



CS-1004, Introduction to Programming  
for Non-Majors, C-Term 2017

## Setting up Python 3.5 and numpy and matplotlib on your own Windows PC or laptop

Hugh C. Lauer<sup>©</sup>  
Adjunct Professor  
Worcester Polytechnic Institute

Programming assignments in CS-1004 will be in the programming language *Python* — specifically, version 3.5.1 of *Python*. In addition, you will need several *Python* packages, including one called *numpy* (meaning “Numerical Python”) and one called *matplotlib*, a *Python* version of the popular *Matlab* system. This document provides instructions for installing *Python 3.5.1* on *Windows 7*, *Windows 8*, and *Windows 10* laptop and desktop computers. This document also includes instructions for installing *Python* packages such as *numpy* and *matplotlib*.<sup>1</sup>

Public laboratory computers at WPI will have *Python 3.5.1*, *numpy*, and *matplotlib* installed on them for the academic year 2016-2017.

In general, it is expected that assignments will be compatible among Windows, Macintosh, and Linux systems, provided that they all use compatible versions of *Python*, *numpy*, and *matplotlib*.

**Note:** There are two different, incompatible versions of *Python* in general use around the world — *Python 2.7* and *Python 3.5.1*. Significant changes to the *Python* language were made between *Python 2.x* and *Python 3.y* (for all values of  $x$  and  $y$ ). The *Python 3* language is cleaner, more self-consistent, and more user-friendly. Programs written for versions of *Python 2* will not necessarily run on *Python 3* installations; if they do run, they may get different answers to the same problem.

That being said, a lot of legacy *Python 2* code is still in use, and new *Python 2.7* code is still being written and distributed by organizations that have not yet upgraded to *Python 3*. Not all *Python 2* packages have been ported to *Python 3*.

**Note 2:** There are a number of other integrated environments for supporting Python programming, including *PyCharm* and *Enthought*. Most of these are more advanced than what is needed for this course. If you choose to use one of them, you are on your own for installation.

---

<sup>©</sup> Copyright 2016, Hugh C. Lauer. All rights reserved. Permission is given for use in courses at Worcester Polytechnic Institute, Worcester, Massachusetts.

<sup>1</sup> If you have a Macintosh or Linux computer or laptop, please refer to these documents instead:— [docx](#), [pdf](#)

## Installing Python 3.5.1 on Windows Systems<sup>2, 3</sup>

There are two variants of *Python 3.5.1* for Windows — a 32-bit version and a 64-bit version. Almost all Windows PCs sold over the past few years are 64-bit systems. Therefore, these instructions focus primarily on installing the 64-bit versions. If you have a 32-bit version of Windows, please seek assistance from the Professor, the TAs, or the Helpdesk.

To obtain the correct 64-bit version of *Python*, click on this link — [python-3.5.1-amd64.exe](#)<sup>4</sup> — and download the resulting file to a convenient folder or directory. Alternatively, you may browse to

<http://www.cs.wpi.edu/~cs1004/c17/Resources>

and download it from there.

*Right-click* on the file **python-3.5.1-amd64.exe** and select *Run as Administrator* to start the installation. You should be greeted by a dialog box resembling the following:–

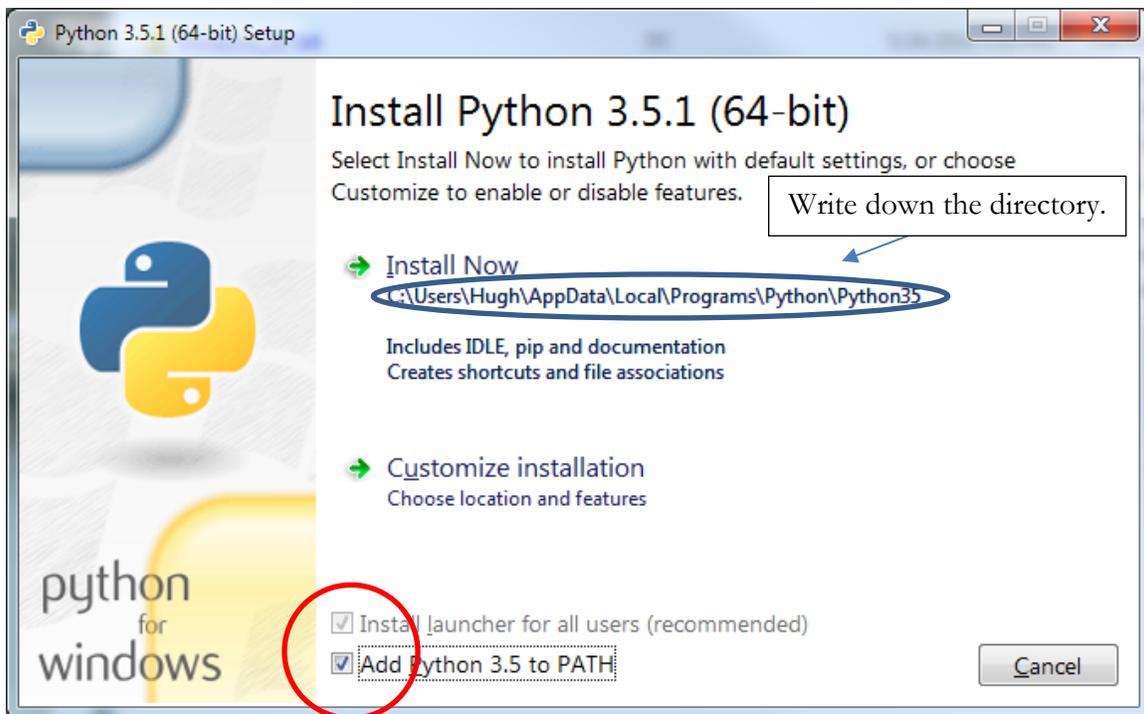


Figure 1

Be sure to check the box at the bottom labeled “Add Python 3.5 to PATH” (shown in a red circle).

**Note1:** This checkbox may seem trivial, but forgetting to check it has led to numerous problems for students in previous terms!

- <sup>2</sup> It is useful to print out the relevant sections of this document. If you try read them on-screen, the dialog boxes of the installation tend to obscure the instructions, just when you need them the most!
- <sup>3</sup> These instructions have been tested on both *Windows 7* and *Windows 10*. The professor no longer has a copy of *Windows 8* for testing.
- <sup>4</sup> The processor designation “**amd64**” applies to microprocessors made by *both* Intel and AMD.

**Note 2:** It is *absolutely essential* that you run the installation as *Administrator*. If you forget to do so, the installation will appear to proceed successfully, but you are likely to get weird errors at runtime and even in the installations of *numpy* and *matplotlib*.

If you forgot to select *Run as Administrator*, uninstall *Python* and start over.

