Introduction to the Python programming language

Laszlo SZATHMARY
University of Debrecen
Faculty of Informatics

Lab #1

- introduction
- string data type

(last update: 2017-08-03 [yyyy-mm-dd])

About the course

Title of the Course: Introduction to the Python Programming Language

Course ID: INGV381L

The course is optional and available for all BSc CS students.

Pre-requisite: Programming Languages 1 (IN[GHJ]K301)

Home Page of the Course:

https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=En.Py2017sept

Time and classroom:

- *Wednesday* 8h-10h, IK-103

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.

Bibliography

- *Guido van Rossum*: **Python Tutorial** (https://docs.python.org/3/download.html, also in PDF), 2017
- *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; since January 2013: Dropbox.
- 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

- 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 [1].
- 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.

[1] see http://pypi.python.org (on August 3, 2017 **113,795** packages were 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.
- There are two branches: Python 2 and Python 3 (since Dec. 2008). Python 2.7 is stable and widely supported. However, current and future developments will concentrate on Python 3. If you start a new project, use Python 3.
- Here, in the labs we will use Python 3. The latest version is Python 3.6. Since Python 2 (2.7) is still widely used, the key differences will be mentioned.

Links

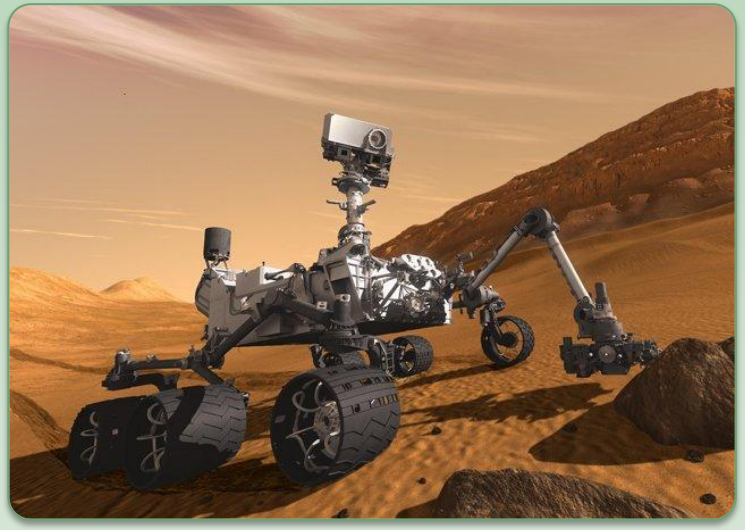
- Python HQ: http://www.python.org/
- Python documentation: http://docs.python.org/
- The Python Standard Library: http://docs.python.org/library/
- Python FAQ: http://docs.python.org/faq/general.html
- PEP 8 -- Style Guide for Python Code: http://www.python.org/dev/peps/pep-0008/

- http://www.reddit.com/r/learnpython
- http://www.reddit.com/r/python
- http://stackoverflow.com/questions/tagged/python

Where is it used?

- Python success stories: http://www.python.org/about/success/

- Scientific
  - Biology
  - Bioinformatics
  - Computational Chemistry
  - Data Visualization
  - Drug Discovery
  - GIS and Mapping
  - Scientific Programing
  - 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)
  „Python where we can,
  C++ where we must”
  (link)

