Learning Python for Numerical Analysis

Kenny Erleben

2017

Appendix A

Getting Python to Work

This book was written to support students to learn sufficient amount of Python program to be able to complete the programming assignments for the course "Introduction to Numerical Analysis" (NumIntro) offered at the Department of Mathematical Sciences, University of Copenhagen. In this book we expect that students are familiar with a computer and the specific operating system running on that computer. Hence, We take it for granted that students know what a terminal is and have skills for installing new software on their own computer.

This chapter is intended as an extra service for those students that need a little help to get started on installing Python on their own computer. If more help is needed we encourage talking to the teaching assistants or with the SCIENCE-IT support.

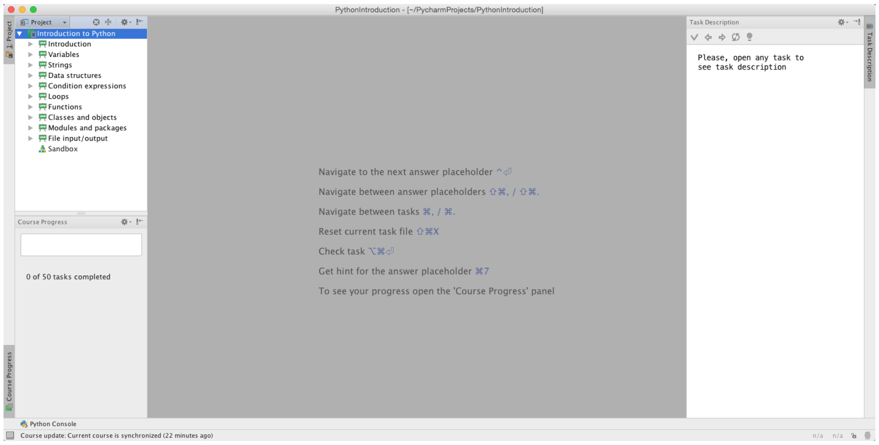
A chapter such as this are outdated almost before it is written. Software versions are continuously upgraded and it would be quite impossible to describe all possible combinations and possibilities of using different operation systems and software. We have chosen to focus on a small subset of common operating systems and warn students reading this book that there could be deviations depending on their versions and software used.

Thanks to Jan Phillip Solovej for providing the Windows description and Niels Martin Møller for the general notes.

A.1 Software Needed for This Course

A.1.1 PyCharm Edu

We recommend the PyCharm Edu software for students who have never done any kind of programming. Those familiar with programming may use other environments as they please. However, two integrated development environments are particularly useful for this course



Navigate to the next answer placeholder ^⏎

Navigate between answer placeholders ⇧⌘, / ⇧⌘.

Navigate between tasks ⌘, / ⌘.

Reset current task file ⇧⌘X

Check task ⌥⌘⏎

Get hint for the answer placeholder ⌘7

To see your progress open the 'Course Progress' panel

0 of 50 tasks completed

Please, open any task to
see task description

Course update: Current course is synchronized (22 minutes ago)

Figure A.1: Screenshot of the PyCharm Edu user interface.

A.1.2 PyCharm Community Edition (CE)

The PyCharm Community Edition (CE) is like the Edu version, except its interface displays more complexity and no "learning tasks" are available. Besides being cross-platform like the Edu version, PyCharm CE comes with extensive online help and a great many online videos for how to use the environment. Check out the documentation and videos from the PyCharm website.

A.1.3 Spyder

If one dislikes PyCharm another open source cross-platform environment is Spyder.