Three big “ifs”:-

- If there is an earlier version of *Python 3.x* installed on your computer (for any value of *x*), you should **Cancel** this installation and remove (i.e., uninstall) the previous version before installing this one.
- If you have a 32-bit installation of Windows (an unlikely event nowadays), you should cancel this installation and download and install [python-3.5.1.exe](#) instead. You will also have to install the 32-bit versions of *matplotlib* and *numpy*. It is suggested that you seek help.
- If instead of Figure 1, you see a dialog box resembling Figure 2 below for any version of *Python*, and if you know what you are doing, you may select *Modify* or *Repair*. However, for most students of this course, it is recommended that you select *Uninstall* to remove any conflicting installation. Uninstalling *Python* will take several minutes and may require you to confirm in one or more additional dialog boxes.

After removing the previous version of *Python*, click *Finish* and start over at Figure 1.

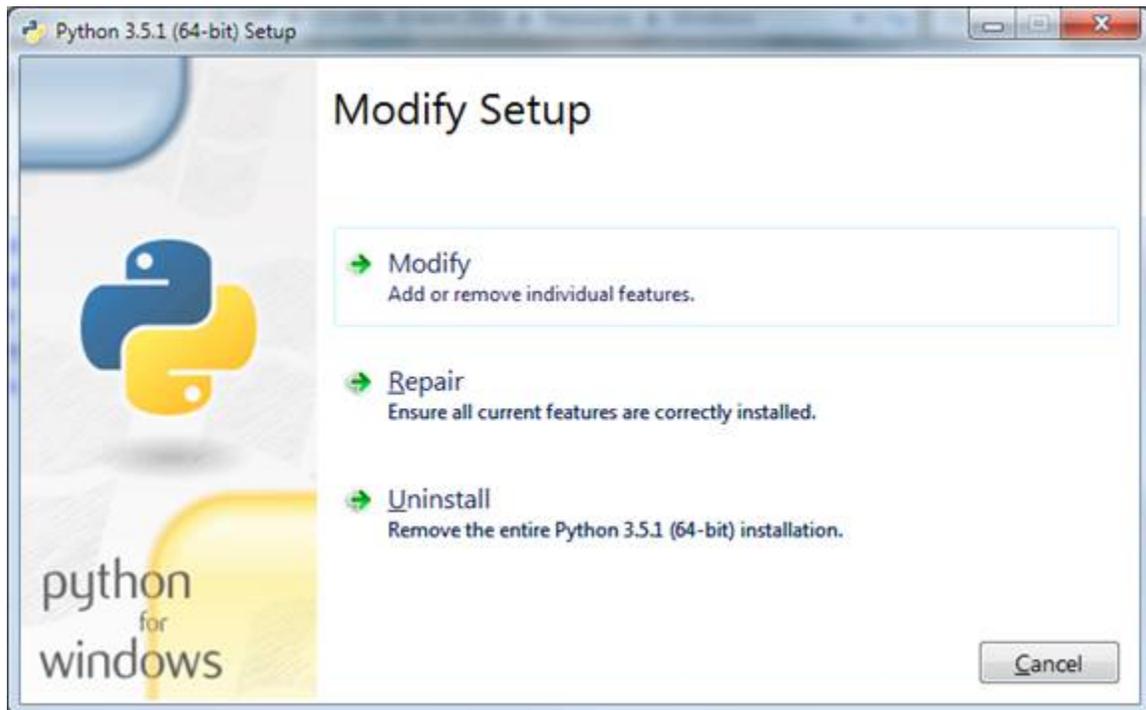


Figure 2

Returning now to Figure 1, if you are the only user of this computer or laptop, you may click on *Install Now* to start the installation. The installation will proceed and will finish with Figure 3 below.

If, however, your computer is used by multiple people with different user names, then click on *Customize Installation* to install *Python* in a more generic place. This will instead bring up Figure 4 below.

