A Short and Incomplete Introduction to Python

Part 0: Introduction

Sigve Haug <sigve.haug@math.unibe.ch>,
Alexander Kashev <alexander.kashev@math.unibe.ch>
Science IT Support (ScITS), University of Bern

Based on a course by Riccardo Murri / Sergio Maffioletti
S3IT: Services and Support for Science IT, UZH

August 20–21, 2018

Welcome!

Prerequisites

Normally, a course like this assumes a basic experience with computer programming.

Any language should do, as long as you are already familiar with the concepts of variables and functions.

However, for the benefit of non-programmers, we will provide a short intro to basic concepts.

If this applies to you, this is a helpful place to start:
https://wiki.python.org/moin/BeginnersGuide/NonProgrammers

Why program at all?

"The good news about computers is that they do what you tell them to do. The bad news is that they do what you tell them to do."

Computers are a wonderful invention, allowing to automate and accelerate processing of data. Like a Swiss army knife, it can be used for a great number of tasks.

However, for each particular workflow computers need to be instructed precisely on what to do. It's the job of the **software**, and software needs to be written.

Why should I program? I

Automate boring, repetitive tasks. Computers excel at mass-processing of data. Automating tasks can save time and reduce the number of errors.

Some tasks are so massive in scope they cannot be solved without computers.

Solve my specific tasks. Even if existing software can be used in solving parts of your task, perhaps no single one does it exactly the way you want from start to finish.

If you can tie other programs together, modify them, or write your own — you can achieve better results.

Why should I program? II

Understanding limitations. Non-programmers often have little idea of what is possible to do with computers, and how hard or easy it is.

Learning programming helps you specify your requirements better, even if someone else will do the bulk of the work.

Fun. Programming is a creative activity. If you get into it, it can be genuinely enjoyable to do.

And if your program helps others, it's an added bonus.

Why Python?

Relative simplicity. Python isn't hard to install on many different systems. Code in Python can be simpler to express, which is great for learning.

High level language. Programming with Python does not require intimate knowledge of how computers work, at least to start with.

Popularity. Python is a very popular language. In practice, this means that a lot of ready recipes and tools exist for many tasks (especially scientific ones), and a lot of advice is available online.

Python 2 *vs* Python 3

There are currently two major versions of Python available, with slightly different syntax and features.

Python 2.7 is the last release in the 2.x series.

Python 3.x has a more polished syntax, removing inconsistencies and some historical baggage.

In this course we will use **Py3 syntax**.

Watch a debate between "Pro" and "Contra" advocates:
http://www.physik.uzh.ch/~nchiapol/webm/3_1_Python3.webm

Explore the key differences:
http://tinyurl.com/py2-and-py3-key-differences

Next steps

The course will be structured as a mixture of slides and hands-on sessions for practicing Python programming.

So, the very first step is making sure you can access the Jupyter/IPython server for running the exercise notebooks.

How to run Python code

The Python shell, I

Python is an *interpreted* language.

This means that Python code is ready to be immediately run, including line by line, without an initial compilation of the whole program.

It also features an interactive "shell" for evaluating expressions and statements immediately.

If you have Python installed, you can use `python` to invoke a basic interactive shell.

```
$ python
Python 3.6.5 |Anaconda, Inc.| (default, Apr 29 2018, 16:14:56)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>  ← here is where you enter commands
```

The Python shell, II

The interactive mode of Python is very bare-bones. There are better options for interactive shells with Python.

One of them is IPython, which is installed by default with Anaconda and can be installed as a module for other Pythons installations.

The IPython shell is started by invoking the command `ipython` in a terminal window.

```
$ ipython
Python 3.6.5 |Anaconda, Inc.| (default, Apr 29 2018, 16:14:56)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 - An enhanced Interactive Python. Type '?' for help.

In [1]: ← here is where you enter commands
```

Python scripts

Interactive mode is good for experimentation, but eventually you want to save your code for future reuse.

In that case, you use a text (or code) editor to write your code in a file, and tell Python to run that code.

