

Python Programming

Introduction

Mihai Lefter



Outline

Introduction

About Python

Running Python Code

Python as a Calculator

Variables

Python's Type System

Ipython Magics

Hands On!

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.

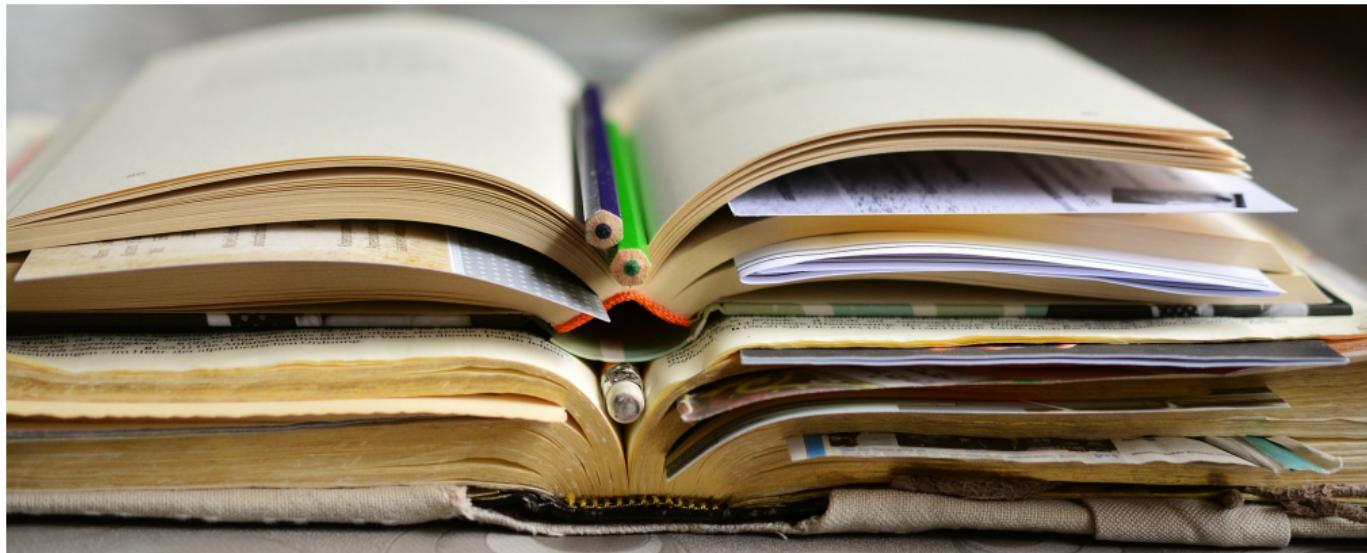


Introduction

Material

All course material is to be found here:

<https://git.lumc.nl/courses/programming-course>



Introduction

Hands On!

Programming is fun!

- You only learn programming by doing it.
- Lecture format:
 - Blended teaching + exercising.
- Have your laptop open during the lessons.
- Repeat the code from the slides, play around with it.
- Do the session exercises.
- There will be a few assignments.



Introduction

Teachers

- Mark Santcroos
m.a.santcroos@lumc.nl
- Jonathan Vis
j.k.vis@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

	Tuesday 26/11 J-1-83	Wednesday 27/11 P-5-34	Thursday 28/11 P-5-34	Friday 29/11 J-1-83
9:00 - 10:00	Welcome, introduction to Python <small>Mihai</small>	Assignments review <small>Mihai/Mark</small>	Assignments review <small>Mihai/Jonathan</small>	Assignments review <small>Mihai/Ruben</small>
10:00 - 11:00	Data types <small>Mark</small>	Functions <small>Mihai</small>	String methods, errors, and exceptions <small>Ruben</small>	Jupyter Notebook <small>Ruben</small>
11:00 - 12:00	Flow control <small>Mihai</small>	Object-oriented programming <small>Jonathan</small>	Standard library, reading, and writing files <small>Mihai</small>	Data mangling with pandas <small>Mihai</small>
12:00 - 13:00	Lunch break			
13:00 - 14:00				
14:00 - 17:00	Practical session			

Introduction

Program

	Tuesday 26/11 J-1-83	Wednesday 27/11 P-5-34	Thursday 28/11 P-5-34	Friday 29/11 J-1-83
		Restricted area: we meet at 8:45 in building 2 restaurant .		
9:00 - 10:00	Welcome, introduction to Python <small>Mihai</small>	Assignments review <small>Mihai/Mark</small>	Assignments review <small>Mihai/Jonathan</small>	Assignments review <small>Mihai/Ruben</small>
10:00 - 11:00	Data types <small>Mark</small>	Functions <small>Mihai</small>	String methods, errors, and exceptions <small>Ruben</small>	Jupyter Notebook <small>Ruben</small>
11:00 - 12:00	Flow control <small>Mihai</small>	Object-oriented programming <small>Jonathan</small>	Standard library, reading, and writing files <small>Mihai</small>	Data mangling with pandas <small>Mihai</small>
12:00 - 13:00	Lunch break			
13:00 - 14:00				
14:00 - 17:00	Practical session			