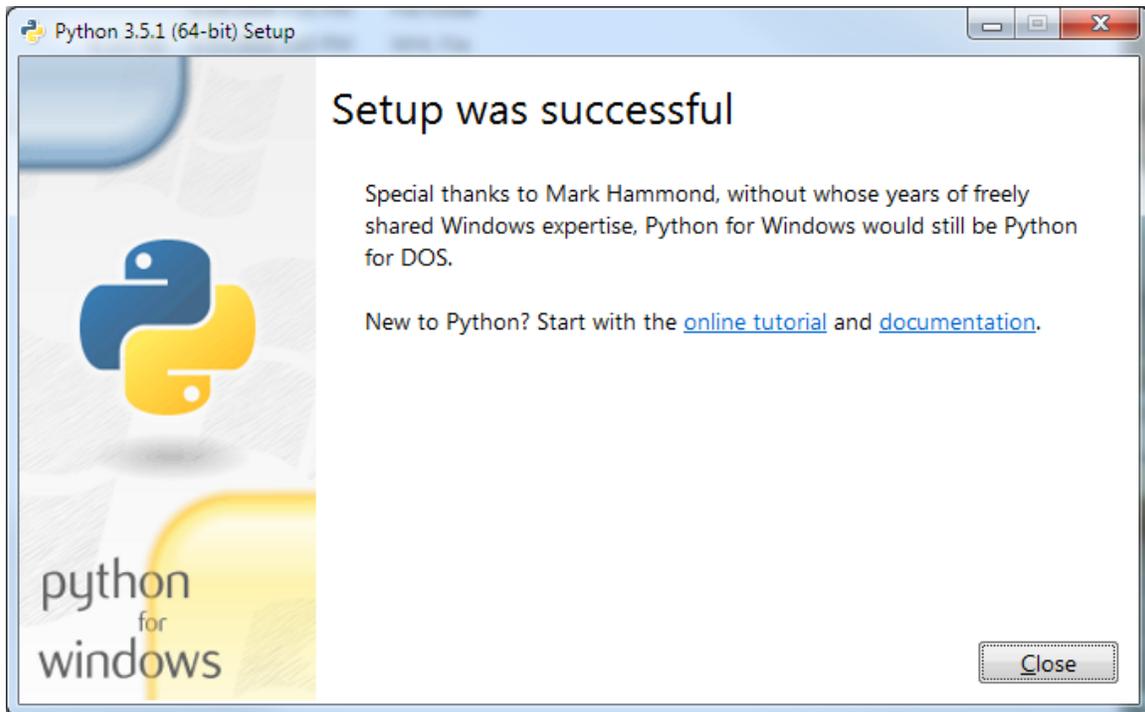


Figure 3

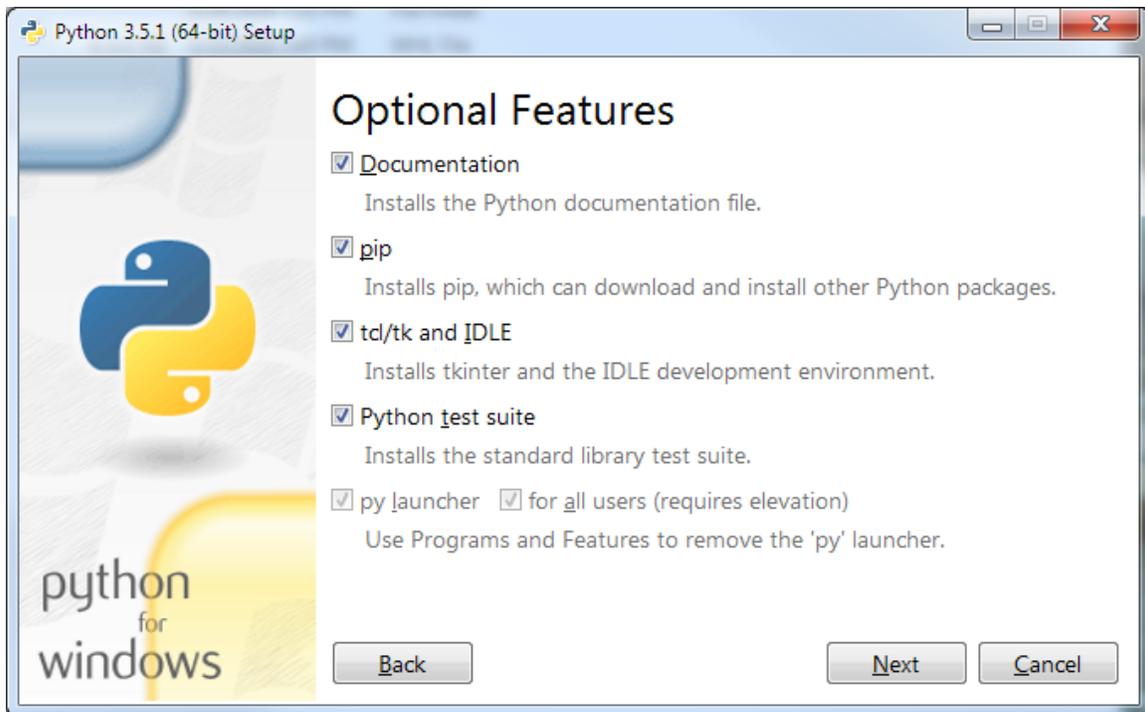


Figure 4

Be sure all of the boxes are checked, and then click *Next* to bring up Figure 5.

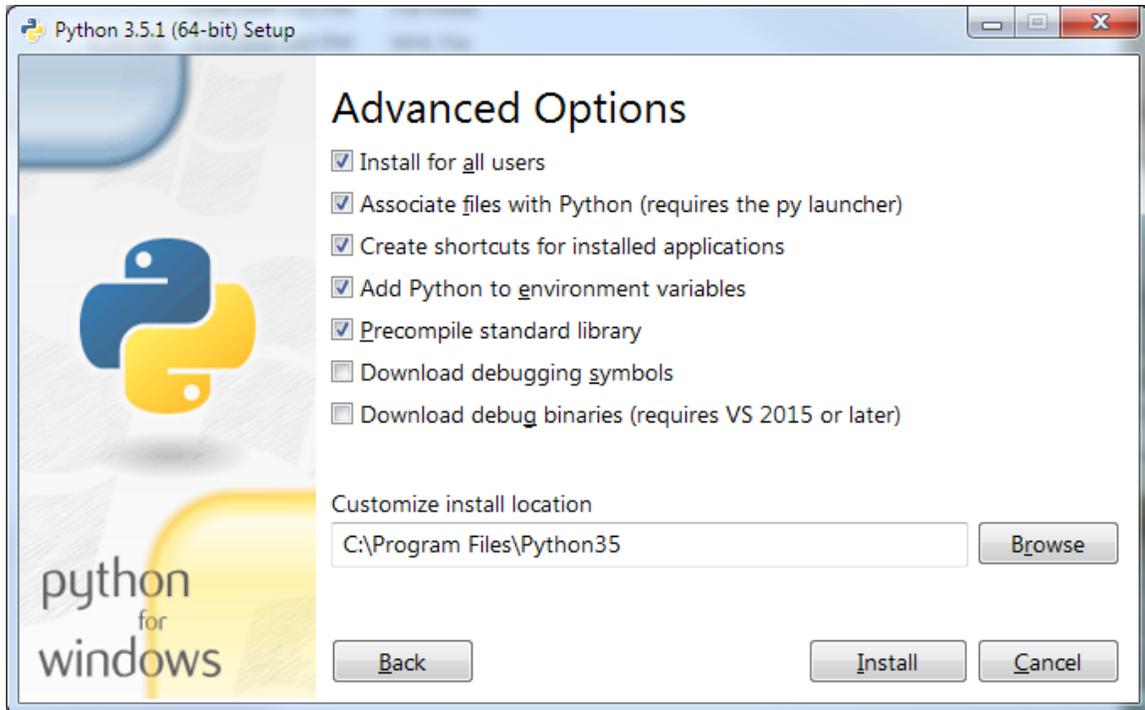


Figure 5

Click *Install for all users* in order to force the installer to choose a commonly accessible directory for all of the users of this computer. Be sure that the check boxes in your installation match those in Figure 5.

Click *Install* to begin the installation. If the installer tells you the directory already exists and asks if you are sure that you want to overwrite existing files, click *Yes*. The progress of the installation will be shown in the dialog box.

The installation will take several minutes and will finish with the dialog box of Figure 3 on page 4 of this document. Click *Close* to complete the installation of *Python 3.5.1*.

## Testing your installation

### *Testing on Windows 7*

To confirm and test your installation, we will start *IDLE*, the *Python Integrated Development Environment* window. If you are running *Windows 7*, click the *Start* button to bring up the *Windows Start* menu. Select *All Programs* and scroll down to *Python 3.5*. This is a folder shown circled in the left side of Figure 6.

When you open this folder, there will be four options, shown in the right side of Figure 6. Select *IDLE (Python 3.5 64-bit)* to start the *IDLE* program shown in Figure 7.

