



# Scripting Languages

Laszlo SZATHMARY

University of Debrecen

Faculty of Informatics

## Lab #1

- introduction
- string data type

(last update: 2021-01-31 [yyyy-mm-dd])

2020-2021, 2nd semester



# About the course

**Title of the Course:** Scripting Languages

**Course ID:** INBPA9942L

**Credits:** 3

**Pre-requisite:** Programming Languages 1

**Home Page of the Instructor:**

<https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=En.En>

**Time and classroom:**

- *Thursday* 8h-10h, IK-207

# Requirements

At the end of the semester you will get a **practical course mark**. For this, you will have to attend the labs. **Max. 3 absences** are tolerated. If you are absent more than 3 times, you will automatically fail the course.

There will be **two classroom tests**. The first one is on paper, while the second one is on computer. Your mark will be the average of the marks you get on the tests. If this mark is a real number (e.g. 3.5, 4.5, etc.), then I will take into consideration your lab work and homeworks. If someone solved less than 80% of the homeworks, then (s)he will get a worse mark than the average of the two tests.

If someone is not satisfied with his/her mark, (s)he will have the **possibility to improve** (or decline) the final mark in the last week of the semester. In this case, the final mark can be max. one mark better (or worse). The student will have to solve some programming exercises on a computer, and (s)he will have to know the concepts related to the Python programming language. If someone failed both classroom tests, (s)he cannot improve the final mark.

If you solve less than 50% of the homeworks, then your final mark will be 1 (fail). In this case there is no possibility to improve your mark.

# Bibliography

- *Guido van Rossum: Python Tutorial* (<https://docs.python.org/3/download.html>, also in PDF), 2020
- *Wesley J. Chun: Core Python Programming* (2nd Edition), 2006
- *Allen B. Downey: Think Python* (How to Think Like a Computer Scientist) <http://www.greenteapress.com/thinkpython/>, O'Reilly, 2012
- *Doug Hellmann: The Python Standard Library by Example* (Developer's Library), 2011  
[online version: **Python Module of the Week** (<https://pymotw.com/3/>)]

# Bibliography (cont.)

## Python 3

- *Mark Pilgrim: Dive Into Python 3* (<http://www.diveintopython3.net/>), 2009
- *Michael Driscoll: Python 101*, Leanpub, 2014 (beginner)
- *Michael Driscoll: Python 201*, Leanpub, 2016 (intermediate)

## Expert

- *Luciano Ramalho: Fluent Python*, O'Reilly, 2015

# Introduction



- Python is a general purpose, very high level programming language.
- Primary design goal: readability.
- Interpreted language, the code can be executed immediately.
- Multiparadigm (imperative, object-oriented, functional).
- The first version was released in 1991 and it was named after the Monty Python group.
- It was designed by Guido van Rossum, a Dutch researcher/programmer (born in 1956). 2005-2012: Google; 2013-2019: Dropbox. At the end of 2019 he retired, but at the end of 2020 he came back and joined Microsoft.
- What languages influenced Python: ABC, ALGOL 68, C, C++, Dylan, Haskell, Icon, Java, Lisp, Modula-3, Perl.
- What languages were influenced by Python: Boo, Cobra, D, Falcon, Groovy, JavaScript, Ruby, Go.

# Introduction



**Guido van Rossum**   
@gvanrossum



I decided that retirement was boring and have joined the Developer Division at Microsoft. To do what? Too many options to say! But it'll make using Python better for sure (and not just on Windows :-). There's lots of open source here. Watch this space.

6:00 PM · Nov 12, 2020 · Twitter Web App

**5.3K** Retweets   **2.2K** Quote Tweets   **38.4K** Likes

# Introduction

- Dynamic types and automatic garbage collection.
- Platform independent (Unix/Linux, Windows, Mac OS, etc.)
- Python has a large standard library („batteries included”), and there are thousands of freely available 3rd party modules<sup>1</sup>.
- The interpreter and the standard library are open source.
- Easy to learn, fun to use. It has a simple syntax. The source code can be read easily.
- It has efficient high-level data structures that help the work of programmers. Object-orientation is done simply and effectively.

