Introduction to the Python programming language

Laszlo SZATHMARY
University of Debrecen
Faculty of Informatics

Lab #1
- introduction
- string data type

(last update: 2016-09-20 [yyyy-mm-dd])

2016-2017, 1st semester

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.Py2016sept

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** (http://docs.python.org/tutorial/index.html, PDF), 2016

- *Wesley J. Chun*: **Core Python Programming** (2nd Edition), 2006

- *Doug Hellmann*: **The Python Standard Library by Example** (Developer's Library), 2011 [online version: **Python Module of the Week** (http://www.doughellmann.com/PyMOTW/contents.html)]

- *Zed A. Shaw*: **Learn Python The Hard Way** (http://learnpythonthehardway.org/), 2011

- *Allen B. Downey*: **Think Python** (How to Think Like a Computer Scientist) http://www.greenteapress.com/thinkpython/, O'Reilly, 2012

- *Mark Summerfield*: **Rapid GUI Programming with Python and Qt** (Chapter 1), 2007

Bibliography (cont.)

Python 3

- *Mark Pilgrim*: **Dive Into Python 3** (http://www.diveintopython3.net/), 2009
- *Mark Summerfield*: **Programming in Python 3**, 2nd Ed., 2009

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 Sept. 2, 2016 **87,794** 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.

- Here, in the labs we will use Python 2.7, but we will also talk a bit about Python 3 (at this introductory level there are no big differences between Python 2 and Python 3).

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/
- Python 2 or Python 3? http://wiki.python.org/moin/Python2orPython3

- 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

- Google (C, C++ / Java / Python / Go)
  „Python where we can,
  C++ where we must"
  (link)
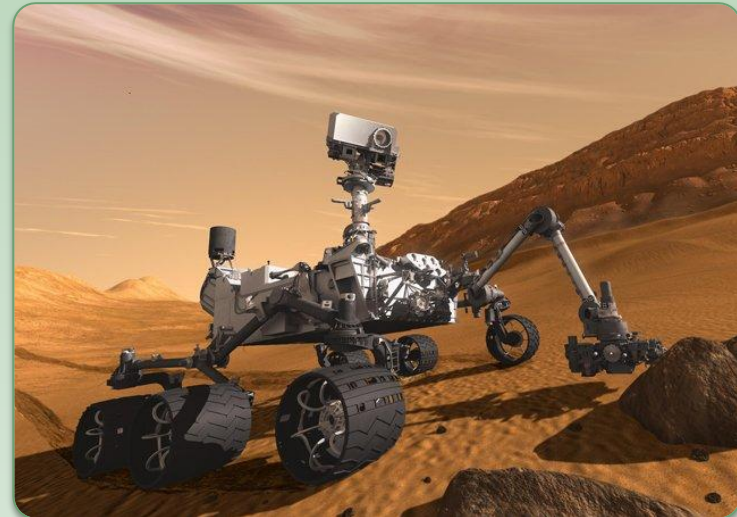
Mars Curiosity (August 6, 2012)

Software: 2.5 million C lines.

Log files were tested with Python scripts.

How popular?

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

Aug 2016	Aug 2015	Change	Programming Language	Ratings	Change
1	1		Java	19.010%	-0.26%
2	2		C	11.303%	-3.43%
3	3		C++	5.800%	-1.94%
4	4		C#	4.907%	+0.07%
5	5		Python	4.404%	+0.34%
6	7	^	PHP	3.173%	+0.44%
7	9	^	JavaScript	2.705%	+0.54%
8	8		Visual Basic .NET	2.518%	-0.19%
9	10	^	Perl	2.511%	+0.39%
10	12	^	Assembly language	2.364%	+0.60%
11	14	^	Delphi/Object Pascal	2.278%	+0.87%
12	13	^	Ruby	2.278%	+0.86%
13	11	v	Visual Basic	2.046%	+0.26%
14	17	^	Swift	1.983%	+0.80%
15	6	vv	Objective-C	1.884%	-1.31%
16	37	^^	Groovy	1.637%	+1.27%
17	20	^	R	1.605%	+0.60%
18	15	v	MATLAB	1.538%	+0.31%
19	19		PL/SQL	1.349%	+0.21%
20	95	^^	Go	1.270%	+1.19%

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

Literature

Conferences

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

PyCon 2016

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

PyCon 2015

- presentations: http://pyvideo.org/events/pycon-us-2015.html (134 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:

```
Python 2.7.5+ (default, Sep 19 2013, 13:48:49)
[GCC 4.8.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Persistent session history and tab completion are enabled.
>>> "Hello, World!"
'Hello, World!'
>>>
```

Writing a script:

```
1   #!/usr/bin/env python
2
3   print "Hello, World!"
```

Using special characters:

```
1   #!/usr/bin/env python
2   # coding: utf-8
3
4   def main():
5       # special char. in text:
6       print "Voilà!"
```

```
 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

colon

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

no semicolon

indenting

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

Executed directly
or
called as a module?

Print the command-line arguments:

From now on we will omit the first line:

`#!/usr/bin/env python`

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

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

```python
 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 2 scripts

```
1   #!/usr/bin/env python
2   # encoding: utf-8
3
4
5   def main():
6       print 'Py2'
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://dl.dropboxusercontent.com/u/144888/teaching/python/skeletons/index.html

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

strings are *immutable* objects (read-only)

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

21

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.

Homework

1. [20121001a] string method