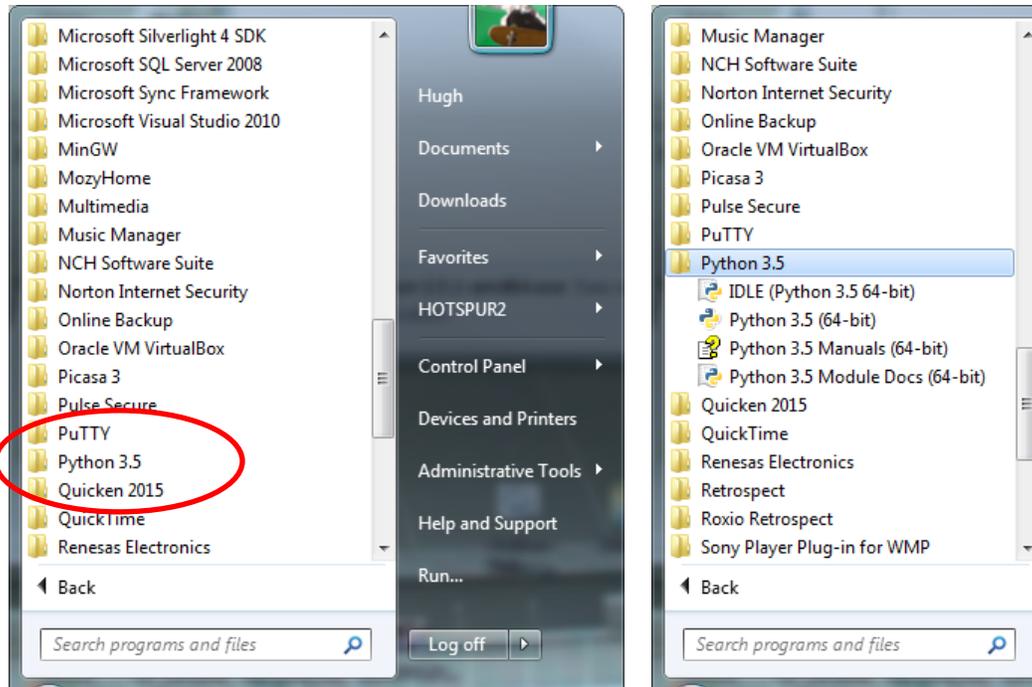


Figure 6

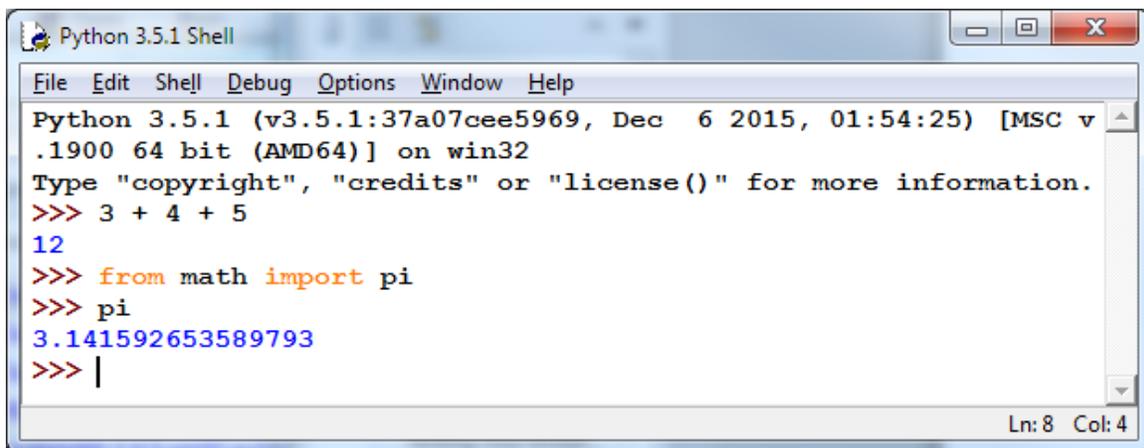


Figure 7

This is *IDLE*, the *Python* command prompt and graphical user interface. This is where we will start all programs and projects in this course. For now, simply type any *Python* statement or expression after the “>>>” prompt. For example, in Figure 6, the expression  $3 + 4 + 5$  was typed and *Python* responded with the value  $12$ .

After the next two “>>>” prompts, type the command

```
from math import pi
```

and the expression

```
pi
```

*Python* responds by printing the value of *pi* to 15 decimal places.

Continue testing by typing out the code on pages 10-11 of the textbook, just to make sure that your installation works as expected.

### Testing in Windows 8

Windows 8 does not have a *Start* button but rather a *Start* screen that is intended to make the user experience more like the smartphone experience. Unfortunately, when *Python* is installed as instructed above, its icon does not automatically appear on the *Start* screen. It also does not appear in the list of apps.

To find it, move the cursor to the upper-right or lower-right corner of the screen to expose the *Windows 8* pallet of “charms”. Select the *Search* charm to bring up a *Search* box. Type the word “Python.” This will bring up a list of matching items, similar to that shown in Figure 8 below but referring to the 64-bit version of *Python 3.5.1*.



Figure 8

Note that this list is similar to the *Python 3.5.1* folder in the Start Menu in Figure 5. *Right-click* on the item labeled *IDLE (Python 3.5.1 GUI)*. From the menu, select “Pin to Start” to cause an icon to be added to the *Start* screen. You may also want to pin the item to the *Task bar* (i.e., the bar of tiny icons at the bottom of the screen). You may also select “Open file location,” which will bring up the following window:–

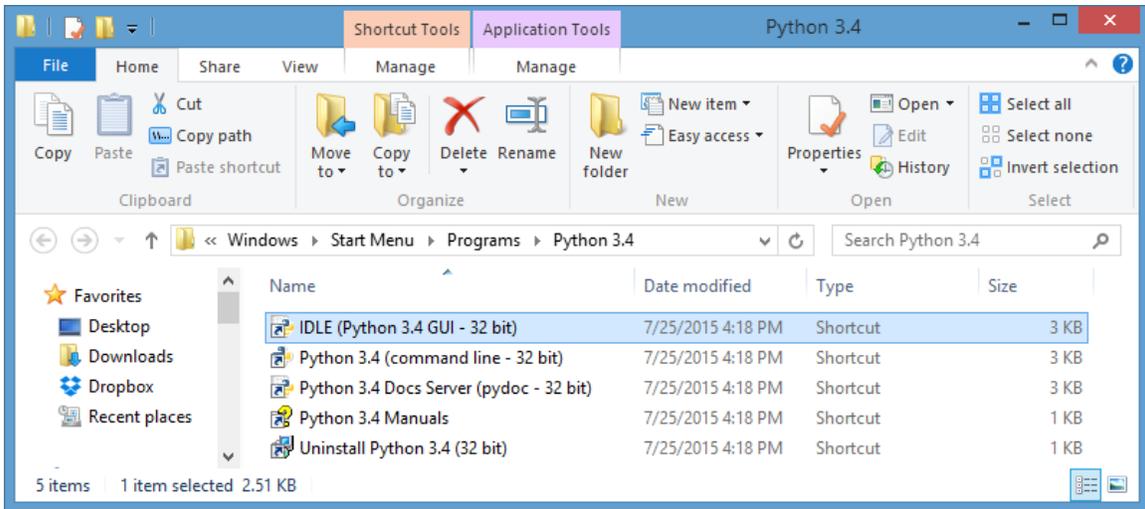


Figure 9

From this window, you can copy any or all of the *Python* links to the desktop.

To test your installation, double-click on the *IDLE (Python GUI)* icon and carry out the same tests as shown above under Figure 7.