<sup>1</sup> see <https://pypi.org> (On January 14, 2021 there were **283,496** packages available ; on September 5, 2020 there were **260,189** packages available)



# Introduction

- Ideal language for scripts and for rapid application development (RAD).
- Supports rapid prototyping.
- Similar programming languages: Perl, Ruby.
- A perfect choice for small scripts (ex. 10-20 lines), but it's also suitable for large projects with several thousands of lines of code! It has modules and packages, which allow us to keep the source code clear.
- For a long time, there were two branches that existed in parallel: Python 2 and Python 3 (Python 3 was forked in Dec. 2008). Python 2.7 is not maintained since January 1, 2020. If you start a new project, use Python 3.
- Here, in the labs we will use Python 3. The latest version is Python 3.9. Since Python 2 (2.7) is still widely used, the key differences will be mentioned. Recommended version: Python 3.6+

# Links

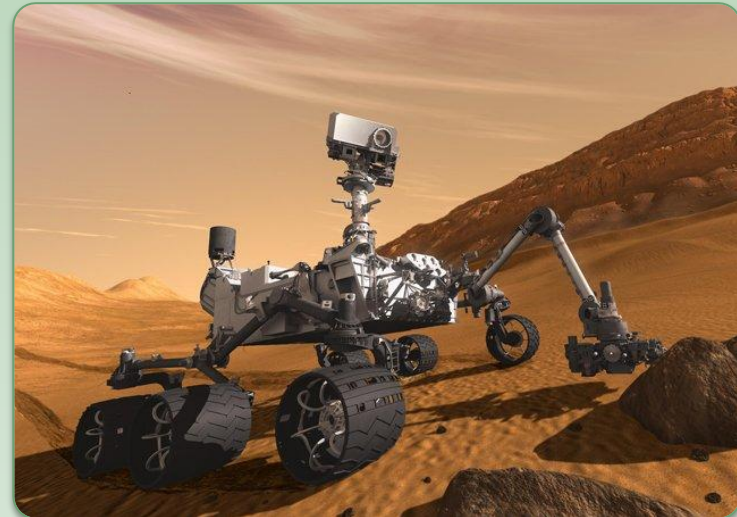
- Python HQ: <https://www.python.org/>
- Python documentation: <https://docs.python.org/>
- The Python Standard Library: <https://docs.python.org/library/>
- Python FAQ: <https://docs.python.org/faq/general.html>
- PEP 8 -- Style Guide for Python Code:  
<https://www.python.org/dev/peps/pep-0008/>
- <https://www.reddit.com/r/learnpython>
- <https://www.reddit.com/r/python>
- <https://stackoverflow.com/questions/tagged/python>

# Where is it used?

- Python success stories: <https://www.python.org/about/success/>

- **Scientific**

- [Biology](#)
- [Bioinformatics](#)
- [Computational Chemistry](#)
- [Data Visualization](#)
- [Drug Discovery](#)
- [GIS and Mapping](#)
- [Scientific Programming](#)
- [Simulation](#)
- [Weather](#)



Mars Curiosity (August 6, 2012)

Software: 2.5 million C lines.

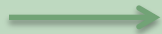
Log files were tested with Python scripts.

- Google (C, C++, Java, Python, Go, Dart, etc.)  
„Python where we can,  
C++ where we must”  
([link](#))

# How popular?



TIOBE index (<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>)



Jan 2021	Jan 2020	Change	Programming Language	Ratings	Change
1	2	⬆	C	17.38%	+1.61%
2	1	⬇	Java	11.96%	-4.93%
3	3		<b>Python</b>	<b>11.72%</b>	<b>+2.01%</b>
4	4		C++	7.56%	+1.99%
5	5		C#	3.95%	-1.40%
6	6		Visual Basic	3.84%	-1.44%
7	7		JavaScript	2.20%	-0.25%
8	8		PHP	1.99%	-0.41%
9	18	⬆	R	1.90%	+1.10%
10	23	⬆	Groovy	1.84%	+1.23%
11	15	⬆	Assembly language	1.64%	+0.76%
12	10	⬇	SQL	1.61%	+0.10%
13	9	⬇	Swift	1.43%	-0.36%
14	14		Go	1.41%	+0.51%
15	11	⬇	Ruby	1.30%	+0.24%
16	20	⬆	MATLAB	1.15%	+0.41%
17	19	⬆	Perl	1.02%	+0.27%
18	13	⬇	Objective-C	1.00%	+0.07%
19	12	⬇	Delphi/Object Pascal	0.79%	-0.20%
20	16	⬇	Classic Visual Basic	0.79%	-0.04%

Jobs: <http://careers.stackoverflow.com/jobs?searchTerm=python>

# Literature



amazon

Deliver to Budapest 1113

python programming

1-16 of over 4,000 results for "python programming"