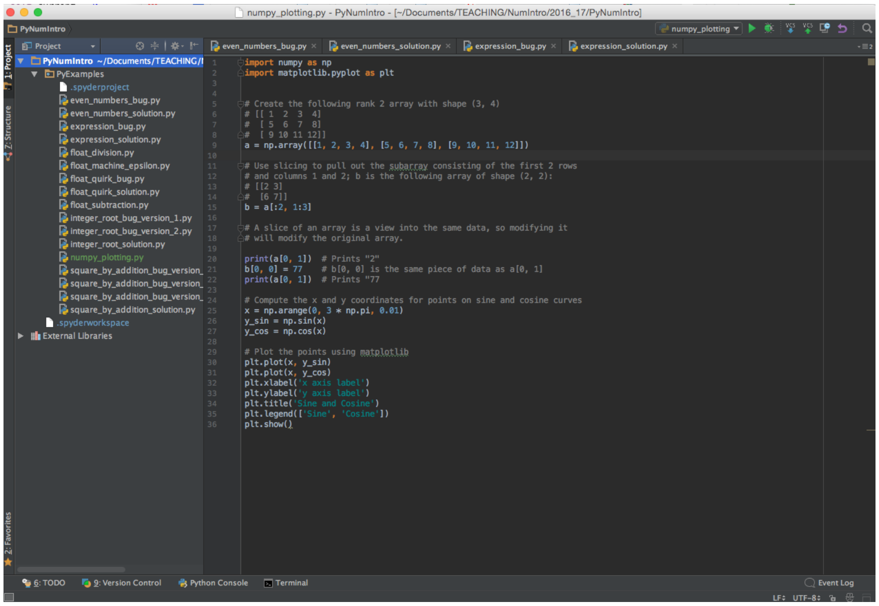
`https://github.com/spyder-ide/spyder`

Spyder is tuned to work with NumPy and Matplotlib Python packages, which makes it convenient for a course in numerics. Spyder does not come with a special learning mode like PyCharm EDU, nor does it have any "learning tasks" or as many official videos for learning the tool. If one is familiar with programming it does not take more than a couple of minutes to get used to this environment.

A.2 Mac OS X Yosemite (ver 10.10.5)

There are many ways to get Python installed on a Mac. Here we describe

- How to use MacPorts to install the Python interpreter and necessary packages for the NumIntro course
- How to use the Python shell to test if your installation work



```
import numpy as np
import matplotlib.pyplot as plt


# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# A slice of an array is a view into the same data, so modifying it
# will modify the original array.

print(a[0, 1])  # Prints "2"
b[0, 0] = 77    # b[0, 0] is the same piece of data as a[0, 1]
print(a[0, 1])  # Prints "77

# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.01)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```

Figure A.2: Screenshot of the PyCharm Community Edition (CE) user interface.

- How to download PyCharm EDU and install it
- How to setup the default Python interpreter used in PyCharm Edu
- How to test if the Python interpreter used in PyCharm EDU works

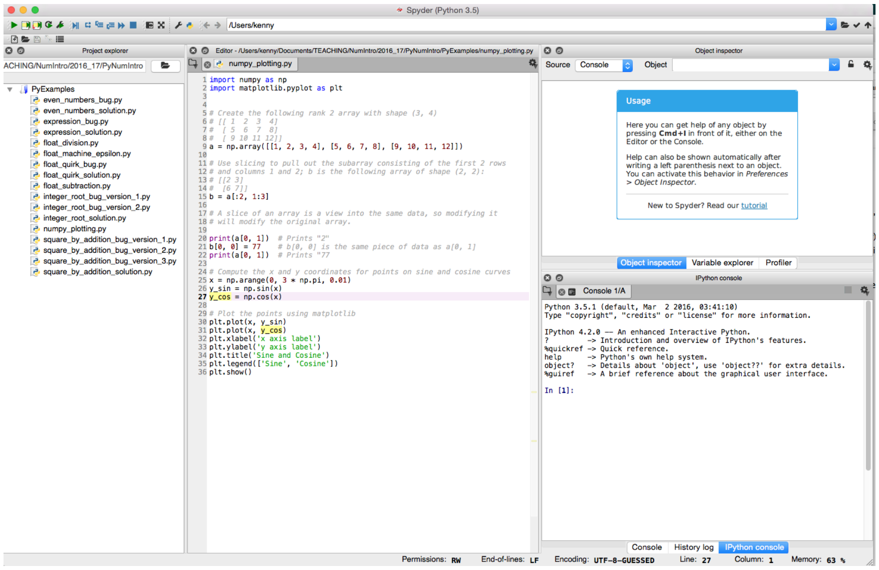
We use MacPorts to install Python on our computer and the Python packages we need for the NumIntro course. First you open up a terminal and write at the prompt something like this

```
sudo port install python35
sudo port install py35−numpy
sudo port install py35−matplotlib
```

There might be some slight changes depending on your specific installation and operating system version. There might also be more recent python version available to you. This book was written mostly using Python version 3.5. Any 3-version should work fine, so do not panic if your version is not exactly the same as seen in this book.

After having installed a “proper” version of the Python interpreter then it is time to check that it is running as it should. You do this by opening a Python shell. This is how you get a Python Shell. First you open a terminal and at the prompt you write

```
python3.5
```



```
import numpy as np
import matplotlib.pyplot as plt


# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# A slice of an array is a view into the same data, so modifying it
# will modify the original array.

print(a[0, 1])  # Prints "2"
b[0, 0] = 77    # b[0, 0] is the same piece of data as a[0, 1]
print(a[0, 1])  # Prints "77

# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.01)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Object Inspector*.