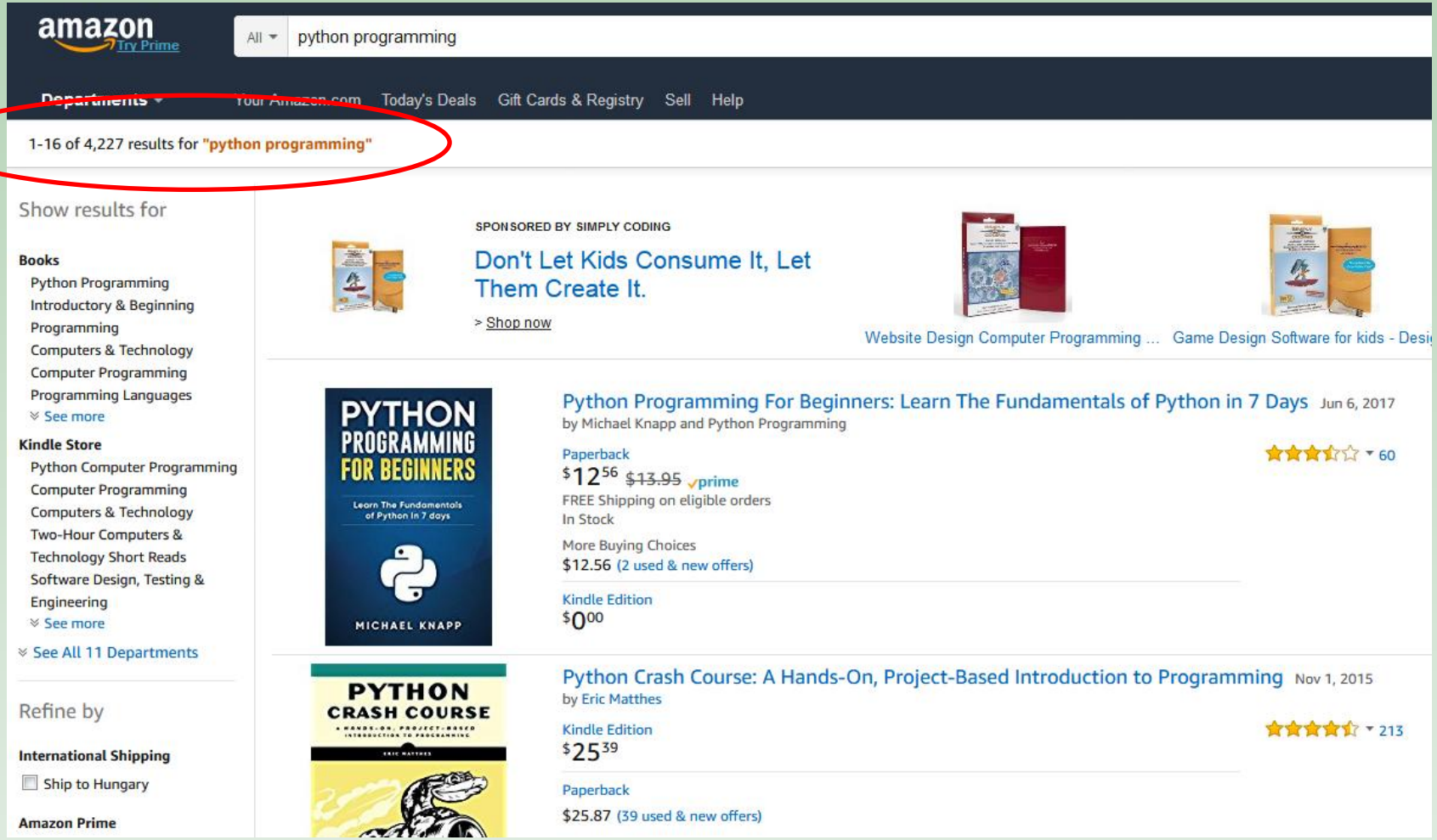
How popular?

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

Aug 2017	Aug 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.961%	-6.05%
2	2		C	6.477%	-4.83%
3	3		C++	5.550%	-0.25%
4	4		C#	4.195%	-0.71%
5	**5**		**Python**	**3.692%**	**-0.71%**
6	8	⌃	Visual Basic .NET	2.569%	+0.05%
7	6	⌄	PHP	2.293%	-0.88%
8	7	⌄	JavaScript	2.098%	-0.61%
9	9		Perl	1.995%	-0.52%
10	12	⌃	Ruby	1.965%	-0.31%
11	14	⌃	Swift	1.825%	-0.16%
12	11	⌄	Delphi/Object Pascal	1.825%	-0.45%
13	13		Visual Basic	1.809%	-0.24%
14	10	⌄⌄	Assembly language	1.805%	-0.56%
15	17	⌃	R	1.766%	+0.16%
16	20	⌃⌃	Go	1.645%	+0.37%
17	18	⌃	MATLAB	1.619%	+0.08%
18	15	⌄	Objective-C	1.505%	-0.38%
19	22	⌃	Scratch	1.481%	+0.43%
20	26	⌃⌃	Dart	1.273%	+0.30%

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

Literature



(last checked on August, 2017)

