

Installing and Testing Python 3.x on your Desktop

Overview

This document provides an overview of how to download and install the latest version of Python on your desktop. This document describes how to use a file for editing and demonstrate the use of an Integrated Development Environment (IDE). Finally, the last section shows how to install pylint and use this tool to validate your code is compliant with acceptable professional coding styles.

Be aware new versions of Python are released on a regular basis resulting in some screen captures found in this document not being aligned perfectly with your experience. However; the basic steps will be quite similar.

What about my Mac?

Python works nicely on your Mac as well. Just be sure to download the Mac OSX version to match your machine.

Steps

1. Download the latest version of Python 3 to your desktop. Go to <https://www.python.org/downloads/> and select “Download Python 3.x.x” to download the installation file. See figure 1.



Figure 1 Downloading the Python Installation Software

You will be prompted to save the files to your desktop.

Note, 32-bit will work on 64-bit systems. If you do have a 64-bit Windows installation, you can look through the latest Windows Python and specifically select the x86-64 version.

2. Double click to install the Python

Navigate to the file you recently downloaded and double click to install Python. A screen similar to figure 2 will be presented. Be sure to select “Install Launcher for all Users” and Add Python 3.x to Path” options.



Figure 2 Installing Python

Select “Install Now” to continue. After accepting any security warnings the Python setup process will continue. (See figure 3.)

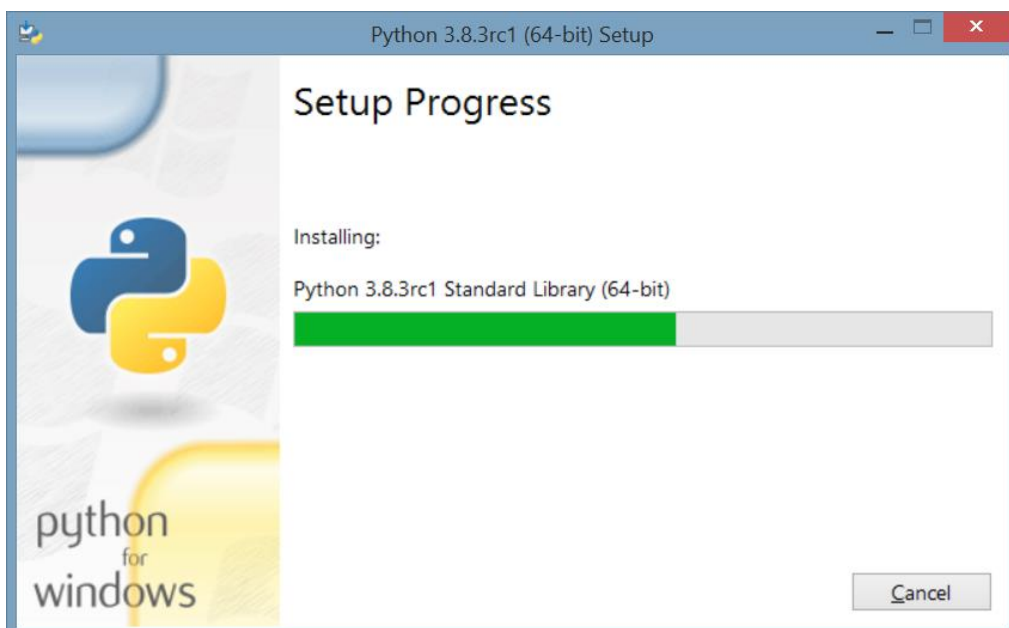


Figure 3 Python Setup Progress

3. Python set-up is complete

Upon successful installation, you will see a screen similar to the display in figure 4.



Figure 4 Successful Installation

Click "Close" to continue.

4. Test your Python environment

To make sure the install worked properly, open a command prompt and enter type python. Figure 5 shows the results of the Python environment running properly. Note, your version number will most likely be different but it should be 3.x.x where x.x could vary.

