

Econ 101 Honors Section

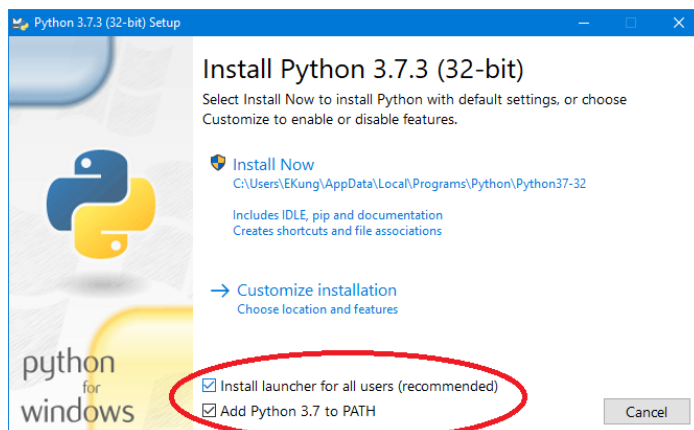
Installing and Running Python

We will use Python 3 for this class. I will show you how to install Python, how to run it interactively from the command line, how to install modules, and how to run scripts. We will do everything from the ground up—that is, without using any additional prepackaged software for Python programming, other than Python, your OS, and a basic text editor.

Installing Python

Follow these steps to download and install Python on your computer. These instructions are primarily for Windows 10. If you are using a different operating system, the instructions should be similar.

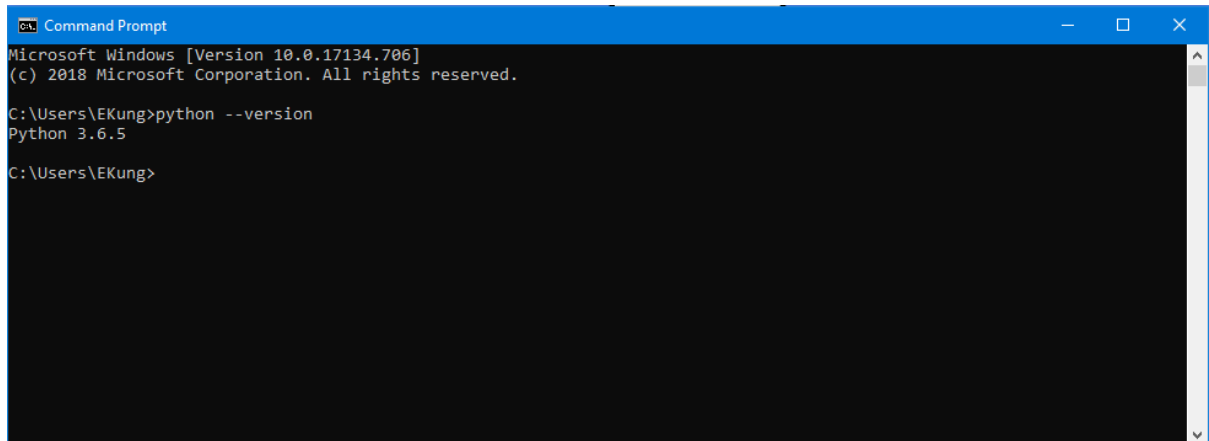
1. Go to www.python.org/downloads and follow the instructions for downloading and installing the latest version of Python for your OS. Unless you know what you're doing, **make sure to check “Add Python to PATH” during installation**. This ensures that your computer knows where to look when you tell it to run Python from your command line.



2. Let's make sure you have Python 3 properly installed. On Windows 10, open the command prompt by typing `cmd` into the search bar. On Mac, open your terminal. Type `python --version` into the command line. It should show "Python 3.x.x". As long as you have some version of Python 3 installed, you should be fine. If it shows "Python 2.x.x", then you had a pre-existing installation of Python 2. Try entering `python3 --version` instead. If that still doesn't work, see me for further help.



... Typing cmd into the search bar...