Software requirements

- Anaconda:
 - Python 3.7.
 - Comes with all that's required:
 - Python interpreter.
 - Jupyter Notebook.
 - Libraries: NumPy, Panda, matplotlib, Bokeh, Biopython, ...
 - [Installation instructions.](#)
- Git:
 - [Installation instructions.](#)



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 (we will be around in the afternoon).
- If it's private, mail one of the teachers.



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.

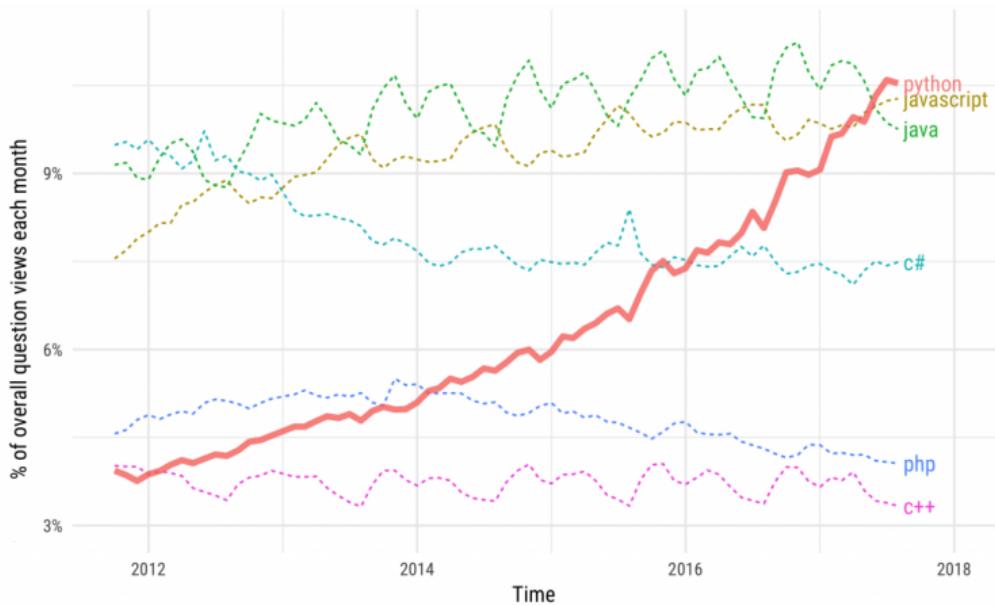
We'll come back to most of this.



About Python

Why Python?

- Widely used with a large community.



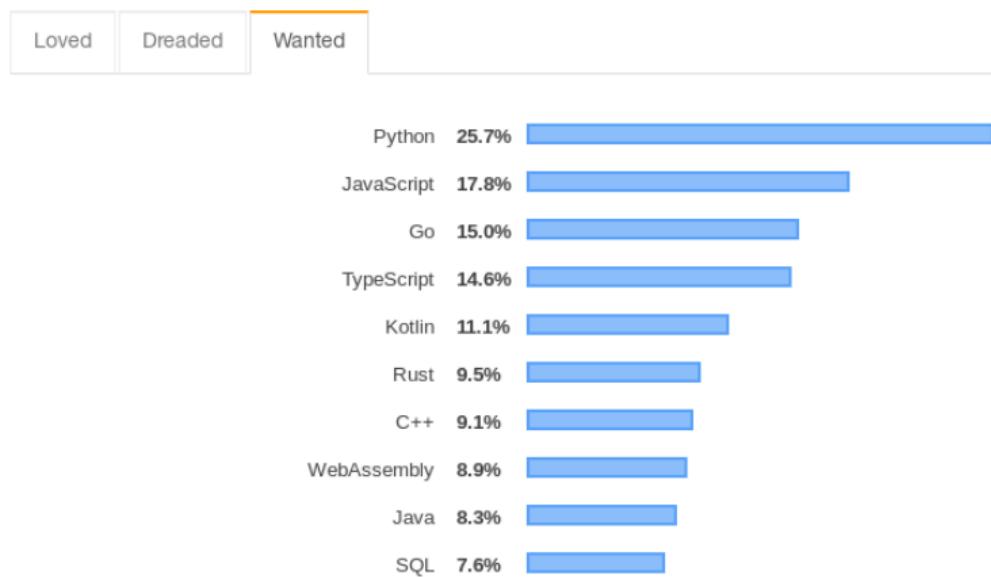
<https://stackoverflow.blog/2017/09/06/incredible-growth-python/>

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>

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.
- Python 2.7 will not be maintained past 2020.