*Testing under Windows 10*

Windows 10 restored something similar to the *Start* button, but it looks a little different —



something like  at the left end of the taskbar. Click on this to bring up a menu something like Figure 10 below:—



Figure 10

In most cases, *IDLE (Python 3.5 64-bit)* should be shown under “Recently added” (five lines up from the bottom of this Start Menu). If you don’t see it there, click on *All apps*, at the bottom of the *Start Menu*. This will provide an alphabetical list of all applications. *Python 3.5* appears under “P” and shows the same four options as the right side of Figure 6. Since you will be using this a lot, it is suggested that you right click on *IDLE (Python 3.5 64-bit)* and select either *Pin to Start* or *More > Pin to Taskbar*.

To test your installation, double-click on this icon and carry out the same tests as shown above under Figure 7.

Congratulations! You now have a usable *Python 3.5* running on your *Windows* computer or laptop.

## Installing graphics.py

*Graphics.py* is a simple drawing package that we will use a lot in this course. It was written in *Python 3* and created by the textbook author for making simple drawings. To install it, click on this link — [graphics.py](#) — and download the file *to the folder where your Python directory*. You should put the *graphics.py* to *<Python directory>/Lib directory*. *<Python directory>* is the directory where you install Python in figure 1.

## Installing *matplotlib*, *numpy*, and other packages

One of the many benefits of *Python* is the vast number of third-party packages<sup>5</sup> that can be downloaded and used by your *Python* programs. Many of these are open-source and free. For this course, we will use at least the following:–

- *matplotlib* (a package for creating 2D plots and graphs similar to *Matlab*),
- *numpy* (meaning “Numerical Python,” a package for efficient handling of large arrays of numerical data, also needed by *matplotlib*), and

Click on the following links to download the respective packages to a convenient folder:–

- [numpy-1.11.0-cp35-none-win\\_amd64.whl](#)
- [matplotlib-1.5.1-cp35-none-win\\_amd64.whl](#)

## Installing *numpy 1.11.0* on Windows

The *numpy* package needs to be installed immediately after you install *Python 3.5.1* itself. To install it, you must open a *Windows Command Prompt*.

A *command prompt* is a window into which you type “commands” to tell the computer and operating system what to do. An example *command prompt* window is shown in Figure 12.

In this window, the system prints a prompt starting with the “path” of the current folder and ending with a “>” character. After the prompt, you type a *command*, consisting of a command name followed by zero or more operands, which control what the command does. When you terminate the command with the *Enter* key, the system performs the command.

---

<sup>5</sup> Recent reports indicate that there are at least 45,000 different packages covering many fields of science, engineering, and mathematics.

A *command* may work silently and then print out its results in the same window, or it may engage in a textual conversation with you, requiring you to respond by typing, or it may open its own window with its own graphical user interface. When the command has completed, the system prints a new prompt for the next command. The command consisting solely of the word “**exit**” causes the system to close the command prompt window.

In order to install numpy and matplotlib, you must run the Command Prompt in *Administrator* mode.

- In *Windows 7*, click on the *Start* button, select *All Programs*, scroll to the *Accessories* folder and open it. This folder will show many useful accessories, such as shown in Figure 11.

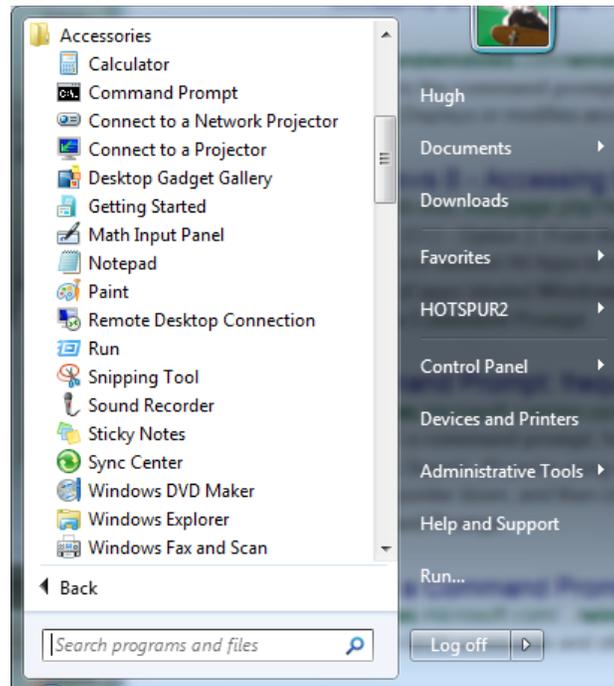


Figure 11

Right click on “Command Prompt” on the second line and select *Run as Administrator*. Windows will ask you if you really want to do this. Click yes. This will open a window into which you can type textual commands.

- In *Windows 8*, search for “Command Prompt” in the pallet of “charms,” the same way you searched in Figure 7. Drag an icon for the *Command Prompt* to your desktop, right-click on it, and select *Run as Administrator*.
- In *Windows 10*, click on the *Windows* icon at the left end of the taskbar, select *All apps*, and scroll down to *Windows system* (listed alphabetically). Open the *Windows system* item to expose a number of system applications, including *Command Prompt*. Right-click on *Command Prompt*, and select *More > Run as administrator*.

If you are unable to find the *Command Prompt* or to run it as *Administrator*, seek help from the Professor, one of the TAs, or the Helpdesk. If you are successful, you should be presented with a window resembling Figure 12 below. The background is likely to be black or some

other dark color. Note that the header of this window labels it as an *Administrator Command Prompt*.<sup>6</sup>

Next, open the folder where you downloaded

**numpy-1.11.0-cp35-none-win\_amd64.whl** and  
**matplotlib-1.5.1-cp35-none-win\_amd64.whl**.

See, for example, Figure 13 below.