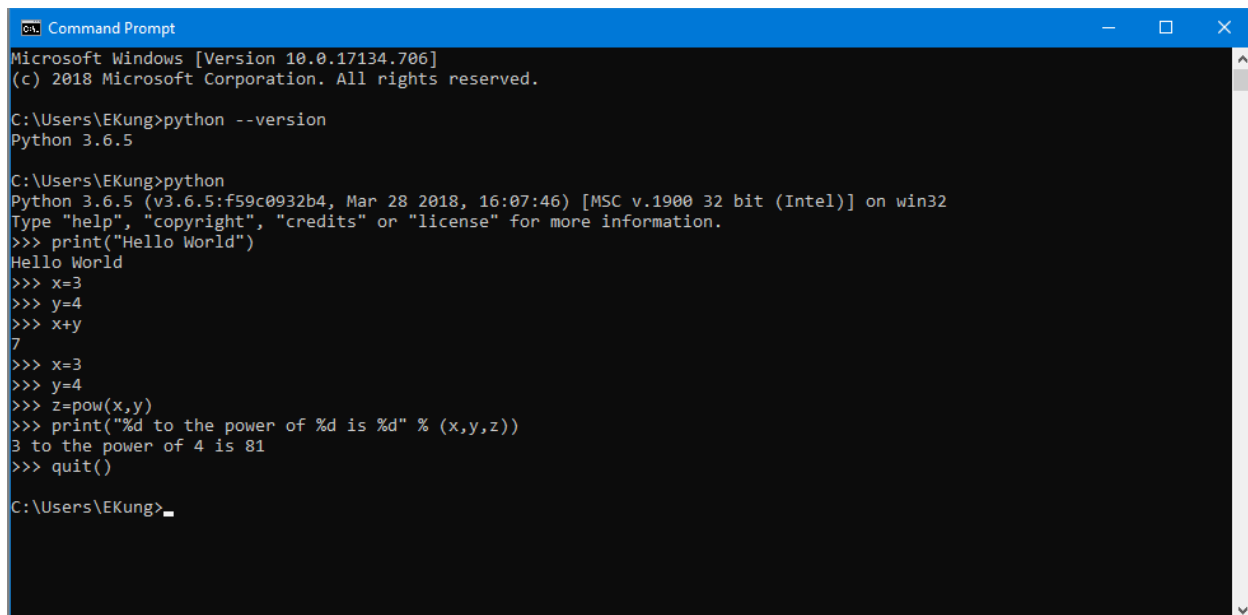
... What the command prompt should look like...

Running Python Interactively

Python can be run **interactively**, meaning you can type commands into Python and get a response immediately. To try this, start Python by typing `python`, or `python3` if `python` didn't work, into the command line. Python will start an **interactive session**, first displaying basic information, then waiting for you to enter a command at the command prompt which looks like `>>>`. Try the following commands:

1. `print("Hello World")`
2. `x=3`
`y=4`
`x+y`
3. `x=3`
`y=4`
`z=pow(x,y)`
`print("%d to the power of %d is %d" % (x,y,z))`
4. `quit()`

Play around with the interactive session some more and try various commands. See if you can make Python do division and multiplication. See if you can get it write something like "`X divided by Y is Z`" for arbitrary values of X and Y.

A screenshot of a Windows Command Prompt window. The title bar is blue and says "Command Prompt". The window content is black with white text. It shows the output of running 'python --version' (Python 3.6.5) and an interactive Python session. The session includes commands for printing 'Hello World', calculating 3+4, and calculating 3 to the power of 4 using pow().

```
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\EKung>python --version
Python 3.6.5

C:\Users\EKung>python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>> x=3
>>> y=4
>>> x+y
7
>>> x=3
>>> y=4
>>> z=pow(x,y)
>>> print("%d to the power of %d is %d" % (x,y,z))
3 to the power of 4 is 81
>>> quit()

C:\Users\EKung>
```

... results from my interactive session ...

