

Introduction to (Scientific) Programming with Python

Jelka Stojanov, Timo Flesch

MT 2021

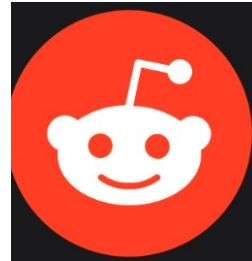
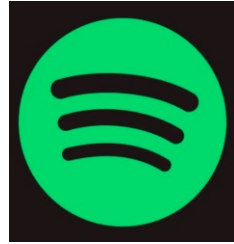
University of Oxford

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!



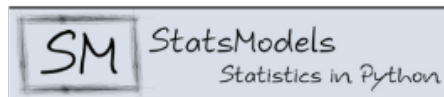
Organise and analyse numerical data!



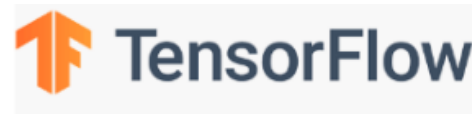
Make beautiful figures!



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>

A brief (2h) introduction to programming in Python for absolute beginners, 2021 Edition

v1.0 (2019): The course was initially developed and delivered by Timo Flesch and Mirta Stantić.

v2.0 (2020): The course was refined and delivered by Timo Flesch and Jelka Stojanov.

v3.0 (2020): The course was refined and delivered by Timo Flesch and Jelka Stojanov.

A brief intro to Python, a powerful general-purpose programming language, designed for second-year undergraduate students in Experimental Psychology and Biomedical Sciences at the University of Oxford.

In this course, you will learn:

- how to assign values to variables and perform different mathematical operations;
- how to manipulate data formats and data sets;
- how to use loops, functions and methods (you will not only use existing functions in Python, but also create your own).

We will also introduce you to some of the most important libraries in the scientist's toolkit.

Finally, you will apply the knowledge you have gained to run a few simple statistical analyses (repeated-measures ANOVA, linear regression) and to visualize the results.

This course does **NOT** assume any previous knowledge of Python or any other programming language. Resources for those who are interested in learning more will be provided in the interactive worksheet in class.

How to get started?

1. Open "slides.pdf"
2. Click on the "launch in binder" button below when you have read through the slides.



Python Practical

2. Click on the “launch binder” button:

How to get started?



1. Open "slides.pdf"
2. Click on the "launch in binder" button below when you have read through the slides.



Python Practical

3. Stare at this screen for a while:

Thanks to Google Cloud, OvH and GESIS Notebooks for supporting us 🙏!



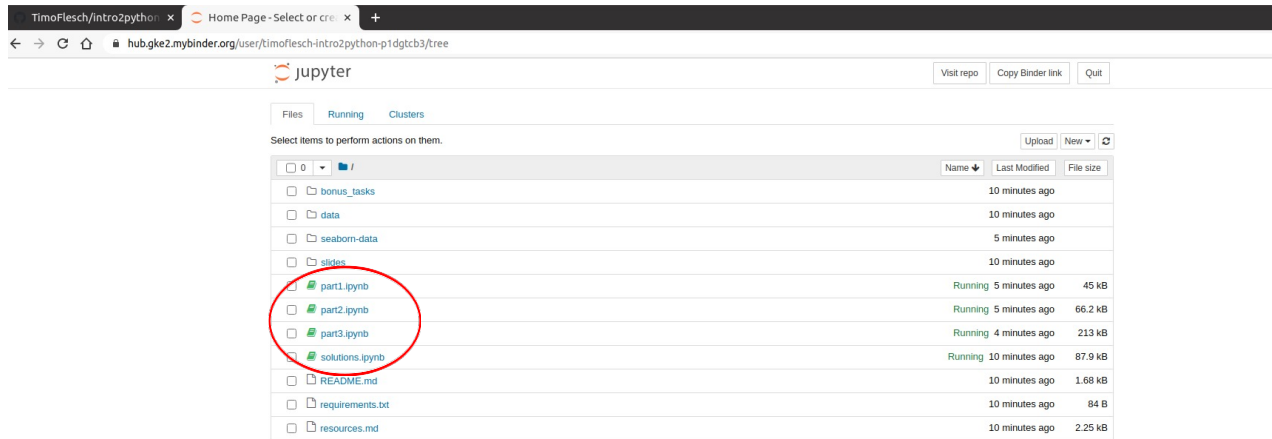
Starting repository: TimoFlesch/intro2python/master
New to Binder? Check out the [Binder Documentation](#) for more information