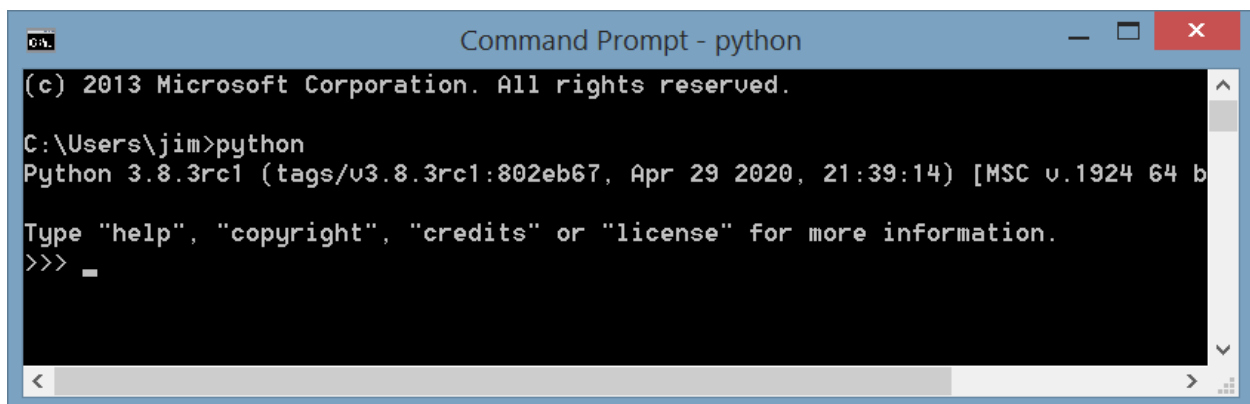
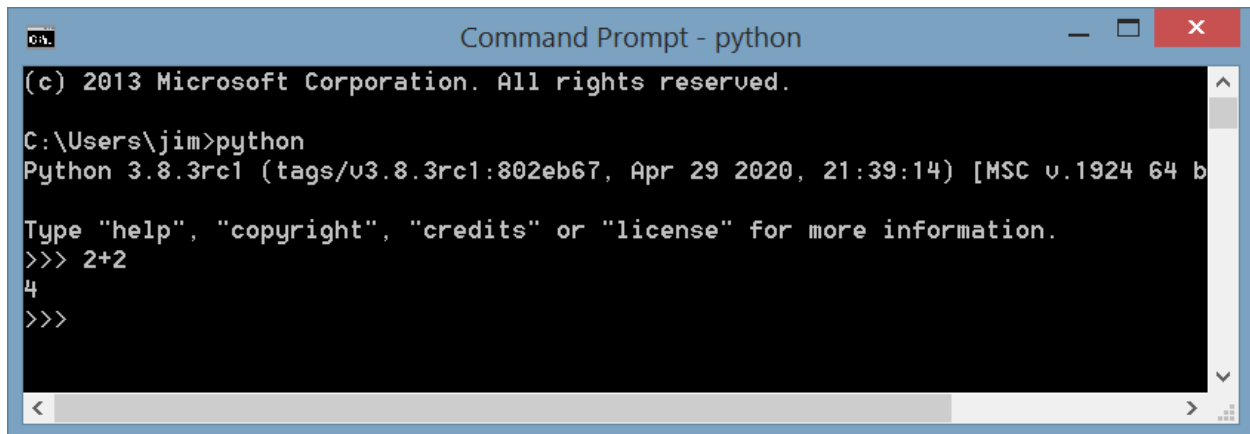


Figure 5 Launching Python from the Command Prompt

For a quick test, beyond the launching of the Python environment, use the prompt as a simple calculator. For example, enter 2+2 to show you can start doing simple statements in the environment right away. (See figure 6).



```
Command Prompt - python
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\jim>python
Python 3.8.3rc1 (tags/v3.8.3rc1:802eb67, Apr 29 2020, 21:39:14) [MSC v.1924 64 b
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+2
4
>>>
```

