

Python Programming

Introduction

Mihai Lefter



Outline

Introduction

About Python

Running Python Code

About the course

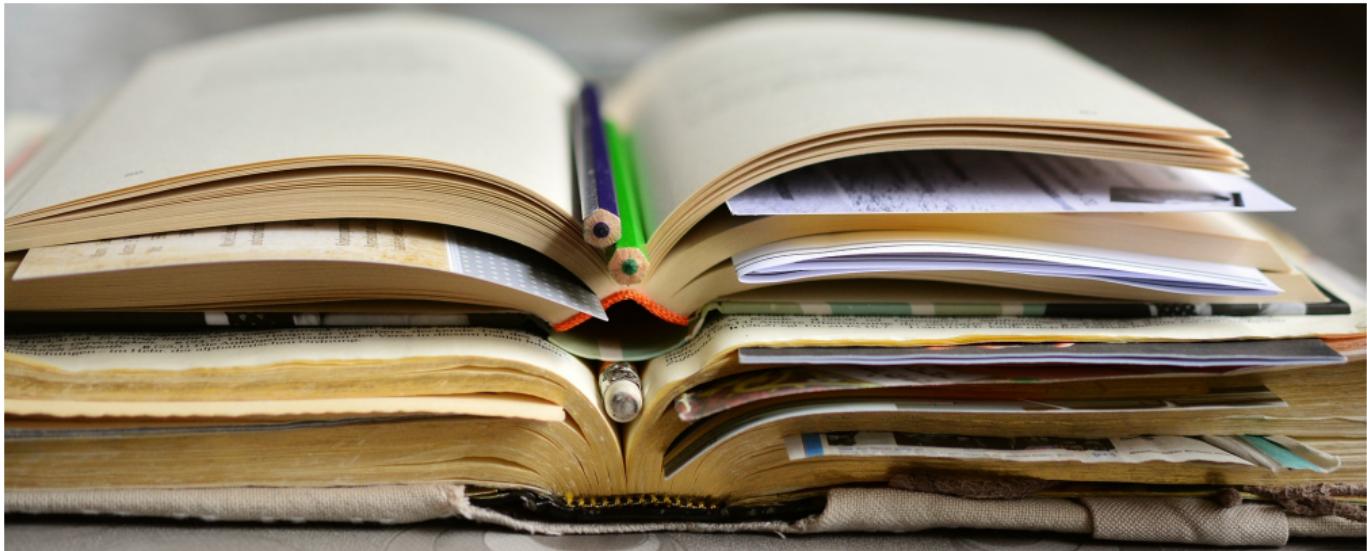
- Aimed at PhD students, Postdocs, researchers, analysts, ...
- Focus on:
 - Basic understanding of Python.
 - Programming as a tool to do your research.
 - Slightly biased on bioinformatics.



Introduction

Material

<https://github.com/LUMC/python-course>



Hands On!

Programming is fun!

- You only learn programming by doing it.
- Lecture format:
 - Blended teaching + exercising.
- Repeat the code from the slides/ lectures and play around with it.
- Do the session exercises.
- Practical sessions.



Introduction

Teachers

- Mark Santcroos
m.a.santcroos@lumc.nl
- Ruben Vorderman
r.h.p.vorderman@lumc.nl
- Redmar van den Berg
r.r.van_den_berg@lumc.nl
- Mihai Lefter
m.lefter@lumc.nl



Introduction

Program

Day 1 Monday 12/10		Day 2 Tuesday 13/10		Day 3 Wednesday 14/10		Day 4 Thursday 15/10	
9:00	30 min	Introduction	9:00	20 min	Sets	9:00	
9:30	40 min	Basics	9:30	20 min	Tuples	10:00	1 h
10:20	10 min	Break	10:20	10 min	Functions	10:30	30 min
11:00	50 min	Lists	11:00	30 min	Text files	11:30	Q&A
11:20	10 min	Break	11:20	30 min	Exceptions	12:00	1 h
11:40	50 min	Builtin Functions and Packages	11:40	10 min	Break	12:30	Practice
12:10	30 min	Lunch break	12:10	30 min	Q&A	12:00	30 min
12:40	20 min	Q&A	12:40	30 min	Lunch break	12:30	Assignments review
13:00	20 min	Flow Control	13:00	2.5 h	Practice	13:00	Lunch break
13:20	20 min	Dictionaries	13:20			13:30	30 min
13:40	1.5 h	Practice	13:40			14:00	Lunch break
15:10	50 min	Assignments review	15:10	60 min	Assignments review	15:10	30 min
16:00			16:00			15:30	Q&A
						16:00	30 min
							Wrap-up