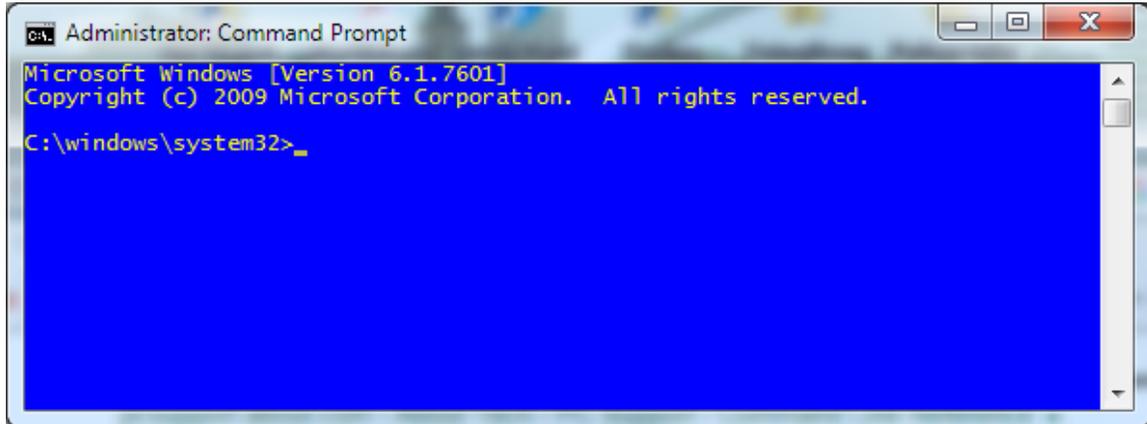


Figure 12

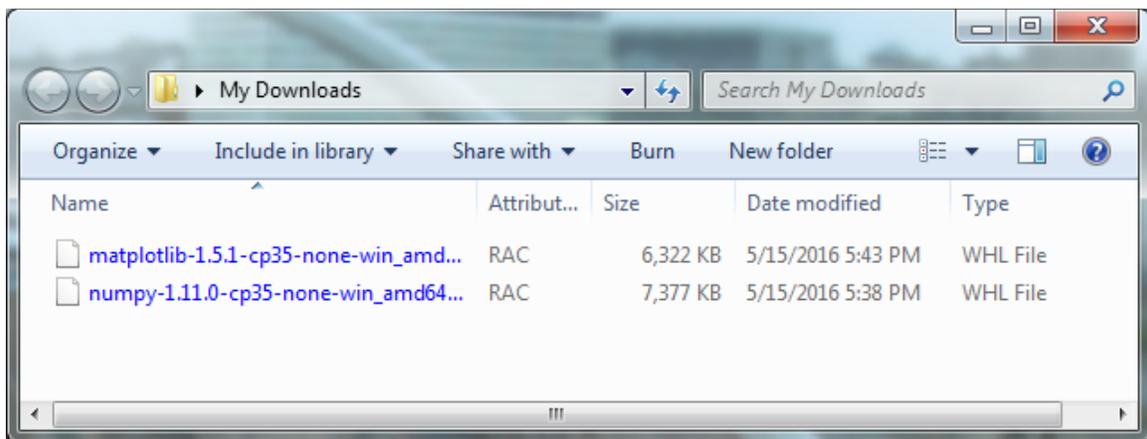


Figure 13

Right-click in the address bar of this folder (i. e., where it says *My Downloads* or whatever the name of your folder is), and select *Copy Address as text* from the resulting pop-up menu.

Next, in the *Command Prompt* window of Figure 12, type the command “**cd**” followed by a space, and then right click to paste the text form of the address that you “copied as text” in the previous paragraph — i.e.,

**cd** <paste directory address here>

The result should look something like

**cd C:\Users\Hugh\Desktop\My Downloads**

---

<sup>6</sup> This means that you have a lot of power to do good or evil to your computer!

but with the folder address of *your* download folder. Type the **ENTER** key. The “**cd**” command is the command to “change directory.” The result will be to point the command prompt to the place where you downloaded the files of Figure 13.

Next, type the command

```
pip install --upgrade pip
```

For Windows 10, use the following command instead:

```
python -m pip install --upgrade pip
```

The **pip** command is the *Python Installation Program*. This will upgrade the **pip** program and output something along the following lines in the command prompt window:–

```
C:\Users\Hugh\Desktop\My Downloads>pip install --upgrade pip  
Collecting pip  
Using cached pip-8.1.2-py2.py3-none-any.whl  
Installing collected packages: pip  
Found existing installation: pip 7.1.2  
Uninstalling pip-7.1.2:  
Successfully uninstalled pip-7.1.2
```

It may then report an error with a lot of error output, *all of which which you may safely ignore*. Please also be aware the version number may be different than yours.

If you type the command

```
pip list
```

it will list the programs that pip knows about — for example,

**DEPRECATION: The default format will switch to columns in the future. You can use --format=(legacy|columns) (or define a format=(legacy|columns) in your pip.conf under the [list] section) to disable this warning.**

```
C:\Users\Hugh\Desktop\My Downloads>pip list  
pip (8.1.2)  
setuptools (18.2)
```

You can ignore the warning message. Next, we will install **numpy**. Type the command

```
pip install numpy
```

Note that the **pip** program will contact the Internet to verify that it has all of the dependencies required by **numpy** and that they are all up-to-date. It should reply

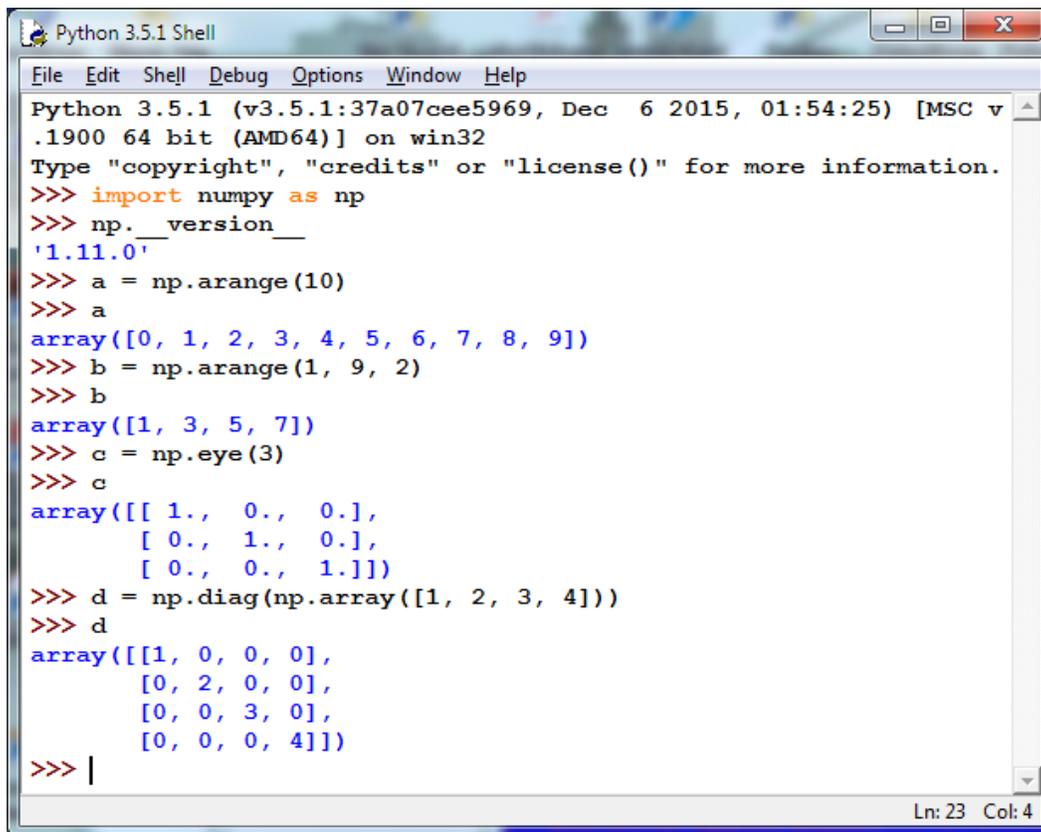
```
Collecting numpy  
Downloading numpy-1.11.3-cp35-none-win_amd64.whl (7.6MB)  
100% |#####| 7.6MB 148kB/s  
Installing collected packages: numpy  
Successfully installed numpy-1.11.3
```