Figure 6 2+2 Test

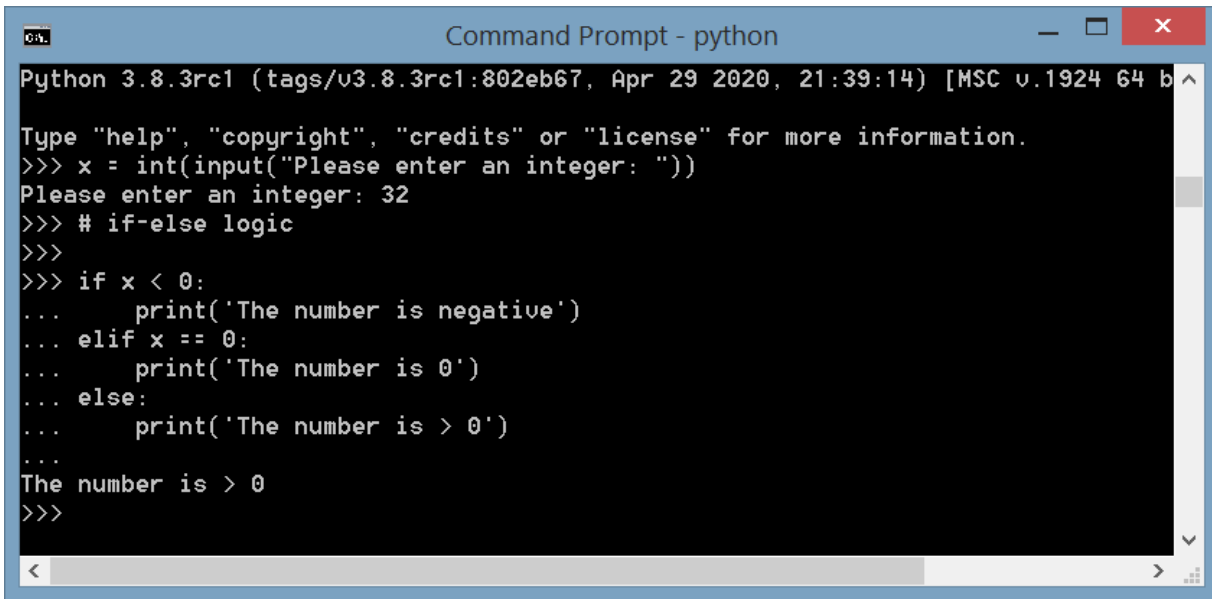
You can type any statement directly into the environment. In fact, multiple statements also work. The interpreter knows to wait until the end of the statement is reached before executing.

The following simple Python example prompts a user to enter a number, displays the number and then determines if it is less than 0, greater than 0, or equal to zero.

```
x = int(input("Please enter an integer: "))

print('You entered:' + str(x))
# if-else logic
if x < 0:
    print('The number is negative')
elif x == 0:
    print('The number is 0')
else:
    print('The number is > 0')
```

Figure 7 illustrates entering this code into the Python environment one line at a time.

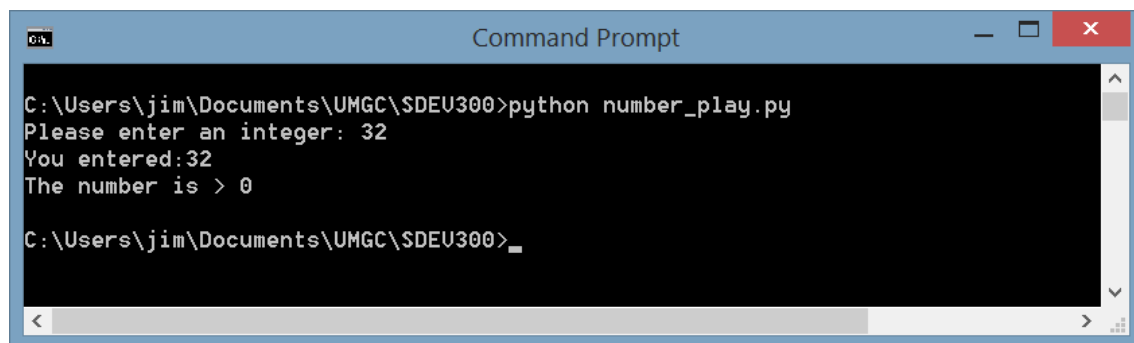


```
Python 3.8.3rc1 (tags/v3.8.3rc1:802eb67, Apr 29 2020, 21:39:14) [MSC v.1924 64 b
Type "help", "copyright", "credits" or "license" for more information.
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 32
>>> # if-else logic
>>>
>>> if x < 0:
...     print('The number is negative')
... elif x == 0:
...     print('The number is 0')
... else:
...     print('The number is > 0')
...
The number is > 0
>>>
```

Figure 7 Copying Code into the environment

Although not particularly efficient, you can enter as many lines of code into the environment as you need to accomplish your task at hand. However, you can (and should) use a text editor to enter the code, save the file and then load the file and execute the code all at once. For example, if the code demonstrated above was saved to a file named `number_play.py` we could execute the code by typing the following at the command prompt:

```
python number_play.py
```



```
C:\Users\jim\Documents\UMGC\SDEV300>python number_play.py
Please enter an integer: 32
You entered:32
The number is > 0

C:\Users\jim\Documents\UMGC\SDEV300>_
```

