Math 246 Unit 1: Introduction to Python

Brenton LeMesurier, Revised October 12, 2015

This is an introduction to some basic features of the Python software that we will be using in MATH 245 and 246. We will mostly use this through the interactive system Spyder, but with some iPython notebooks too.

IPython notebooks can show the output given by various Python commands, but I encourage you to also attempt each exercise at your computer.

Note that we use the newer version, Python 3.4; that is important if you are downloading software to use on your own computer.

Getting Python software for scientific computing

In the computer classroom MYBK 200, Python is already available in several forms. I recommend using Spyder (http://pythonhosted.org/spyder/), either as a stand-alone app or from within the bundle *Anaconda*, which includes Spyder along with *IPython* and the IPython notebook system, now called *Jupyter*.

Spyder includes a collection of useful resources beyond basic Python: extra tools for numerical and scientific computing (Numpy and Scipy), for producing graphs (Matplotlib), and for editing files and debugging programs. You can get Spyder installers from https://github.com/spyder-ide/spyder/releases (https://github.com/spyder-ide/spyder/releases) If you want more information about downloading and installing, see https://pythonhosted.org/spyder/installation.html (https://pythonhosted.org/spyder/installation.html)

You can download Anaconda from http://continuum.io/downloads#py34 (http://continuum.io/downloads#py34) Make sure that you get the version with Python 3.4, not Python 2.7; if necessary, click on "I want Python 3.4".

One advantage of the stand-alone version of Spyder (as opposed to the version within Anaconda) is that you can specify it as the app in which to open Python files when you double-click on them. With the Anaconda version, you must instead open such files from within Spyder.

Online resources

There is a vast array of free online tutorials and documentation for Python and the other tools that we are using, such as

- http://www.learnpython.org (http://www.learnpython.org)
- https://docs.python.org/2/tutorial/ (https://docs.python.org/2/tutorial/)

and even complete free online courses like

- http://www.codecademy.com/tracks/python (http://www.codecademy.com/tracks/python)
- http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-189-a-gentle-introduction-to-programming-using-python-january-iap-2011/index.htm (http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-189-a-gentle-introduction-to-programming-using-python-january-iap-2011/index.htm)

These might be useful sometimes, but our direction is a bit different: aiming at what is most important to a mathematics student, rather than a computer science or engineering student.

Using Python as a calculator

Arithmetic

This is mostly easy stuff, but with a few quirks of notation and such to note.

First, let's see how to run Python code with the following famous formula. To run the code in a "code" box, select the box with the mouse (single click) and then click on the "play" button above.

In [ ]:

```
2 + 2
```

You should also start Spyder and enter the same expression into its command window.

No prize for predicting the above result, but division of two integers requires care:

In [ ]:

```
11 / 4
```

Here Python3 gives a real number answer, whereas some programing languages always given an integer answer for the division of integers, by rounding down ("truncating") if necessary.

When you want the truncated, integer answer, use a double slash:

In [ ]:

```
11 // 4
```

and to get the remainder, use the percent sign:

In [ ]:

```
11 % 4
```

With multiplication, the only catch is that *there is no multplication sign on the keyboard*, so the asterisk is used instead:

In [ ]:

```
7 * 4
```

There is no way to type superscripts either, so exponentiation uses a double-asterisk (not the "caret" or up-arrow as in some languages):

In [ ]:

```
2**5
```

Numbers with exponents can be expressed using this exponential notation, but note how Python outputs them:

In [ ]:

```
5.3*10**21, 7*10**-8
```

This "E" notation is the standard way to describe numbers with exponents, and oyu can input them that way too.

When printing numbers, Python decides if the number is big enough or small enough to be worth printing with an exponent:

In [ ]:

```
2e-10, 3e-5, 3e-4, 5e+4, 6e15, 6e16
```

Mathematical functions and constants: module math

Python includes many **modules** and **packages** that add useful defintions of functions, constants and such, and for us the most important is `math` which is a standard part of Python.

(Later will see some other modules that are not part of standaerd Pythom, but which Spyder adds to the basic collection: `numpy`, `matplotlib`, `pylab`, and `scipy`)

The quickest way to get all the basic mathematical stuff is with

In [ ]:

```
from math import *
```

which makes everything in module `math` available. For example, it gives our two favorite irrational numbers

In [ ]:

```
pi, e
```

and a lot of familiar functions, like

In [ ]:

```
sin(pi/4), cos(pi), cosh(0)
```

Note that **all** functions need parentheses around their arguments; no lazy shortcuts with trig. functions and such.

The asterisk in the above command *from math import* * means "use everything in this module". Sometimes it is better to specify just the items that you will use: we could have got the five items used above with

In [ ]:

```
from math import pi, e, sin, cos, cosh
```

Logarithmic functions

There is also a function log(), but which base does it use?

In [ ]:

```
log(10), log(e)
```

Running the above cell reveals that "log()" is the *natural logarithm*, base e; what mathematicians usually call "$\ln$".

It is time to explore: try to find the base 10 logarithm. (Use Spyder to test your ideas.) A big hint: look at the following, which is yet another logarithm; one of particular interest in computer science.

In [ ]:

```
log2(64)
```

Powers and roots

A square root can of course be computed using the 1/2 power, as with

In [ ]:

```
16**0.5
```

but this function is important enough to have a named form:

In [ ]:

```
sqrt(2)
```

Exercise: here is a *wrong* way to use the exponential notation in an attempt to compute a square root: explain what goes wrong!

In [ ]:

```
16**1/2
```

Explain what went wrong

Type your explanation into the following empty "markdown" text cell, and save your modified version of this file to submit. This gives you something to practice with submitting work through OAKS, which is how most work will be done in this course.

More about input to IPython notebooks: devising and evaluating your own formula

It is time to practice with basic Python commands, and using them in notebooks.

- Start by reproducing some of the above calculations in Spyder.
- Next, with the above warnings and examples in mind, evaluate $\frac{1+\sqrt{2}}{2\pi}$. (The correct answer is 0.384...)
- Then explore the command history and editing features, and use these to evaluate $\frac{1-\sqrt{2}}{2\pi}$ by modifying the previous formula, *without* typing the entire new expression from scratch.

Do this first in Spyder, typing the Python commands into the input window ("iPython console").

Then to conclude, put your work into a copy of this notebook: add cells below here, put your commands into them, run the cells, and save the resulting work -- and submit the file through OAKS.