When this installation is complete, make a quick test *numpy* by opening an *IDLE* window, as in Figure 6. Type or paste the following commands into *IDLE*, *one line at a time, exactly* as

written. In particular, the word **version** is preceded by *two* underscore characters and followed by two more. Also, the word **arange** is spelled with *one* “r”.<sup>7</sup>

```
import numpy as np
np.__version__
a = np.arange(10)
a
b = np.arange(1, 9, 2)
b
c = np.eye(3)
c
d = np.diag(np.array([1, 2, 3, 4]))
d
```

The result should resemble Figure 14:–



```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:54:25) [MSC v
.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import numpy as np
>>> np.__version__
'1.11.0'
>>> a = np.arange(10)
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> b = np.arange(1, 9, 2)
>>> b
array([1, 3, 5, 7])
>>> c = np.eye(3)
>>> c
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
>>> d = np.diag(np.array([1, 2, 3, 4]))
>>> d
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])
>>> |
```

Figure 14

We will carry out a more comprehensive test of *numpy* later in this document.

---

<sup>7</sup> This is a variation of the “**range**” function that we will learn about in the first week of class.

## Installing Matplotlib on Windows

Installing *matplotlib* requires several steps. To get started with *Matplotlib*, type the following command into the Administrator Command window of Figure 11.

```
pip install matplotlib
```

As with **numpy**, you may paste the file name into this command line. This installation will contact the Internet and download a bunch of supporting packages required by **matplotlib**. The result of the installation should resemble Figure 15.

```
C:\Users\HaoLoi>pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-1.5.3-cp35-cp35m-win_amd64.whl (6.5MB)
    100% |#####| 6.5MB 167kB/s
Collecting python-dateutil (from matplotlib)
  Downloading python_dateutil-2.6.0-py2.py3-none-any.whl (194kB)
    100% |#####| 194kB 1.3MB/s
Collecting pytz (from matplotlib)
  Downloading pytz-2016.10-py2.py3-none-any.whl (483kB)
    100% |#####| 491kB 1.3MB/s
Requirement already satisfied: numpy>=1.6 in
c:\users\haoloi\appdata\local\programs\python\python35\lib\site-packages (from
matplotlib)
Collecting pyparsing!=2.0.4,!2.1.2,>=1.5.6 (from matplotlib)
  Downloading pyparsing-2.1.10-py2.py3-none-any.whl (56kB)
    100% |#####| 61kB 2.0MB/s
Collecting cycler (from matplotlib)
  Downloading cycler-0.10.0-py2.py3-none-any.whl
Collecting six>=1.5 (from python-dateutil->matplotlib)
  Downloading six-1.10.0-py2.py3-none-any.whl
Installing collected packages: six, python-dateutil, pytz, pyparsing, cycler, matplotlib
Successfully installed cycler-0.10.0 matplotlib-1.5.3 pyparsing-2.1.10 python-dateutil-2.6.0
pytz-2016.10 six-1.10.0
```

Figure 15

Finally, type the following two commands

```
pip install nose
pip list
```

The **nose** command finds and installs a comprehensive test suite for **numpy**, and the last command lists the packages that are currently installed with your *Python 3.5.1* installation. The list should include the following:–

```
cycler (0.10.0)
matplotlib (1.5.1)
nose (1.3.7)
numpy (1.11.0)
pip (8.1.2)
pyparsing (2.1.4)
python-dateutil (2.5.3)
```

```
pytz (2016.4)
setuptools (18.2)
six (1.10.0)
```

Finally, type the **exit** command to close the Administrator Command Prompt.

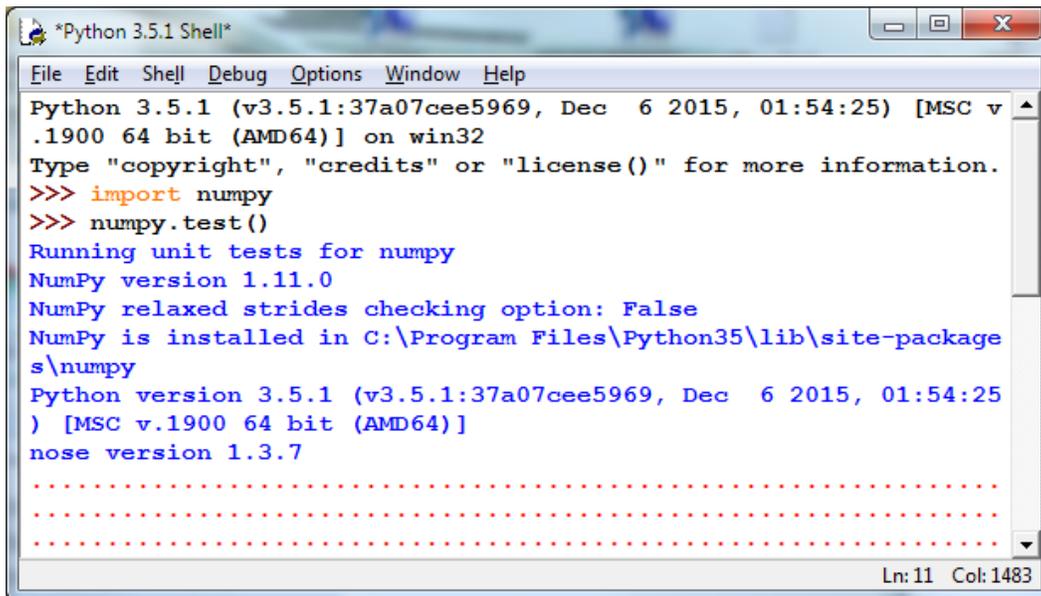
## Testing Your Installation

### Testing *numpy*

To carry out the comprehensive test of *numpy*, open a new *IDLE* window and type the following two commands:–

```
import numpy
numpy.test()
```

This uses **nose** to run the standard package of *numpy* tests for three or more minutes. It prints a bunch of stuff in the *IDLE* window. Although some of the output may look like error messages, these are known issues with the tests. The test should start with something resembling the following:–



```
*Python 3.5.1 Shell*
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:54:25) [MSC v
.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import numpy
>>> numpy.test()
Running unit tests for numpy
NumPy version 1.11.0
NumPy relaxed strides checking option: False
NumPy is installed in C:\Program Files\Python35\lib\site-packag
s\numpy
Python version 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:54:25
) [MSC v.1900 64 bit (AMD64)]
nose version 1.3.7
.....
.....
.....
Ln: 11 Col: 1483
```

Figure 16

After a few minutes, it should end with something resembling the following:–

```
.....
-----
Ran 5772 tests in 133.722s

OK (KNOWNFAIL=8, SKIP=12)
<nose.result.TextTestResult run=5772 errors=0 failures=0>
>>>
```

Figure 17

Congratulations! You have now installed a working versions *numpy 1.11.0* and *matplotlib 1.5.1*.