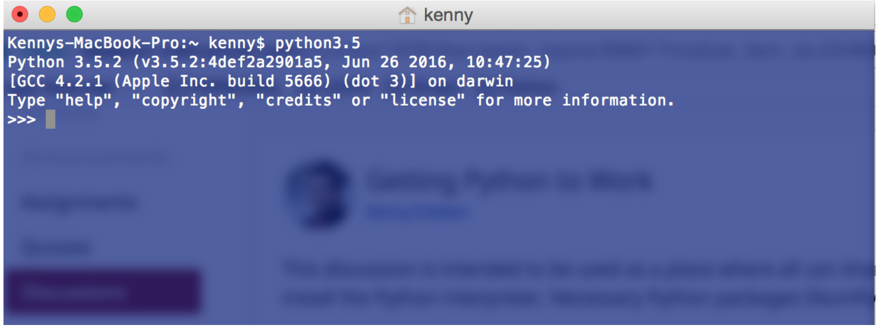
New to Spyder? Read our tutorial

```
Python 3.5.1 (default, Mar  2 2016, 03:41:10)
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]:
```

Permissions: RW End-of-lines: LF Encoding: UTF-8-GUESSED Line: 27 Column: 1 Memory: 63 %

Figure A.3: Screenshot of the Spyder user interface.

It should look roughly like the screen shoot below

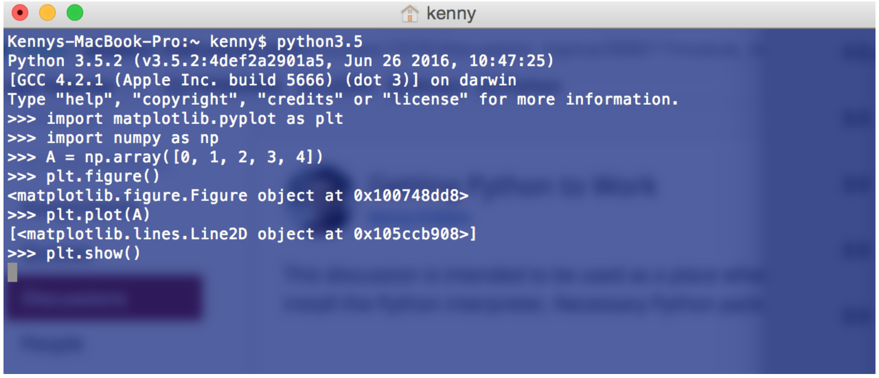


```
Kennys-MacBook-Pro:~ kenny$ python3.5
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

This shows we have the right version of Python (We need 3.X for this course). To verify if the packages work we just enter some very simple code that is supposed to a straight line plot. If we get such a plot displayed then we know the packages are installed properly. Here is the code you have to write to verify if you have all you need

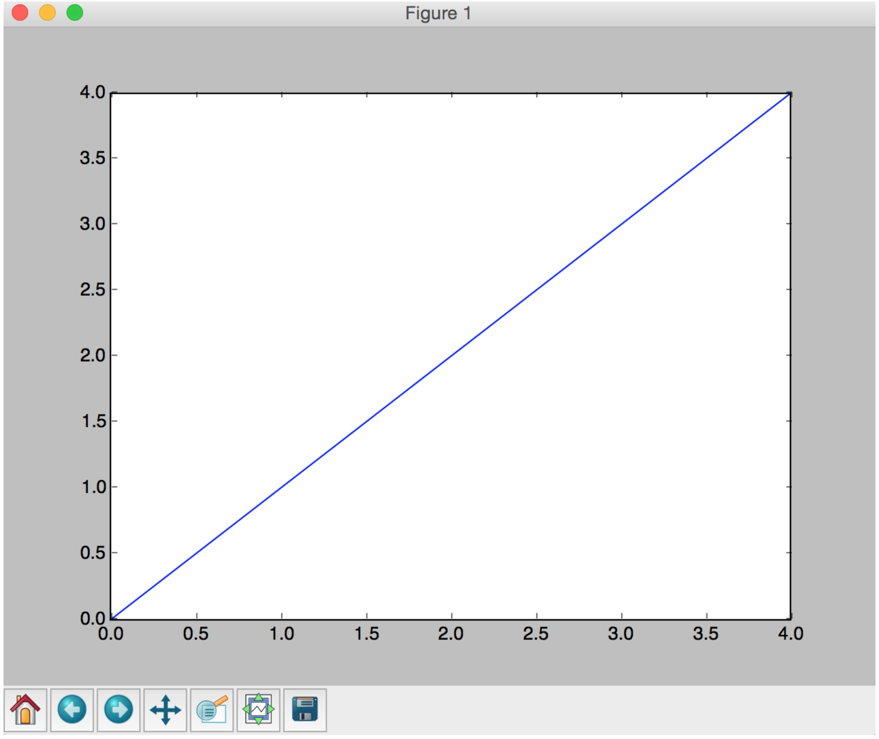
```
import matplotlib.pyplot as plt
import numpy as np