We'll use Python 3.7 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
>>>
```

- 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
$
```

The IPython interpreter

Similar to the standard Python interpreter, but with:

- syntax highlighting;
- tab completion;
- cross-session history;
- etc.

terminal

```
$ ipython
Python 3.7 (default, Nov 12 2018, 13:43:14)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.
```

In [1]:

Python as a Calculator

Integers

IPython

```
In [1]: 17
Out[1]: 17

In [2]: (17 + 4) * 2 - 10
Out[2]: 32

In [3]: 14 // 4 # floor division - quotient
Out[3]: 3

In [4]: 14 % 4    # modulus - remainder
Out[4]: 2

In [5]: 2 ** 3    # exponentiation
Out[5]: 8
```

Python as a Calculator

Floating point numbers

IPython

```
In [6]: 3.2 * 18 - 2.1
```

```
Out[6]: 55.5
```

```
In [7]: 36 / 5
```

```
Out[7]: 7.2
```

Floating point numbers

Scientific notation:

IPython

```
In [8]: 1.3e20 + 2
```

```
Out[8]: 1.3e+20
```

```
In [9]: 1.3 * 10**20
```

```
Out[9]: 1.3e+20
```

Variables

- Names (identifiers) used to reference values. Can consist of:
 - the uppercase and lowercase letters `A` through `Z`;
 - the underscore `_`;
 - the digits `0` through `9`, except for the first character.
- Notes:
 - Python is case sensitive.
 - Reserved words (`if`, `then`, `for`, etc.) are not to be employed.

IPython

```
In [10]: a = 17
In [11]: leiden_population = 123856
In [12]: f18_speed = 1.8
```

Variables

- No need to declare them first or define the type.
- However, you do need to give variables values before you use them.

IPython

```
In [13]: a = 17
In [14]: c = a + b
-----
NameError                                 Traceback (most recent call last)
<ipython-input-2-1674f6151070> in <module>
----> 1 c = a + b

NameError: name 'b' is not defined
```

Python's Type System

Every value has a type

- View it using `type`.

IPython

```
In [15]: type(27)
```

```
Out[15]: int
```

```
In [16]: type(3 * 2)
```

```
Out[16]: int
```

```
In [17]: type(3 / 2)
```

```
Out[17]: float
```

```
In [18]: type(a)
```

```
Out[18]: int
```

Python's Type System

Some operations are defined on more than one type

- Possibly with different meanings.

IPython

```
In [19]: type(3 * 2.0)
Out[19]: float

In [20]: drinks = 'beer' * 5 + 'whiskey'
In [21]: drinks
Out[21]: 'beerbeerbeerbeerbeerwhiskey'

In [22]: type(drinks)
Out[22]: str
```

Dynamic typing

- At runtime variables can be assigned values of different types.

IPython

```
In [23]: a  
Out[23]: 17  
  
In [24]: type(a)  
Out[24]: int  
  
In [25]: a = 'spezi'  
In [26]: type(a)  
Out[26]: str
```

Python's Type System

Strongly typed

- Operations on values with incompatible types are forbidden.

IPython

```
In [27]: 'beer' + 34
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-17-ec918fbfdf41> in <module>()
      1 'beer' + 34
TypeError: Can't convert 'int' object to str implicitly
```

Ipython Magics

IPython

```
In [28]: %save introduction 1-25
The following commands were written to file 'introduction.py':
...
In [29]: %run introduction.py

In [30]: exit
```

Documentation of all the magics is to be found [here](#).

Hands On!

1. We've seen that $b = 2$ is legal.
 - a. What about $2 = b$?
 - b. How about $a = b = 1$?
2. In math notation you can multiply x and y like this: xy . What happens if you try that in Python?
3. How many seconds are there in 42 minutes and 42 seconds?
4. How many miles are there in 16 kilometers? (1 mile = 1.61 km)
5. Let's assume that you run a 42 km race in 4 hours 42 minutes and 42 seconds.
 - a. What is your average pace (time per mile in minutes and seconds)?
 - b. What is your average speed in miles per hour?
6. Use string operations to reference '**'tra la la la'**' in a variable named song.
7. If an article costs 249 Euros including the 19% Value Added Tax (VAT), what is the actual VAT amount in Euros for the corresponding article?

Acknowledgements

Martijn Vermaat

Jeroen Laros

Jonathan Vis