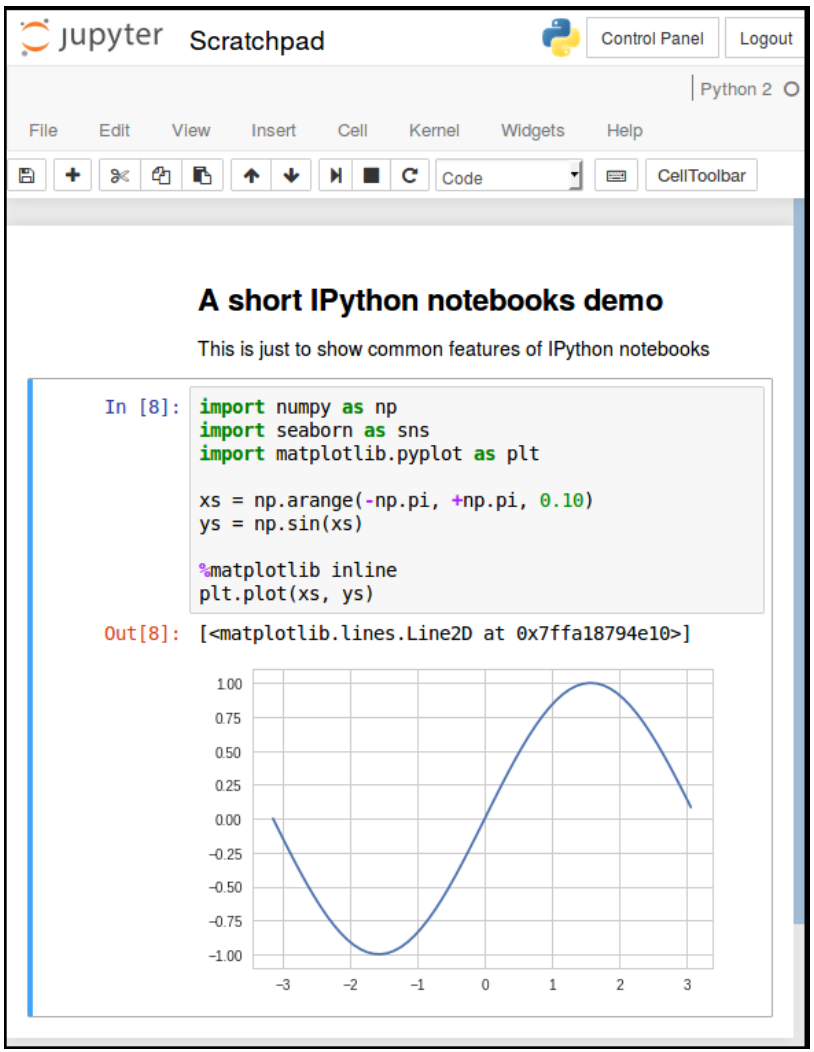
As an obligatory first example, **hello.py:**

```
print("Hello world!")
```

And running it:

```
$ python hello.py
Hello world!
```

The IPython notebook, I



A more appealing way of interacting with Python is through the IPython (Jupyter) notebooks.

Notebooks are made of “cells”, which come in two flavors:

- documentation cells, containing text formatted according to the Markdown conventions;
- code cells, containing arbitrary Python code

The IPython notebook, II

Notebooks are somewhere in-between interactive shell and written and saved script.

To run Python code in the notebook:

- Type your code in a cell besides the `In [ ]:` (multiple lines are allowed)
- Press **Ctrl+Enter** to evaluate the cell (prompt changes to `In [*]:`) — or press **Alt+Enter** to evaluate the code *and* open a new code cell.
- When the Python kernel has done computing, the result appears *under* the code cell marked with a `Out [ ]:` label.

The Python shell, III

Expressions can be entered at the Python shell prompt; they are evaluated and the result is printed:

```
In [1]: 2+2
Out[1]: 4
```

Note that the classic Python shell uses '>>>' as a prompt; expression evaluation works exactly the same, though:

```
>>> 2+2
4
```

Throughout these slides, all Python code marked with either 'In [*]' or '>>>' can also be entered and evaluated in the IPython notebook cells.