Leiden University
Medical Center

Python Programming

Introduction

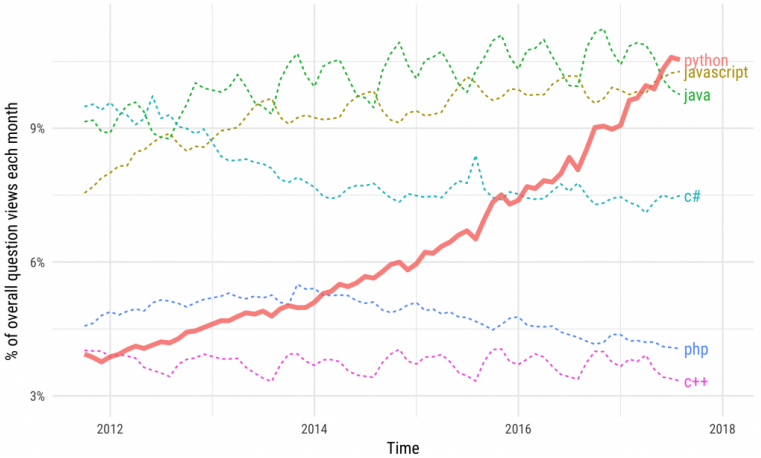
Mihai Lefter

Outline

Introduction
About Python
Running Python Code
This Course

Why Python?

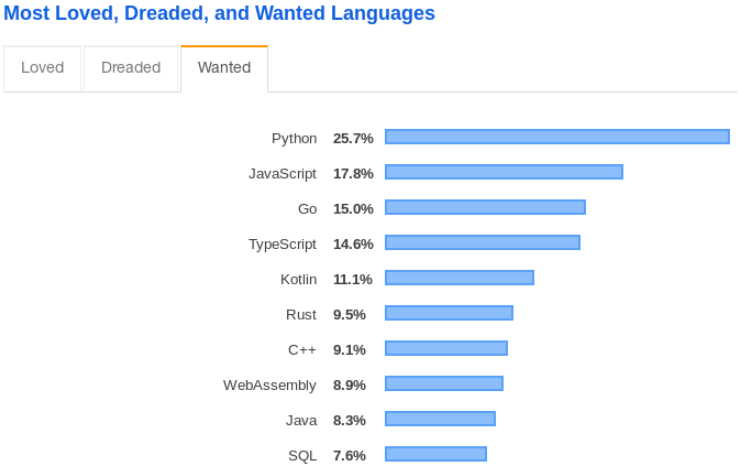
- Widely used with a large community.



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

Why Python?

- Widely used with a large community.



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

Why Python?

- Widely used with a large community.

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

Why Python?

- Widely used with a large community.
- Rich (scientific) libraries.
- Low barrier to entry.

History

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

CWI Centrum Wiskunde & Informatica

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.

Features

- General purpose, high-level programming language.

Features

- General purpose, high-level programming language.
- Interpreted, no separate compilation step needed.

Features

- General purpose, high-level programming language.
- Interpreted, no separate compilation step needed.
- Imperative and object-oriented programming.
  - And some functional programming.

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.

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.

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
>>>
```

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

Non-interactively:

- By editing a file and running the code afterwards.

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.

Non-interactively:

- By editing a file and running the code afterwards.

first_script.py

```
1 print("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.

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
$
```

This Course

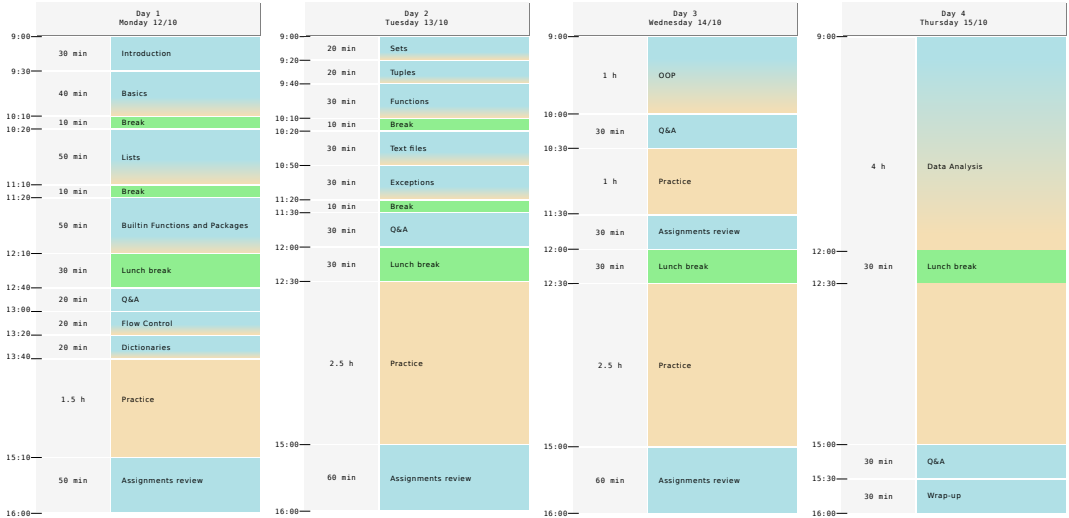
- Aimed at PhD students, Postdocs, researchers, analysts, ...
- Focus on:
  - Programming as a tool to do your research.
  - Basic understanding of Python.

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.

Program


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

Material

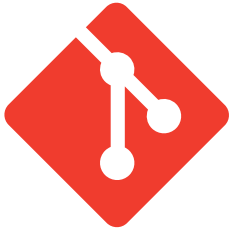
https://github.com/LUMC/python-course

Practical Sessions

- 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.

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.



Getting help

- Ask a question.

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

Leiden University
Medical Center

Acknowledgements

Martijn Vermaat
Jeroen Laros
Jonathan Vis