**Department**

- Books
  - Python Programming
  - Introductory & Beginning Programming
  - Computers & Technology
  - Computer Programming
  - Programming Languages
- Kindle Store
  - Python Computer Programming
  - Computer Programming

See All 3 Departments

**Avg. Customer Review**

- ★★★★★ & Up
- ★★★★☆ & Up
- ★★★☆☆ & Up
- ★★☆☆☆ & Up

**Computer User Books**

- Beginners & Seniors
- Advanced & Power Users
- Kids & Teens

**Book Format**

- Paperback
- Kindle Edition
- Audible Audiobook
- Large Print

**Amazon Global Store**

☐ Amazon Global Store

**Condition**

Now

**Best Seller**

**PYTHON CRASH COURSE**  
2ND EDITION  
A HANDS-ON, PROJECT-BASED INTRODUCTION TO PROGRAMMING  
ERIC MATTHES  
OVER 500,000 COPIES SOLD

Python Crash Course, 2nd Edition: A Hands-On, Project-Based  
by Eric Matthes | May 3, 2019  
★★★★☆ ~ 1,232

**Paperback**  
\$17<sup>00</sup> ~~\$39.95~~  
Ships to Hungary  
More Buying Choices  
\$13.29 (38 used & new offers)

**Kindle**  
\$24<sup>14</sup> ~~\$33.48~~  
Available instantly

**Best Seller**

**Learning Python, 5th Edition**  
by Mark Lutz | Jul 9, 2013  
★★★★☆ ~ 755

**Paperback**  
\$44<sup>94</sup> ~~\$74.99~~  
Ships to Hungary  
More Buying Choices  
\$31.78 (50 used & new offers)

**Kindle**  
\$12<sup>25</sup> to rent  
\$30.23 to buy  
Available instantly

(last checked on January 14, 2021)



# Conferences

PyCon US is the largest Python conference (<https://us.pycon.org/>).

## PyCon US 2020 (organized online)

- presentations : <https://pyvideo.org/events/pycon-us-2020.html> (90 videos)
- <https://www.youtube.com/c/PyCon2020/videos>
- there are several *tutorials* among the presentations

## PyCon 2019

- presentations : <https://pyvideo.org/events/pycon-us-2019.html> (146 videos)
- <https://www.youtube.com/channel/UCxs2IIVXaEHHA4BtTiWZ2mQ/playlists>
- there are several *tutorials* among the presentations

<https://pyvideo.org/> collects the videos of Python conferences.

# Quiz

What's the name of the creator of Python?

- ☐ Larry Wall
- ☐ Yukihiro Matsumoto
- ☐ Guido van Rossum
- ☐ Rasmus Lerdorf

**Homework:** Look after the others. Who are they?

## Using the interpreter:

```
[11:57:44] ~ $ python3
Python 3.6.5 (default, Apr 1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

## Writing a script:

```
1 #!/usr/bin/env python3
2
3 print("Hello, World!")
```

In Python 2, if you use accented characters, then you **must** add this line.

In Python 3, under Linux, this encoding is the default, thus this line can be omitted.

## Using special characters:

```
1 #!/usr/bin/env python3
2 # coding: utf-8
3
4 def main():
5     # using special characters
6     print("Jyväskylä")
```



```
4  >>> a = 6
5  >>> a
6  6
7  >>> a = "hello"
8  >>> len(a)
9  5
10 >>> a
11 'hello'
12 >>> A
13 Traceback (most recent call last):
14   File "<stdin>", line 1, in <module>
15 NameError: name 'A' is not defined
16 >>> "hello " + "world"
17 'hello world'
18 >>> "hello " + 6
19 Traceback (most recent call last):
20   File "<stdin>", line 1, in <module>
21 TypeError: cannot concatenate 'str' and 'int' objects
22 >>> "hello " + str(6)
23 'hello 6'
```

no need to declare variables

def

```
1 #!/usr/bin/env python3
2
3
4 def main():
5     print("Hello, World!")
6
7
8 main()
```

colon

indenting

no semicolon

**Style:** leave 2 empty lines before and after a function.

```
1 #!/usr/bin/env python3
2
3
4 def main():
5     print("Hello, World!")
6
7
8 if __name__ == "__main__":
9     main()
```

} Executed directly  
or  
called as a module?

Print the command-line arguments:

From now on, we will omit  
the first line:

```
3 import sys
4
5
6 def main():
7     print(sys.argv)
8
9
10 if __name__ == "__main__":
11     main()
```

`#!/usr/bin/env python3`

**Then:** provide a name as an argument (ex. `./hello.py Bob`),  
and greet the person („Hello Bob!”).