Introduction

Software requirements

- Anaconda:
 - Python 3.x.
 - Comes with all that's required:
 - Python interpreter.
 - Jupyter Notebook.
 - Libraries: NumPy, Panda, matplotlib, Bokeh, Biopython, ...
 - [Installation instructions.](#)
- Git:
 - [Installation instructions.](#)



Introduction

Assignments

- We make use of GitHub Classroom.
 - GitHub account required.
 - Receive link with assignment repository.
- Own forked repository to work on:
 - Clone it.
 - Code it.
 - Push it.
- Direct file upload to repository is also possible.



Getting help

- Ask a teacher.



History

- Created early 90's by Guido van Rossem at CWI.
 - Name: Monty Python.
- Design is driven by code readability.



Centrum Wiskunde & Informatica



Features

- General purpose, high-level programming language.
- Interpreted, no separate compilation step needed.
- Imperative and object-oriented programming.
 - And some functional programming.
- Dynamic type system.
- Automatic memory management.

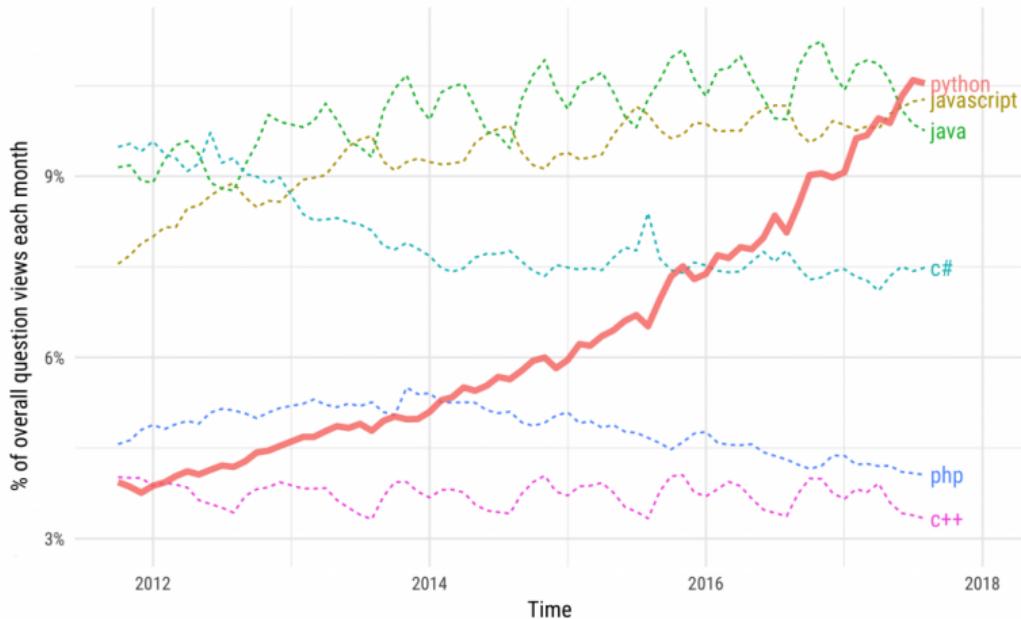
We'll come back to most of this.



About Python

Why Python?

- Widely used with a large community.



