the file

1. In Spyder's File menu, select "New file".
2. In the new file, type the following: `print("Hello, World!")`



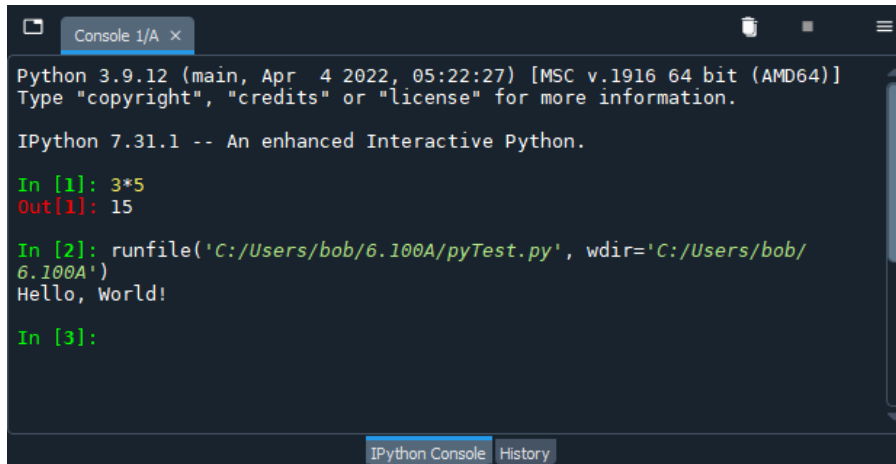
Saving the file

1. To aid in organization, save your files in a location specific for this course. First create a directory/folder for your 6.100A material, with an appropriate name.
2. From Spyder's File menu, click "Save as..." and then navigate to your course folder before typing a name for this file, e.g., "pyTest". Make sure that you're saving the file as type Python before clicking "Save".



Running the file

1. Go to Spyder's "Run" menu, click on "Run". (A popup may appear asking to confirm run settings. Go ahead and click "Run".) You should see two parts to the output in the IPython console. First, the `runfile(...)` line shows you the path for the file you just ran. Second, the output of the file, "Hello, world!", appears.



```
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

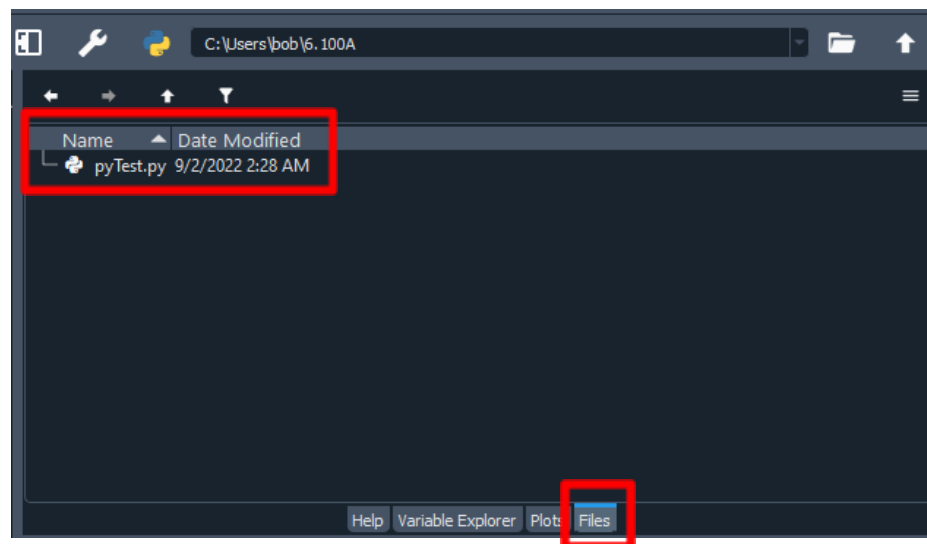
IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: 3*5
Out[1]: 15

In [2]: runfile('C:/Users/bob/6.100A/pyTest.py', wdir='C:/Users/bob/
6.100A')
Hello, World!

In [3]:
```

2. Go back to the Code Editor, and add the following line: `print("I like 6.100A!")`
3. Select "Run" again (you can also use the green triangle Run button on the toolbar or the keyboard shortcut F5), and observe the change in result.
4. Close your test file by clicking the X in its filename tab. Then reopen it by double-clicking its name in the Files pane towards the top right.