A = np.array([0, 1, 2, 3, 4])
plt.figure()
plt.plot(A)
plt.show()
```

When you enter this code into the Python shell running in the terminal then it will look like the screenshot shown below.



```
Kennys-MacBook-Pro:~ kenny$ python3.5
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> A = np.array([0, 1, 2, 3, 4])
>>> plt.figure()
<matplotlib.figure.Figure object at 0x100748dd8>
>>> plt.plot(A)
[<matplotlib.lines.Line2D object at 0x105ccb908>]
>>> plt.show()
```

After writing the last line of code a plot similar to the one shown below should pop up on your screen



At this point we know that the Python installation works as we need it to. We may now proceed to install the PyCharm Edu software. We will just refer to this software as “PyCharm” in the following for brevity. PyCharm is what is called an IDE. These three letters stands for “Integrated Development Environment”. An IDE is the equivalent of Microsoft Word, but specific for programing. There exist many different IDEs that support python programming. For this course we have chosen to use PyCharm. If you decide to use something other then you are welcome to do so, but you have to battle with learning to use the IDE on your own.

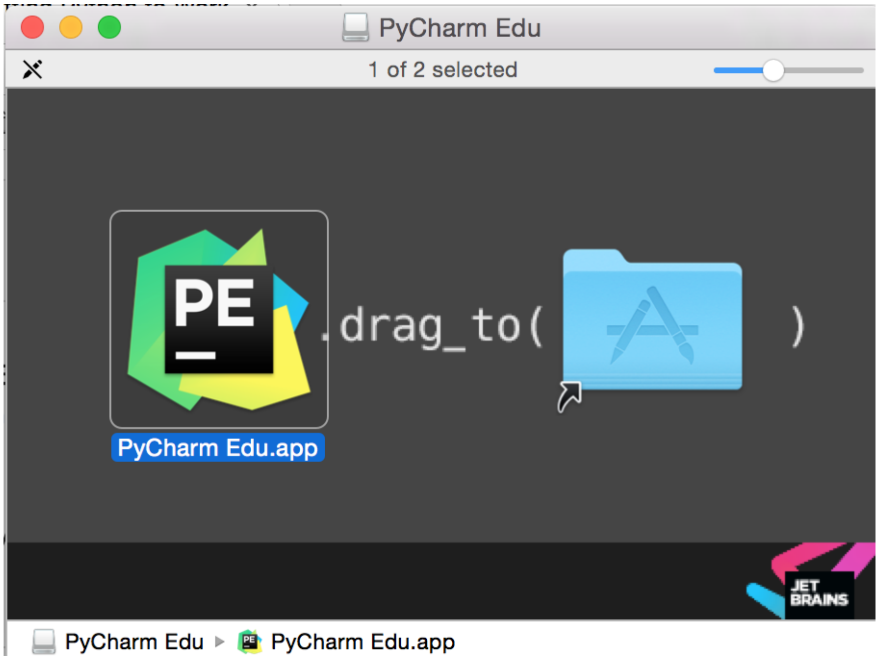
First go to the download page

`https://www.jetbrains.com/pycharm-edu/download/`

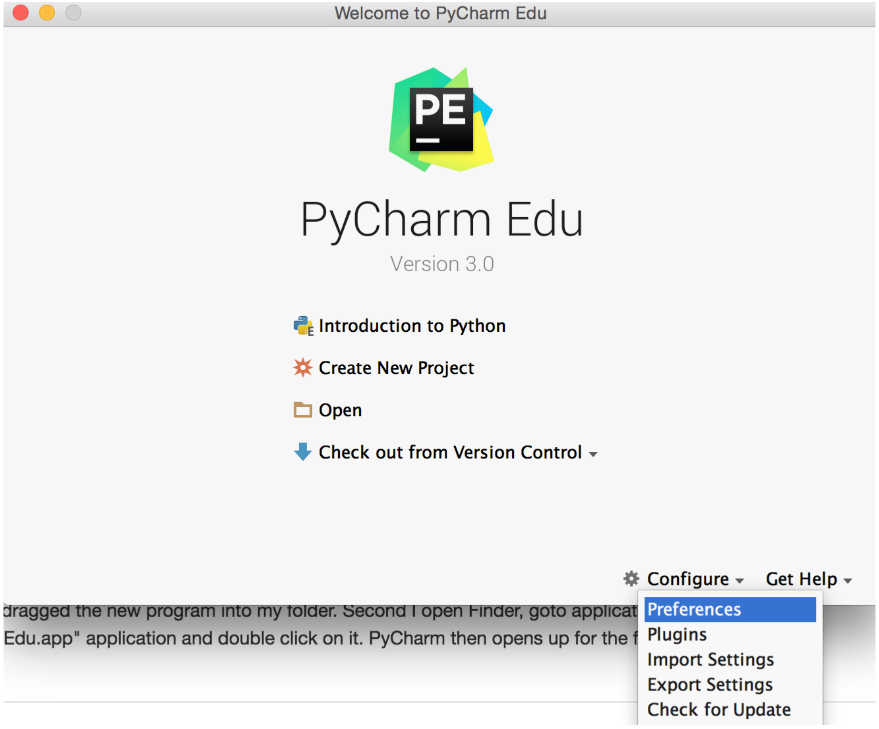
Select the version of PyCharm that matches your operating system. The click the download button. The downloaded file will on OS X be located in your download folder

