

Introduction to (Scientific) Programming with Python

Jelka Stojanov, Timo Flesch

MT 2020

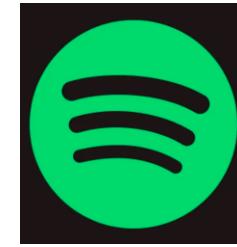
University of Oxford

Outline

- Why Python?
- Practical
 - Part 1: Intro to Python
 - Part 2: Python for scientific programming
 - Part 3: Python for statistical data analysis
- Additional Resources

Why Python?

**It's a general purpose programming language,
used to power many of the apps you use every day!**



Why Python?

There are free (!!) add-ons for everything:

Create your own experiments



Make beautiful figures



Organise and analyse numerical data



Perform sophisticated statistical analyses



Hop on the machine learning hype train!



Why Python?

It's just better than its competitors!



>



Matlab is very expensive! Python is free. When using the right packages/add-ons, Python has the same functionality and syntax as Matlab.



>



Python is much easier to learn, as the syntax is less complex. At least for science, there is a better community/ecosystem. Some Python packages run C(++) code in the background, which makes them almost as fast as their C++ counterparts.



>



Python is a general purpose language. It can be used for almost everything. R was developed for statistical analyses.



>



Again, much simpler syntax. You can achieve the same results with fewer lines of code

Python Practical

1. Go to <http://www.github.com/timoflesch/intro2python>

The screenshot shows the GitHub repository page for 'intro2python'. The repository has 54 commits, 3 branches, and 0 tags. The README.md file contains the following text:

A brief (2h) introduction to programming in Python for absolute beginners, 2020 Edition

v1.0 (2019): The course was initially developed and delivered by Timo Flesch and Mirta Stantic.
v2.0 (2020): The course was refined and delivered by Timo Flesch and Jelka Stojanov.

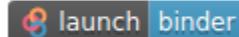
The repository page also includes sections for About, Releases, Packages, Contributors, and Languages.