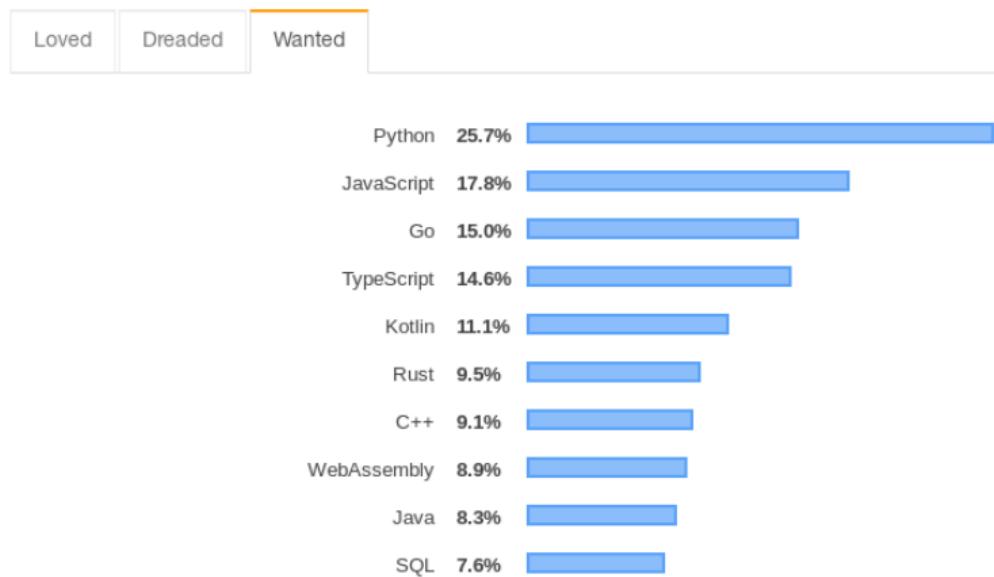
<https://stackoverflow.blog/2017/09/06/incredible-growth-python/>

About Python

Why Python?

- Widely used with a large community.

Most Loved, Dreaded, and Wanted Languages



<https://insights.stackoverflow.com/survey/2019/#most-loved-dreaded-and-wanted>

About Python

Why Python?

- Widely used with a large community.

Language Ranking: IEEE Spectrum				
Rank	Language	Type	Score	
1	Python	🌐💻🖱️	100.0	
2	Java	🌐📱💻🖱️	96.3	
3	C	📱💻🖱️	94.4	
4	C++	📱💻🖱️	87.5	
5	R	💻	81.5	
6	JavaScript	🌐	79.4	
7	C#	🌐📱💻🖱️	74.5	
8	Matlab	💻	70.6	
9	Swift	📱💻	69.1	
10	Go	🌐💻	68.0	

<https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019>

Why Python?

- Widely used with a large community.
- Rich scientific libraries.
- Many other libraries available.
- Readable and low barrier to entry.



Python 2 versus Python 3

- Python 2.7 is the last Python 2.
- Python 3 is backwards incompatible.
- Some libraries don't support it yet.
- Some Python 3 features are backported in Python 2.7.
- Last Python 2 release in April 2020.

We'll use Python 3 for this course.

Running Python Code

Interactively:

- Statement by statement, directly in the interpreter.

Running Python Code

Interactively:

- Statement by statement, directly in the interpreter.

```
terminal
$ python
Python 3.7 (default, Nov 26 2019, 10:23:46)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> print('Hello world')
Hello world
>>>
```

Running Python Code

Interactively:

- Statement by statement, directly in the interpreter.

```
terminal
$ python
Python 3.7 (default, Nov 26 2019, 10:23:46)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> print('Hello world')
Hello world
>>>
```

- Interpreters are great for prototyping.
- But not really suitable if you want to share or release code.

Running Python Code

Interactively:

- Statement by statement, directly in the interpreter.

```
terminal
$ python
Python 3.7 (default, Nov 26 2019, 10:23:46)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> print('Hello world')
Hello world
>>>
```

Non-interactively:

- By editing a file and running the code afterwards.

- Interpreters are great for prototyping.
- But not really suitable if you want to share or release code.

Running Python Code

Interactively:

- Statement by statement, directly in the interpreter.

terminal

```
$ python
Python 3.7 (default, Nov 26 2019, 10:23:46)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> print('Hello world')
Hello world
>>>
```

Non-interactively:

- By editing a file and running the code afterwards.

first_script.py

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

- Interpreters are great for prototyping.
- But not really suitable if you want to share or release code.

Running Python Code

Interactively:

- Statement by statement, directly in the interpreter.

terminal

```
$ python
Python 3.7 (default, Nov 26 2019, 10:23:46)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> print('Hello world')
Hello world
>>>
```

Non-interactively:

- By editing a file and running the code afterwards.

first_script.py

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

terminal

```
$ python first_script.py
Hello world
$
```

- Interpreters are great for prototyping.
- But not really suitable if you want to share or release code.

Acknowledgements

Martijn Vermaat

Jeroen Laros

Jonathan Vis