```
4 import sys
5
6 def hello(name):
7     if name == "Batman" or name == "Robin":
8         print("Batman or Robin")
9     else:
10        print(NoSuchFunction())
11
12 def main():
13     hello(sys.argv[1])
14
15 if __name__ == "__main__":
16     main()
```

no parenthesis after the if

We only get an error  
if code execution  
gets here!

One more reason to do unit tests for larger programs.  
Every branch must be tested!

## A general template for Python 3 scripts

```
1 #!/usr/bin/env python3
2 # encoding: utf-8
3
4
5 def main():
6     print('Py3')
7
8 #####
9
10 if __name__ == "__main__":
11     main()
```

**Tip:** save this file under the name `basic.py`, then if you want to write a new Python script, just make a copy of this file.

You can find this template here: <http://bit.ly/307H2PS>

```
>>> print "hello"
File "<input>", line 1
    print "hello"
          ^
SyntaxError: Missing parentheses in call to 'print'
>>>
>>> print("hello")
hello
>>>
>>> 7 / 2
3.5
>>>
>>> 7 // 2
3
>>>
```

most important  
changes  
in Python 3



```
>>> print "hello"
hello
>>>
>>> 7 / 2
3
>>>
>>> 7 // 2
3
>>>
```

# Strings

```
4 >>> s = "Hello"
5 >>> s
6 'Hello'
7 >>> s = 'Hello'
8 >>> s
9 'Hello'
10 >>> s = "isn't"
11 >>> s
12 "isn't"
13 >>> s = 'he said: "go home"'
14 >>> s
15 'he said: "go home"'
16 >>> s = "he said: \"go home\""
17 >>> s
18 'he said: "go home"'
19 >>> s = 'batman'
20 >>> len(s)
21 6
22 >>> s[0]
23 'b'
24 >>> s[0] = 'B'
25 Traceback (most recent call last):
26   File "<stdin>", line 1, in <module>
27   TypeError: 'str' object does not support item assignment
28 >>> s
29 'batman'
30 >>> s + '!'
31 'batman!'
32 >>> s = 'Joker'
33 >>> s.lower()
34 'joker'
35 >>> s.upper()
36 'JOKER'
37 >>> s.find('k')
38 2
39 >>> s.find('a')
40 -1
41 >>> s[20]
42 Traceback (most recent call last):
43   File "<stdin>", line 1, in <module>
44   IndexError: string index out of range
```

## String methods:

<http://docs.python.org/library/stdtypes.html#string-methods>

<https://goo.gl/uBQPYA>

strings are *immutable* objects  
(read-only)

**Homework:** select a string method and write a simple program that demonstrates the usage of this method.

## Some frequently used string methods

`s.lower()`, `s.upper()`

returns a lowercase, uppercase version of the string

`s.strip()`

removes the whitespace characters from both ends of the string

`s.isalpha()` / `s.isdigit()` / `s.isspace()`...

verifies if all characters of the string belong to the given character class

`s.startswith('other')`, `s.endswith('other')`

verifies if the string starts / ends with the other string

`s.find('other')`

Does the string include the other? If yes, return the index of the first character's occurrence. If not, return -1.

`s.replace('old', 'new')`

in the string replace all occurrences of 'old' with 'new'

`s.split('delim')`

Splits a string by a delimiter. Returns a list. See later.

`s.join(list)`

Opposite of *split*. Concatenates a list of strings by a delimiter. See later.



# Python is another tool



Consider Python as a new tool on your toolbelt.

Analyze the problem and choose the most appropriate tool.

# Eastern wisdom

*"I hear and I forget. I see and I remember. I do and I understand."*

Confucius



That is: practice [1], practice [2] and practice [3][4][5]...

[1] <https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=En.PyExercises>

[2] <http://www.pythonchallenge.com/>

[3] <http://projecteuler.net/> (mainly mathematical exercises)

[4] <https://www.hackerrank.com>

[5] <https://adventofcode.com/>

One more tip: Try to solve the exercises of the Programming 2 course in Python.  
And vice versa: try to solve the exercises that you get here in Java / C# / etc.

# Tips and tricks

Some useful aliases:

```
# ~/.bashrc
alias p2='python2'
alias p3='python3'
alias p='python3'

alias ..='cd ..'
alias ...='cd ../; cd ..'
# can be continued...
```

Don't type unnecessarily...



# Homework

1. [20121001a] string method