Leiden University
Medical Center

Python Programming

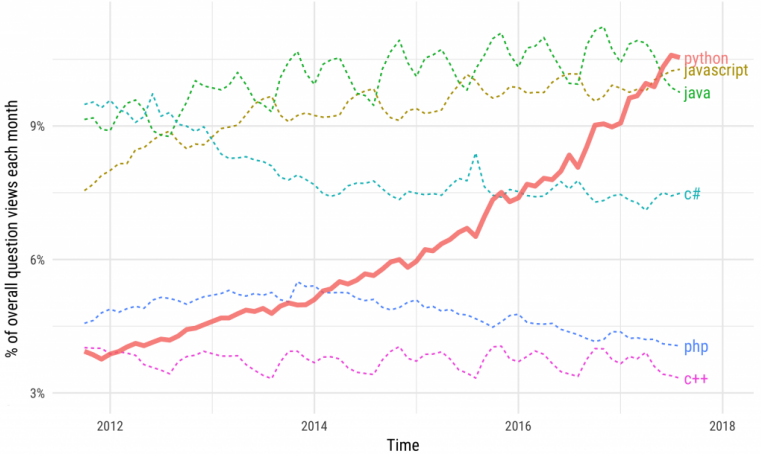
Introduction

Mihai Lefter

Outline

Why Python?

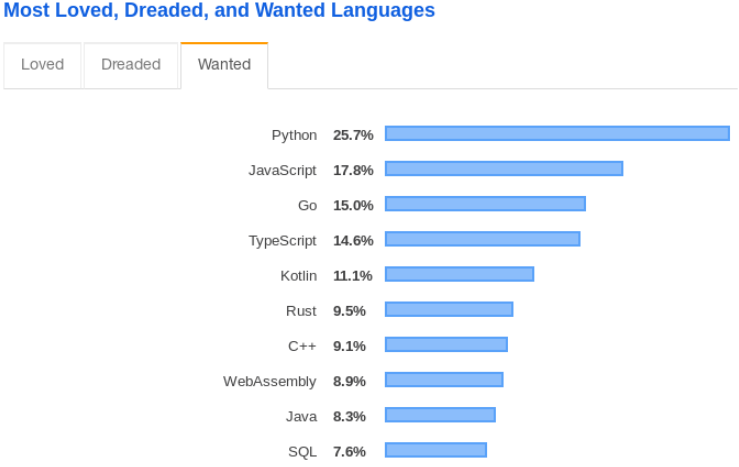
- Widely used with a large community.



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

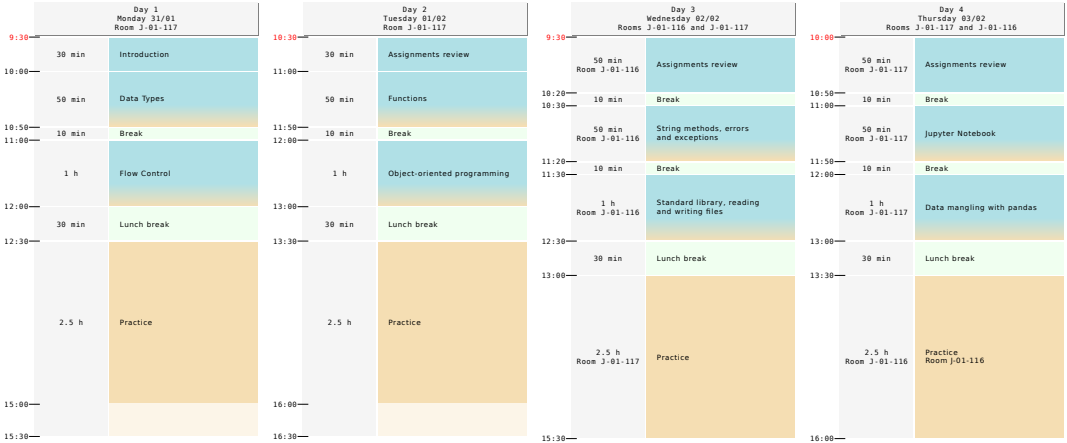
- Aimed at PhD students, Postdocs, researchers, analysts, ...
- Focus on:
  - Programming as a tool to do your research.
  - Basic understanding of Python.

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 31/01
Room J-01-117
9:30
30 min
Introduction
10:00
50 min
Data Types
10:50
10 min
Break
11:00
1 h
Flow Control
12:00
30 min
Lunch break
12:30
2.5 h
Practice
15:00
15:30
Day 2
Tuesday 01/02
Room J-01-117
10:30
30 min
Assignments review
11:00
50 min
Functions
11:50
10 min
Break
12:00
1 h
Object-oriented programming
13:00
30 min
Lunch break
13:30
2.5 h
Practice
16:00
16:30
Day 3
Wednesday 02/02
Rooms J-01-116 and J-01-117
9:30
50 min
Room J-01-116
Assignments review
10:20
10 min
Break
10:30
50 min
Room J-01-116
String methods, errors and exceptions
11:20
10 min
Break
11:30
1 h
Room J-01-116
Standard library, reading and writing files
12:30
30 min
Lunch break
13:00
2.5 h
Room J-01-117
Practice
15:30
Day 4
Thursday 03/02
Rooms J-01-117 and J-01-116
10:00
50 min
Room J-01-117
Assignments review
10:50
10 min
Break
11:00
50 min
Room J-01-117
Jupyter Notebook
11:50
10 min
Break
12:00
1 h
Room J-01-117
Data mangling with pandas
13:00
30 min
Lunch break
13:30
2.5 h
Room J-01-116
Practice
Room J-01-116
16:00

Material

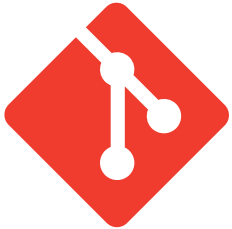
https://git.lumc.nl/courses/programming-course/-/blob/master/README.md

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.

Leiden University Medical Center

Acknowledgements

Martijn Vermaat
Jeroen Laros
Jonathan Vis