Figure 8 Running `number_play.py`

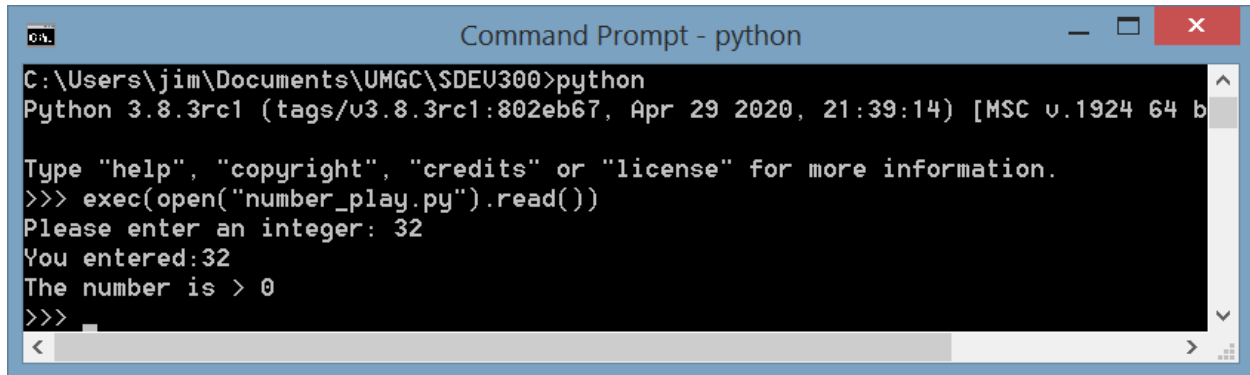
Important, you need to change to the directory where the Python is located to run without a directory or folder structure. If you aren't in the location where the directory is found, you will need to include the path. For example:

```
python c:\mydocuments\UMGC\SDEV300\number_play.py
```

If you want to run a file you have created with the Python environment already open, you can use the `exec()` function. For example:

```
exec(open("number_play.py").read())
```

Figure 9 illustrates the use of the `exec()` function call within the Python environment.



```
Command Prompt - python
C:\Users\jim\Documents\UMGC\SDEU300>python
Python 3.8.3rc1 (tags/v3.8.3rc1:802eb67, Apr 29 2020, 21:39:14) [MSC v.1924 64 b...
Type "help", "copyright", "credits" or "license" for more information.
>>> exec(open("number_play.py").read())
Please enter an integer: 32
You entered:32
The number is > 0
>>>
```

Figure 9 Using `exec()` in the Python environment

The ability to use files makes coding much easier. You can select your favorite texteditor (Notepad, Notepad++, Ultraedit...) and just start typing away. The use of an IDE makes coding even easier as code is often color coded and includes smart text entry and other features that can be real time savers for debugging. However, a texteditor of your choice is all that is required.

There are multiple free IDEs that support Python available for download on the Internet. Feel free to grab one that you like and use it for this course.

Figure 10 shows the same code using the community edition or PyCharm.