```
/Users/kenny/Downloads/pyharm-edu-3.0.dmg
```

Now you should just open up the downloads folder and double clicked on the dmg-file and do the usual drag into the Applications folder to install the software. The popup should look a similar to the one shown below.



We are not quite done with installing PyCharm. We only put the software onto the computer, we also need to set it up so it works more to our own liking. So now we open Finder, locate the applications folder and find the “PyCharm Edu.app” application. Having found the application in Finder we now double click on the application and PyCharm Edu will opened up for the first time. You will see a window pop up similar to the one shown below.

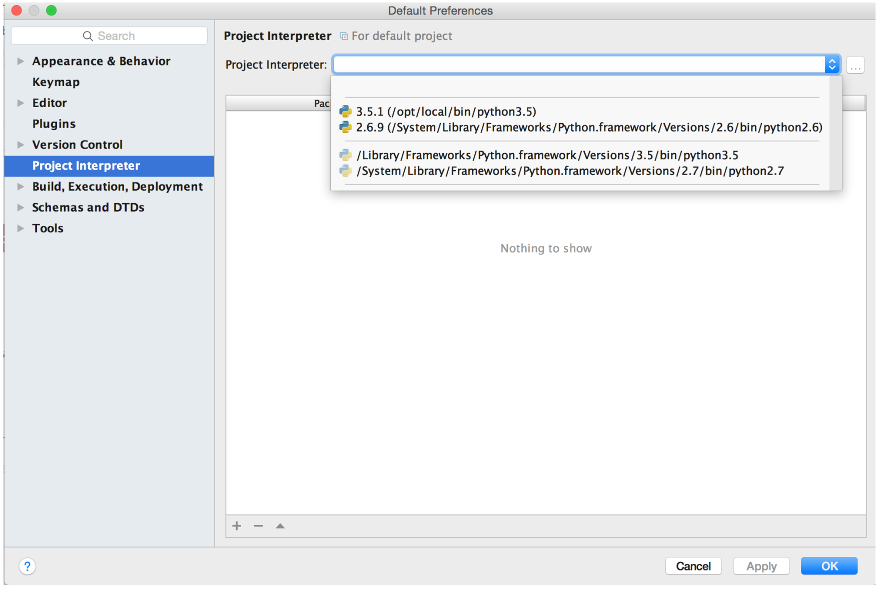


dragged the new program into my folder. Second I open Finder, goto applicat

Edu.app" application and double click on it. PyCharm then opens up for the f

To work more fluent with PyCharm we will go into the settings and specify that we wish to use the Python interpreter we just installed with MacPorts. This is how we do it. When PyCharm EDU shows its welcome screen, we click on the configure drop-down box at the right-bottom of the screen and select the preferences menu item. This is shown in the screenshot above.

A new window pops up, where we in the list-view select the item with the name "Project Interpreter". We now click on the drop-down box to see what interpreters that PyCharm have found and will let us select from. This is shown in the screenshot below.

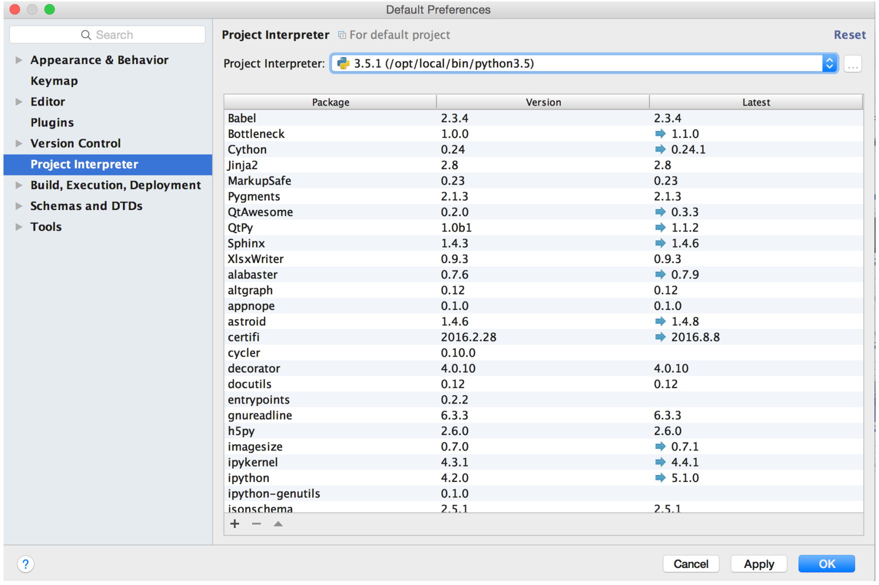


Project Interpreter — For default project

Project Interpreter:

- 3.5.1 (/opt/local/bin/python3.5)
- 2.6.9 (/System/Library/Frameworks/Python.framework/Versions/2.6/bin/python2.6)