Conferences

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

PyCon 2017

- presentations : http://pyvideo.org/events/pycon-us-2017.html (143 videos)
- there are several *tutorials* among the presentations

PyCon 2016

- presentations : http://pyvideo.org/events/pycon-us-2016.html (186 videos)
- there are several *tutorials* among the presentations

http://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:

```
$ python3
Python 3.5.2 (default, Jul  5 2016, 12:43:10)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> "Hello, World!"
'Hello, World!'
>>>
```

Writing a script:

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

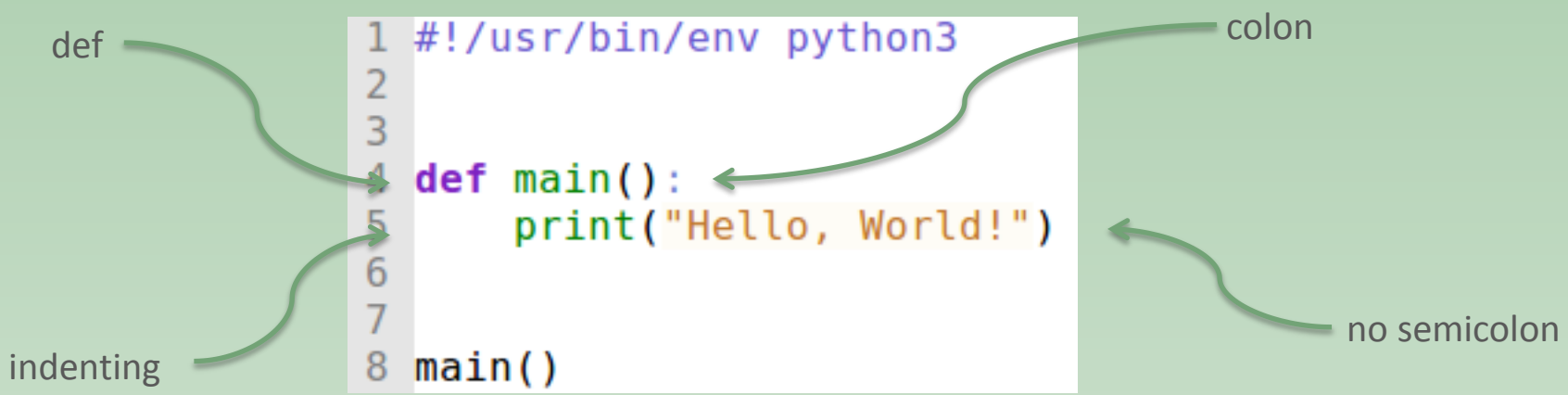
In Python 2, if you use accented characters, then you **must** add this line.
In Python 3, under Linux, this encoding is the default, thus this line can be omitted.

Using special characters:

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

```
>>> a = 6
>>> a
6
>>> a = "hello"
>>> len(a)
5
>>> a
'hello'
>>> A
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'A' is not defined
>>> "hello " + "world"
'hello world'
>>> "hello " + 6
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> "hello " + str(6)
'hello 6'
```

no need to declare variables



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:

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

From now on we will omit the first line:

```
#!/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.

More templates: https://goo.gl/lEL0Yy

```
>>> 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
>>>
```

function

mathematical division

integer division (always)

most important changes in Python 3


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

statement

integer division (like in C)

integer division (always)

Strings

```
>>> s = "Hello"
>>> s
'Hello'
>>> s = 'Hello'
>>> s
'Hello'
>>> s = "isn't"
>>> s
"isn't"
>>> s = 'he said: "go home"'
>>> s
'he said: "go home"'
>>> s = "he said: \"go home\""
>>> s
'he said: "go home"'
>>> s = 'batman'
>>> len(s)
6
>>> s[0]
'b'
>>> s[0] = 'B'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> s
'batman'
>>> s + '!'
'batman!'
>>> s = 'Joker'
>>> s.lower()
'joker'
>>> s.upper()
'JOKER'
>>> s.find('k')
2
>>> s.find('a')
-1
>>> s[20]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
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]...

[1] https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=En.PyExercises
[2] http://www.pythonchallenge.com/
[3] http://projecteuler.net/
[4] https://www.hackerrank.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