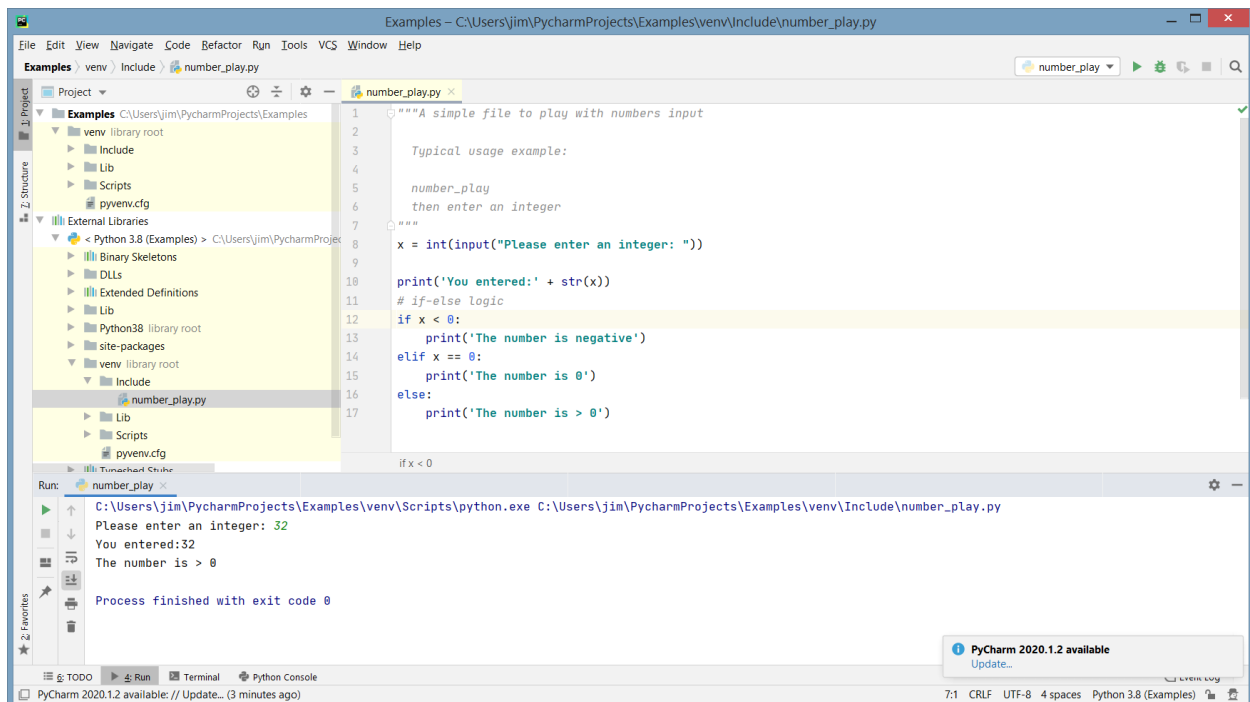


Figure 10 Community Edition of PyCharm

There will be a learning curve associated with navigating and using an IDE. Popular Python IDEs include Eclipse, Spider, Cloud9, Atom, PyCharm and others.

5. Use of the book examples.

The textbook is geared more towards showing line-by-line entry as opposed to editing a file and then running that file. You should start right away using the file approach. It will save you significant time and frustration as you work on the projects and assignments. Just type the syntax into a file as opposed to the Python command. Save the file, change to the directory where the file is located and then run the file using:

```
python my_file.py
```

As you debug, you can edit the code and then rerun at any time.

From your previous programming courses, you should be very comfortable with the concepts of selection (if/else), assignment (x=3), and repetition (for,while) statements as well as logic flow, arrays, functions, data structures and objects. Python has all of these programming components and they work the same way. They just have slightly different syntax and processes to use them correctly.

Each week you will be given you the opportunity to demonstrate your Python learning.

6. Installing packages - pylint

Once Python is installed, it is quite easy to install other packages. Packages are prepare libraries that allow you to immediately start using additional functionality with simple function calls. This course will provide more details through the semester.

As a brief introduction, consider pylint, a python style guide checker, we will be using extensively in this course to make sure your code is well-documented and uses acceptable formatting and style standards. Pylint can be installed by running the following command at the command prompt:

```
pip install pylint
```

Once installed, you can check any file for compliance by navigating to the location of the python file and running the command:

```
pylint my_file.py > my_file.txt
```

In this example, the python file name my_file.py is analyzed by pylint with the results being output to my_file.txt. Figure 11 shows how to the results of the command being run on number_play.py.

```
-----  
Your code has been rated at 10.00/10
```

This is the goal for each of your submissions –to be rate 10 out of 10. In this case, I had to modify the code to add docstring to earn this rating. This was the feedback from a previous run

```
number_play.py:1:0: C0114: Missing module docstring (missing-  
module-docstring).
```

The completed code with all recommended adjustments is shown below:

```
"""A simple file to play with numbers input

Typical usage example:

number_play
then enter an integer
"""
x = int(input("Please enter an integer: "))

print('You entered:' + str(x))

# if-else logic
if x < 0:
    print('The number is negative')
elif x == 0:
    print('The number is 0')
else:
    print('The number is > 0')
```

It takes some time to determine exactly how to modify the Python files to earn the 10 rating. Documentation related to the google recommended style guide rules as well as the pylint validation rules are found here:

- <https://www.pylint.org/>
- <https://google.github.io/styleguide/pyguide.html>

As you can see from these documents, there are a significant amount of rules. Hence, the value of automating the checks using pylint. Each week you will have more practice using pylint and editing the code to be in compliance with professional style guides.