- /Library/Frameworks/Python.framework/Versions/3.5/bin/python3.5
- /System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7

Fortunately in our case PyCharm did find the interpreter we installed with MacPorts. So we simply select this version as our default interpreter in PyCharm. This is shown in the screenshot below.



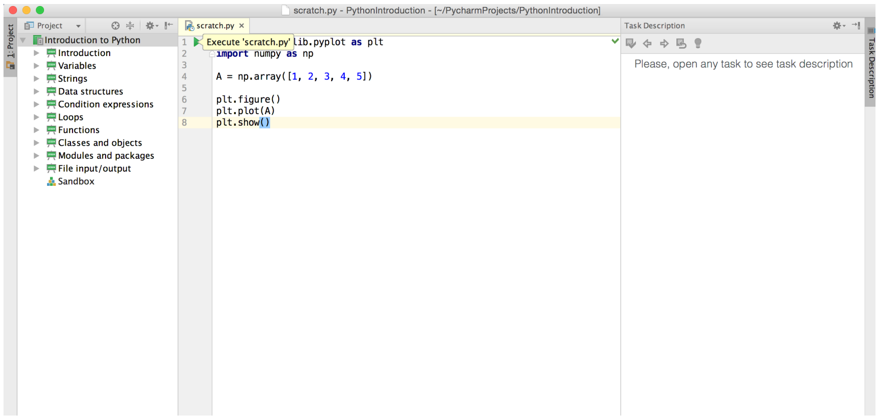
Project Interpreter — For default project

Project Interpreter: 3.5.1 (/opt/local/bin/python3.5)

Package	Version	Latest
Babel	2.3.4	2.3.4
Bottleneck	1.0.0	➡ 1.1.0
Cython	0.24	➡ 0.24.1
Jinja2	2.8	2.8
MarkupSafe	0.23	0.23
Pygments	2.1.3	2.1.3
QtAwesome	0.2.0	➡ 0.3.3
QtPy	1.0b1	➡ 1.1.2
Sphinx	1.4.3	➡ 1.4.6
XlsxWriter	0.9.3	0.9.3
alabaster	0.7.6	➡ 0.7.9
altgraph	0.12	0.12
appnope	0.1.0	0.1.0
astroid	1.4.6	➡ 1.4.8
certifi	2016.2.28	➡ 2016.8.8
cycler	0.10.0	
decorator	4.0.10	4.0.10
docutils	0.12	0.12
entrypoints	0.2.2	
gnureadline	6.3.3	6.3.3
h5py	2.6.0	2.6.0
imagesize	0.7.0	➡ 0.7.1
ipykernel	4.3.1	➡ 4.4.1
ipython	4.2.0	➡ 5.1.0
ipython-genutils	0.1.0	
jsonschema	2.5.1	2.5.1

Your system might be a little different from ours, so it could happen you may need to help PyCharm locate your Python interpreter if it is now already on the list.

If one wish to verify that all are in order then open up PyCharm Edu and select the “introduction to python” option on the welcome screen (you can also choose to create a new project, it does not matter that much). Now make a new scratch python file and write the same code as was used in the Python shell before. Once the code have been written press the little green triangle as shown in the screenshot below.



scratch.py - PythonIntroduction - [~/PycharmProjects/PythonIntroduction]