### Testing *matplotlib*

You can carry out a simple test of your *matplotlib* installation by typing or pasting the following commands into an IDLE window, one line at a time, *exactly* as written:–

```
from matplotlib import pyplot
pyplot.plot([1, 2, 3, 4], [1, 4, 9, 16])
pyplot.show()
```

The IDLE window should look something like the following:–

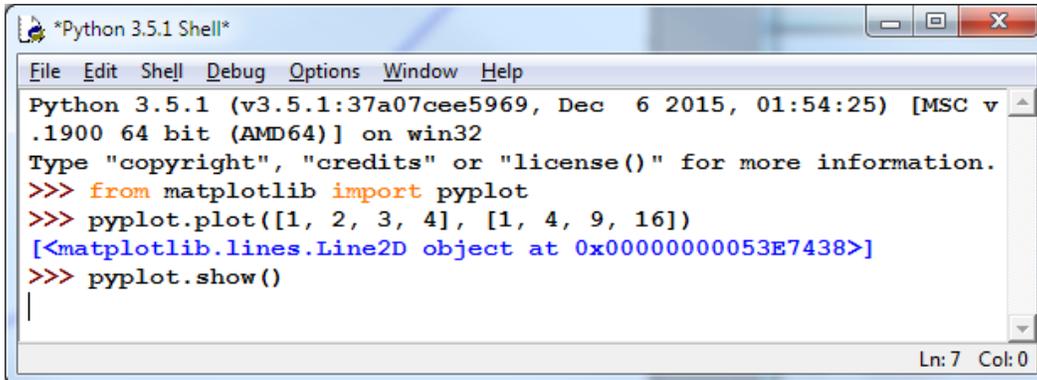


Figure 18

After you type the **ENTER** key following the last line, the following window should appear:–

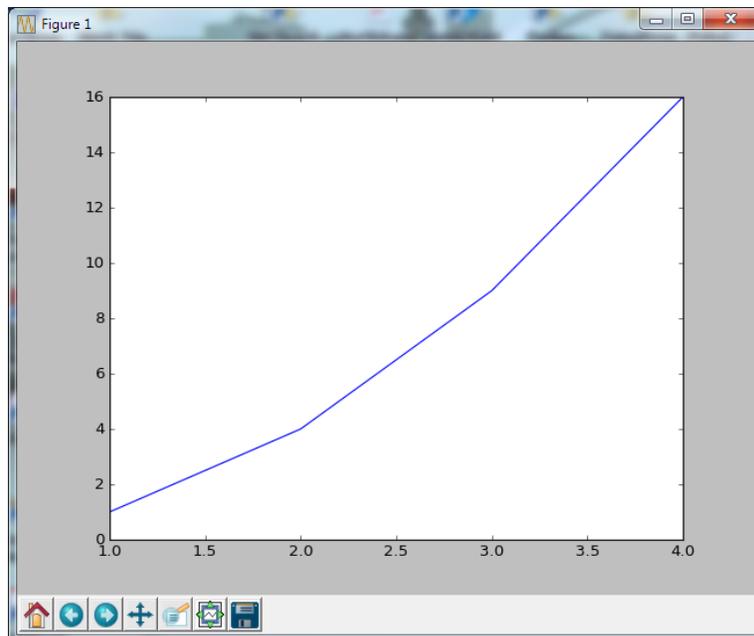


Figure 19

To close this window, click on the “close” button in the upper right.

For a more interesting test, download the following file:–

[TestMatplotlib2.py](#)

Use the *File* menu in the *IDLE* window to open this file, which resembles the following:–

```
TestMatplotlib2.py - C:\Hugh\WPT\CS-1004, A-term 2016\Resources\TestMatplotlib2.py (...)
```

```
File Edit Format Run Options Window Help
```

```
# TestMatplotlib2.py
```

```
from pylab import *
```

```
def testPyplot(low, high, incr):
```

```
    t = arange(low, high, incr)
```

```
    s = sin(2*pi*t)
```

```
    plot(t, s)
```

```
    xlabel('time (s)')
```

```
    ylabel('voltage (mV)')
```

```
    title('About as simple as it gets, folks')
```

```
    grid(True)
```

```
    savefig("test.png")
```

```
    show()
```

```
    return
```

```
if __name__ == '__main__':
```

```
    testPyplot(-1.0, 1.0, 0.001)
```

Ln: 1 Col: 0

Figure 20

Click the *Run > Run Module* command in the menu at the top of the window to produce the following window:–

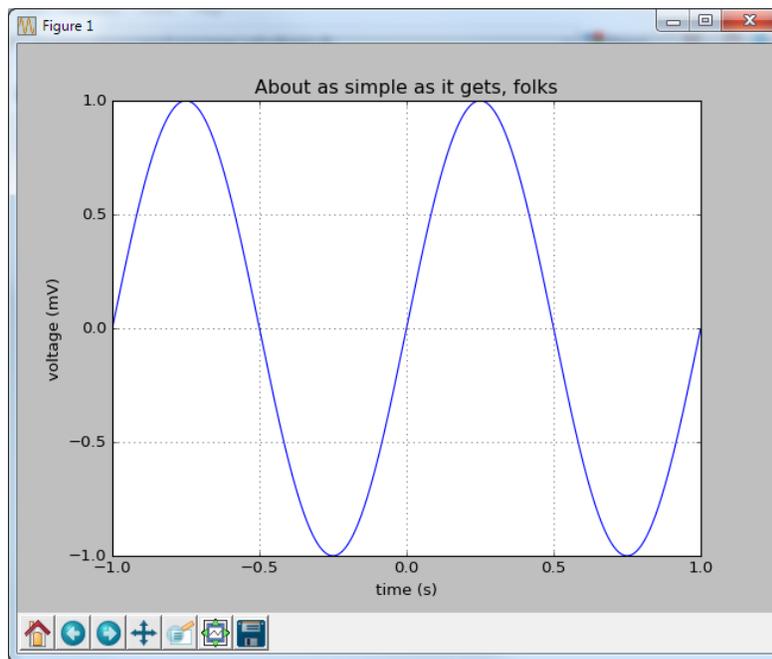


Figure 21

Congratulations! You now have a working version of *matplotlib* installed.

**Note:** Be sure to conduct these tests early in the term. There won't be enough time to discover problems and fix them when a homework assignment is due the next day.