Build logs hide

```
Step 32/52 : ARG REPO_DIR=${HOME}
--> Using cache
--> 0f786aa72c16
Step 33/52 : ENV REPO_DIR ${REPO_DIR}
--> Using cache
--> 79b94ba47762
Step 34/52 : WORKDIR ${REPO_DIR}
--> Using cache
--> 0d4a73bfcba
Step 35/52 : ENV PATH ${HOME}/.local/bin:${REPO_DIR}/.local/bin:${PATH}
--> Using cache
--> c44c857cfd08
Step 36/52 : ENV CONDA_DEFAULT_ENV ${KERNEL_PYTHON_PREFIX}
--> Using cache
--> 340932067956
Step 37/52 : USER root
--> Using cache
--> 5f1702c7a308
Step 38/52 : COPY src/ ${REPO_DIR}
```


Python Practical

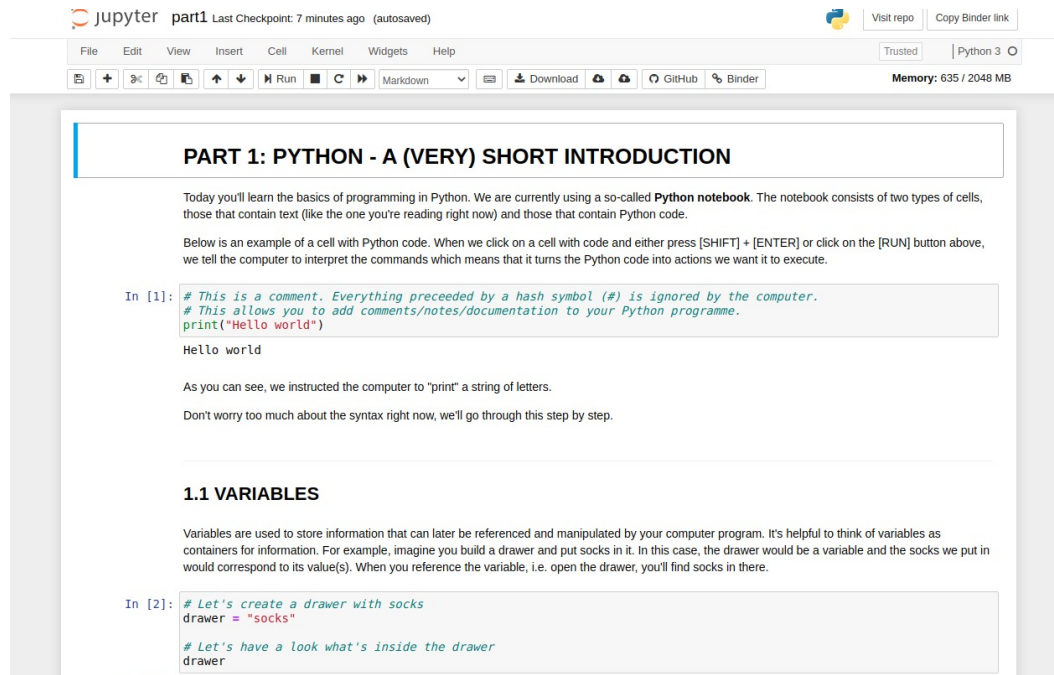
4. Now this screen (or similar) should show up:



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 are now running an interactive worksheet with Python code in your web-browser. **Have fun** 😊



The screenshot shows a Jupyter Notebook interface. At the top, the Jupyter logo is followed by 'part1' and 'Last Checkpoint: 7 minutes ago (autosaved)'. On the right, there are links for 'Visit repo' and 'Copy Binder link'. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. To the right of the menu bar are 'Trusted' and 'Python 3' buttons. Below the menu bar is a toolbar with icons for file operations, running, and saving. The main content area has a title 'PART 1: PYTHON - A (VERY) SHORT INTRODUCTION'. The text explains that the notebook consists of two types of cells: text and Python code. It then shows an example of a code cell with a comment and a print statement. The output of the code is 'Hello world'. Below this, it explains that the computer is instructed to 'print' a string of letters. The next section is '1.1 VARIABLES', which explains that variables are used to store information that can later be referenced and manipulated by a computer program. It then shows a code cell with a variable 'drawer' assigned the value 'socks'.

Jupyter part1 Last Checkpoint: 7 minutes ago (autosaved) Visit repo Copy Binder link

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Memory: 635 / 2048 MB

PART 1: PYTHON - A (VERY) SHORT INTRODUCTION

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.

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.

```
In [1]: # This is a comment. Everything preceeded by a hash symbol (#) is ignored by the computer.
# This allows you to add comments/notes/documentation to your Python programme.
print("Hello world")
```

Hello world

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

Don't worry too much about the syntax right now, we'll go through this step by step.

1.1 VARIABLES

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.

```
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.python.org/downloads/>
2. Jupyter Notebook (that you have used/will use 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 <https://atom.io/>
2. Visual Studio Code <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/>