5. Congratulations – you now know how to enter expressions in Python, and how to save them into files and run them!

Installing and Testing Matplotlib and Numpy

To work with charts, you will need the Python library packages [matplotlib](#) and [numpy](#). You will not need them until later assignments, but they will be used in class demos, so we encourage you to install them now.

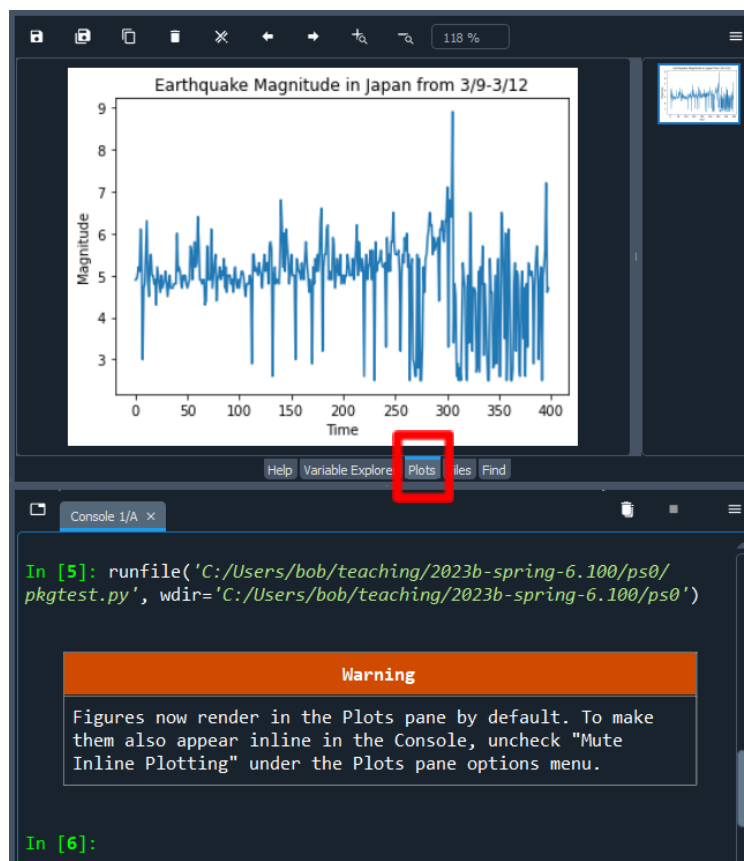
Anaconda Distribution already includes matplotlib and numpy in its installation. Start Spyder and type “import matplotlib” and “import numpy” into the prompts. If no errors show up, you already have them installed. If errors show up, reach out via Piazza!

```
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: import matplotlib
In [2]: import numpy
In [3]: |
```

We have provided a test script **pkgtest.py** that you can run to verify installation of matplotlib and numpy. It's okay if a warning like the one below appears, and you can find the generated chart in the “Plots” pane.



Created(?) by Asfandiyar Qureshi, Feb 2006

Edited by Vladimir Bychkovsky, Sept 2006

Edited by Calvin On, Feb 2007

Edited by Yang Zhang, Sep 2008

Edited by Chih-yu Chao, Feb 2009

Edited by Sari Canelake, Dec 2009

Edited by Anjali Muralidhar, Feb 2013

Edited by Niki Castle, Feb 2013

Edited by Prashan Wanigasekara, Feb 2014

Edited by Zachary Gil Freeman, Feb 2015

Edited by Ana Bell, August 2016

Edited by Andrew Wang, September 2022

Edited by Andrew Wang, January 2023