Python Practical

2. Click on “launch binder”

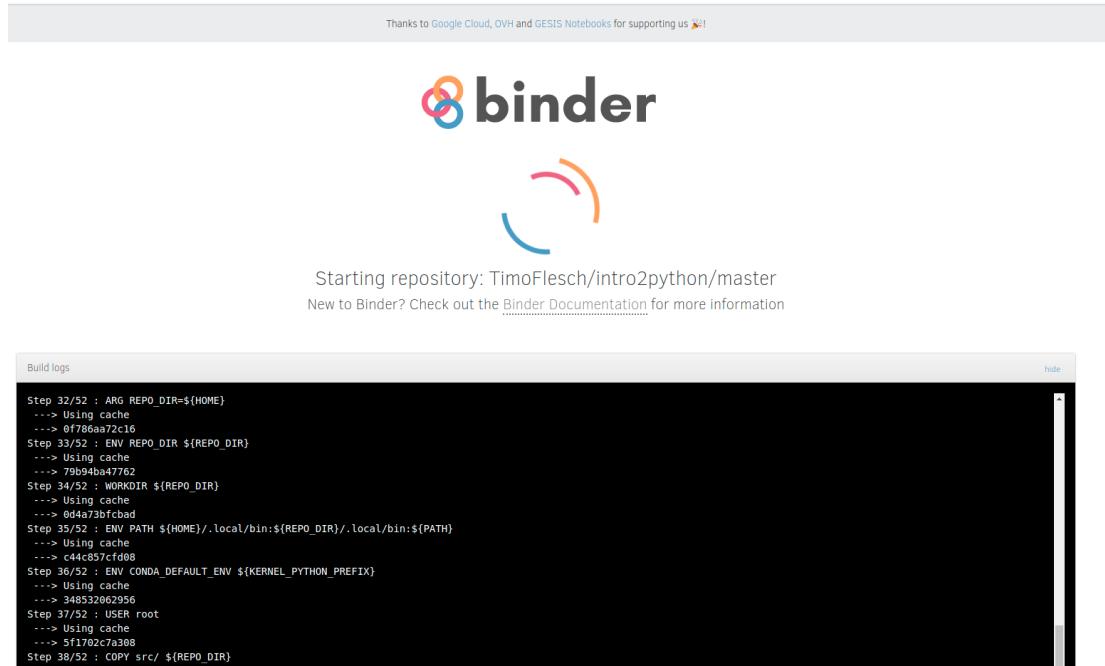
How to get started

Click on the "launch in binder" button below. This will open a new website



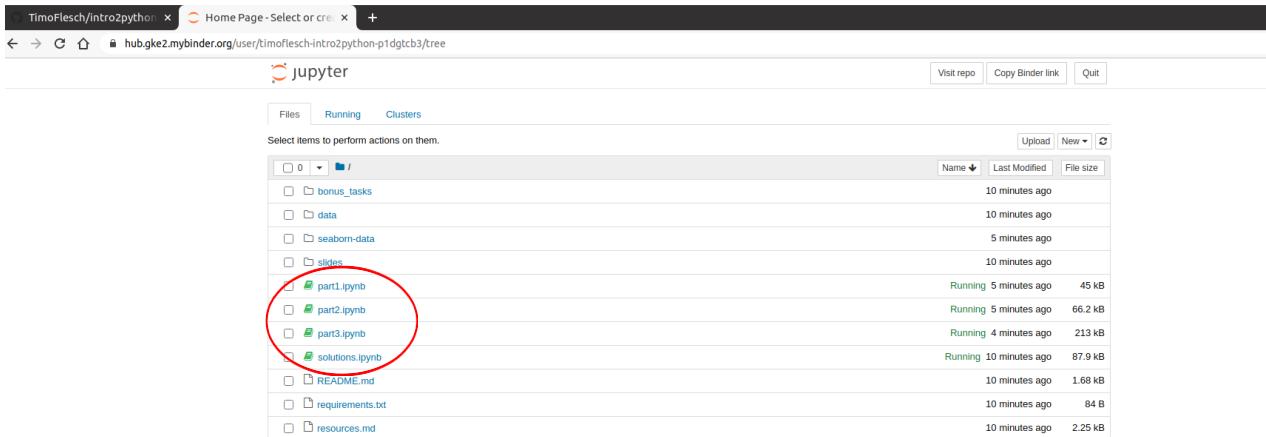
Python Practical

3. Stare for a while at this screen



Python Practical

4. Now this screen should show up



From here you can open the worksheets that we have provided for you. This will run an interactive Python session in your browser, where you can write and execute your own Python programs.

Python Practical

5. You're now running an interactive worksheet with Python code in your web-browser. **Have fun** 😊

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter part1 Last Checkpoint: 7 minutes ago (autosaved)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, Memory: 635 / 2048 MB
- Section Header:** PART 1: PYTHON - A (VERY) SHORT INTRODUCTION
- Text:** Today you'll learn the basics of programming in Python. We are currently using a so-called **Python notebook**. The notebook consists of two types of cells, those that contain text (like the one you're reading right now) and those that contain Python code.
- Text:** Below is an example of a cell with Python code. When we click on a cell with code and either press [SHIFT] + [ENTER] or click on the [RUN] button above, we tell the computer to interpret the commands which means that it turns the Python code into actions we want it to execute.
- Code Cell (In [1]):**

```
# This is a comment. Everything preceded by a hash symbol (#) is ignored by the computer.  
# This allows you to add comments/notes/documentation to your Python programme.  
print("Hello world")
```

Output: Hello world

Text: As you can see, we instructed the computer to "print" a string of letters.

Text: Don't worry too much about the syntax right now, we'll go through this step by step.
- Section Header:** 1.1 VARIABLES
- Text:** Variables are used to store information that can later be referenced and manipulated by your computer program. It's helpful to think of variables as containers for information. For example, imagine you build a drawer and put socks in it. In this case, the drawer would be a variable and the socks we put in would correspond to its value(s). When you reference the variable, i.e. open the drawer, you'll find socks in there.
- Code Cell (In [2]):**

```
# Let's create a drawer with socks  
drawer = "socks"  
  
# Let's have a look what's inside the drawer  
drawer
```

Outlook

Learn to code!

1. A complete course <https://www.learnpython.org>
2. ditto <https://www.w3schools.com/python/>
3. Advanced Course <https://automatetheboringstuff.com>

How to install Python

1. Just Python <https://www.codecademy.com/articles/install-python>
2. Jupyter Notebook (that you have used today) <https://jupyter.org/install>
3. Anaconda (a collection of useful packages and other software for data scientists)
<https://www.anaconda.com/distribution/>

Outlook

Text editors

1. atom editor <https://atom.io/>
2. vscode <https://code.visualstudio.com/>
3. Sublime <https://www.sublimetext.com/>

All in One Solutions (Similar to the Matlab interface or R-Studio)

1. Spyder (free) <https://www.spyder-ide.org/>
2. Pycharm (free basic and commercial pro version) <https://www.jetbrains.com/pycharm/>

Outlook

Coding Challenges

1. Hackerrank <https://www.hackerrank.com/>
2. Leetcode <https://leetcode.com>

Python for Psychologists

<https://www.marsja.se/best-python-libraries-psychology/>