```
lib.pyplot as plt
import numpy as np

A = np.array([1, 2, 3, 4, 5])

plt.figure()
plt.plot(A)
plt.show()
```

Please, open any task to see task description

A.3 Installing on Windows 7 Enterprise

Here is a short description of how we successfully installed Python 3.5 and PyCharm Edu 3.0 on our windows PC.

First we install Anaconda. Anaconda is a platform that contains Python (several versions) as well as over 100 packages among them numpy and matplotlib, which are the ones we need for the NumIntro course. Go to

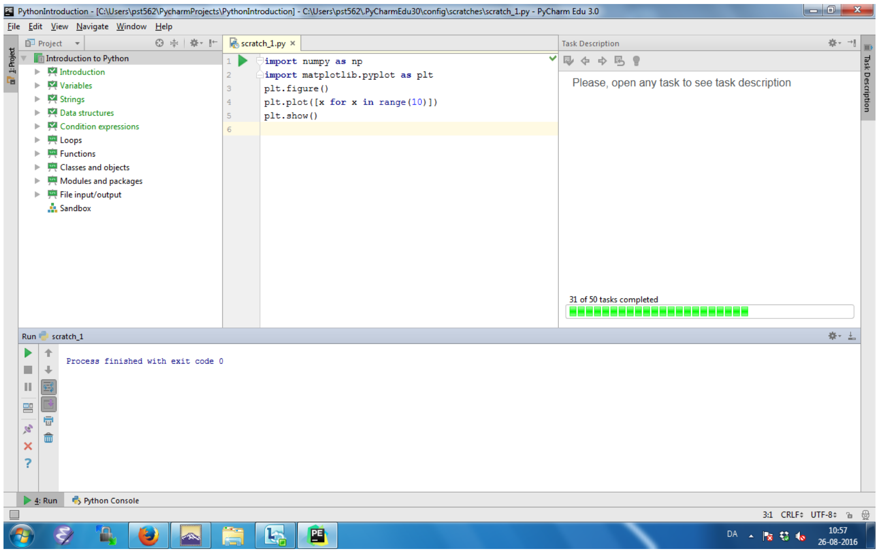
`https://www.continuum.io/downloads`

Download and install Anaconda. After installation you do not have to open the Anaconda navigator. Install PyCharm Edu from

`https://www.jetbrains.com/pycharm-edu/download/`

When asked choose to use the latest version of Python: Python 3.x Hopefully you should see the Anaconda version, which you should select. You may have only this version. Wait for PyCharm to load all the packages. Then press OK.

Back to the welcome screen you can now go to Introduction to Python and start going through the introduction doing the tasks. It may be an idea to first open a scratch file to check that the installation works. Notice that at this point I had done 31 on the 50 tasks.



```
import numpy as np
import matplotlib.pyplot as plt
plt.figure()
plt.plot([x for x in range(10)])
plt.show()
```

Please, open any task to see task description

31 of 50 tasks completed