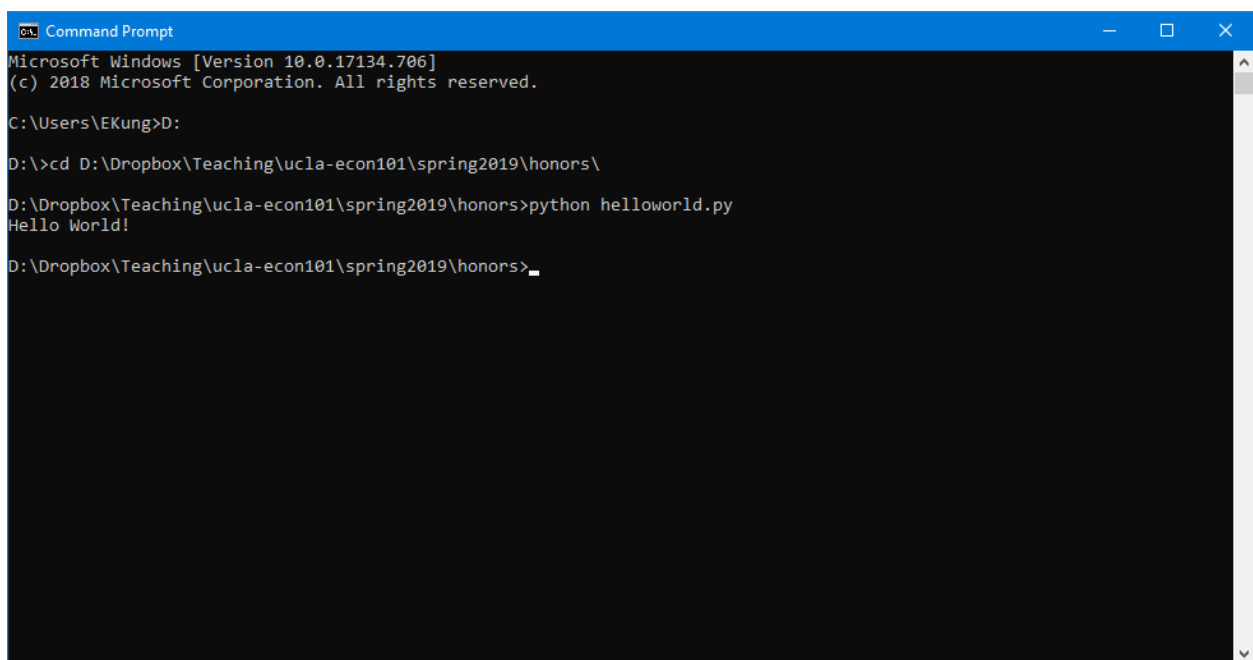
Writing Scripts

Using Python interactively is handy. We can input a set of commands and get immediate results. However, if the set of commands to get the result you want is very long and complicated, you might want to save those commands so you don't forget. You may also want to be able to reproduce your results on a different set of inputs. Or you may want to transmit your list of commands to another person.

A list of Python commands saved in a file, usually with file extension .py is called a **script**. Let's now write a simple script that prints "Hello World!" on the screen.

1. First, choose a directory (or folder) you want to save the script in. I chose `D:\Dropbox\Teaching\ucla-econ101\spring2019\honors`.
2. Open up a plain text editor. For Windows, I recommend the free **Notepad++**, but you can use anything, even the built-in **Notepad**. For Mac, you can use **BBEdit** or the built-in one. The important thing is that it saves plain text files, so Microsoft Word is *not* a plain text editor.
3. In the text editor, simply write the line `print("Hello World!")`, then save it as `helloworld.py` in your destination folder.