```
Process finished with exit code 0
```

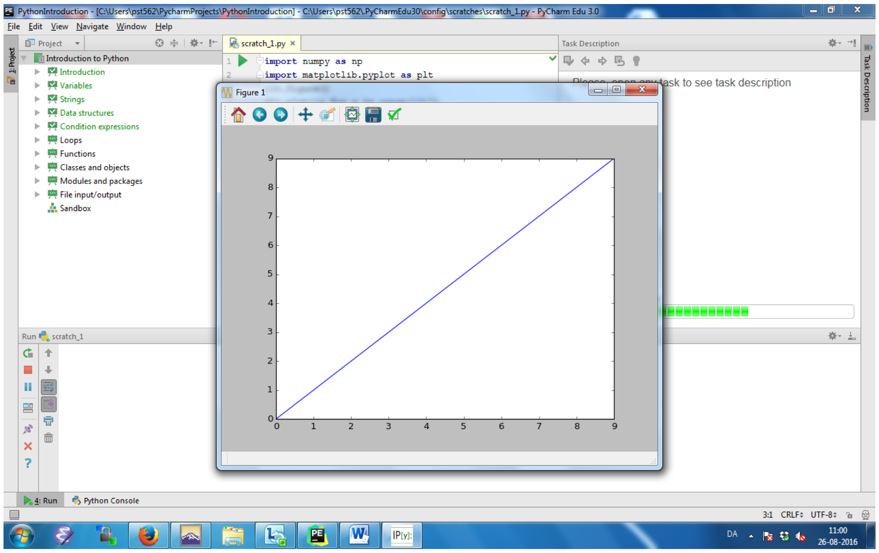
To check the installation type what you see in the scratch window

```
import numpy as np
import matplotlib.pyplot as plt

plt.figure()
plt.plot([x for x in range(30)])
plt.show()
```

Wait for the green arrow to appear. Press the green arrow and hopefully the console window below should return

```
Process finished with exit code 0
```

Then a pop up window should appear with your first graph.



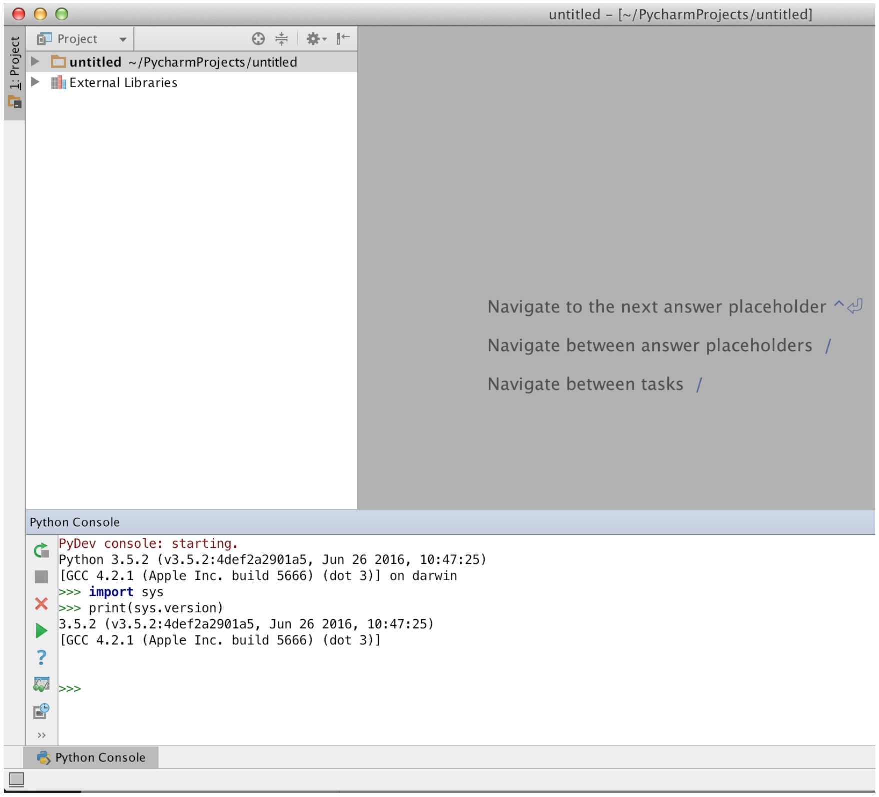
```
import numpy as np
import matplotlib.pyplot as plt
```

A.4 General Notes

Here is another way to check on any platform that the version number being used is correct (Python 3.X). From the Python shell (command-line: In PyCharm it is found at the bottom left of the screen and called “Python Console”), run the two commands (press <enter> or <return> after each)

```
import sys

print(sys.version)
```

Sample output (you can see that on my machine it actually already displays the version as soon as I open the console):


untitled – [~/PycharmProjects/untitled]
1: Project
Project
untitled ~/PycharmProjects/untitled
External Libraries
Navigate to the next answer placeholder ^⏎
Navigate between answer placeholders /
Navigate between tasks /
Python Console
Python Console

```
PyDev console: starting.
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
>>> import sys
>>> print(sys.version)
3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)]


>>>
```