4. Now, go back to your command prompt. Type in `cd D:\Dropbox\Teaching\ucla-econ101\spring2019\honors` (or whatever your folder is) to go to your folder. If your folder is in a different drive from the default drive (i.e. D: instead of C:), you'll need to change drives first by typing `D:.`
5. Once you're in the folder that contains the script, type `python helloworld.py`. Python will run the script and print out "Hello World!" on the screen.
6. Try writing other scripts. For example, try writing a script that adds or subtracts two numbers.



```
Command Prompt
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\EKung>D:

D:\>cd D:\Dropbox\Teaching\ucla-econ101\spring2019\honors\
D:\Dropbox\Teaching\ucla-econ101\spring2019\honors>python helloworld.py
Hello World!
D:\Dropbox\Teaching\ucla-econ101\spring2019\honors>_
```

... running helloworld.py ...

Installing Modules

Python 3 comes with some basic functionality, like doing basic arithmetic. However, we are going to need some more advanced capabilities, like doing matrix algebra, which Python doesn't come with. Thankfully, Python has a very active open source community that develops new **modules** for Python that expand its basic functionality. One of the modules we'll use in this course is called **numpy**, which offers enhanced mathematical capabilities.

Python comes with a handy tool for installing modules called **pip**. To install a module, make sure you are at your command prompt (not an interactive session). Then type `pip install numpy` or the name of the module you want to install. If `pip` didn't work, try `pip3`. Pip will automatically check an online repository and download the module if it finds it.

Using numpy

Now let's try to solve a basic matrix equation with numpy. The matrix equation we're trying to solve is:

$$\begin{pmatrix} 3 & 2 \\ -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

which we write as: $\mathbf{Mx} = \mathbf{y}$ and therefore $\mathbf{x} = \mathbf{M}^{-1}\mathbf{y}$.

You may already know how to solve this by hand. But let's solve it using numpy in a script. Try writing a script with the following commands.

```
import numpy

M = numpy.zeros((2,2)) # initializes M as a 2x2 matrix of zeros
y = numpy.zeros(2) # initializes y as a length 2 vector

# Fill in the numbers for M
M[0,0] = 3
M[0,1] = 2
M[1,0] = -1
M[1,1] = 4

# Fill in the numbers for y
y[0] = 5
y[1] = 6

# Solve x = M^-1*y
x = numpy.linalg.solve(M,y)

# Print the solution
print("x1 = %g" % x[0])
print("x2 = %g" % x[1])
```

Comments in Python

In Python, any code that follows a hashtag (#) is called a **comment**.

A comment is used by coders to communicate what they are trying to do with each line of code.

Python ignores comments when running a script.

Save this as a script, i.e. `usingnumpy.py` in your folder, then run it. You should get that $x_1 = 0.571429$ and $x_2 = 1.64286$.

Using for-loops in Python

One of the most basic tasks in computer programming is to tell the computer to repeat a task multiple times. For example, you might want the computer to go through a list of students and find all the students with a GPA below a threshold and send them an automatic warning email. One way to do this is with **for-loops**. In programming, a for-loop is a line of code that tells the computer to repeat a set of commands *for* each item in a list.

Here's an example:

```
for name in ["Ashley", "Bob", "Charles"]:  
    print(name)
```

This code will print "Ashley", "Bob" and "Charles" to the screen. The for loop repeats all the commands that are **tabbed and under the colon**, for each item in the list. The list here is the list of names, Ashley, Bob, and Charles, and `name` is the placeholder variable that will reference the current item in the list as we go through them.

Let's give an example where we make the computer count from 0 to 10, and then to count by 2's from 0 to 100.

```
for i in range(11):  
    print(i)  
  
for i in range(51):  
    print(2*i)
```

Here, `range(n)` is a list of `n` numbers, starting from 0 and going to `n-1`.

We can also do **nested for-loops**. These are for-loops inside for-loops. To make sure that they're inside, just make sure they are tabbed. Once you stop tabbing the commands, Python assumes the for-loop has ended.

```
for i in range(11):  
    print(i)  
    for j in range(51):  
        print(2*j)
```

This produces a different result from the previous code because the second for-loop is inside the first for-loop. Note also that we've switched `i` to `j` in the inner for-loop, because `i` is still being used by the outer for-loop.