Programming 1

Laszlo SZATHMARY
University of Debrecen
Faculty of Informatics

Lecture #1

- introduction
- string data type

(last update: 2026-02-08 [yyyy-mm-dd])

About the course

Title of the Course: Programming 1

Course ID: INBGA0212 (for Business Informatics BSc students)

Classes/week: 2 + 0 + 2

Credits: 6

Status: Obligatory

Assessment: Practical mark

Pre-requisite: INBGA0105 (Introduction to Programming)

Home Page of the Instructor:

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

Time and classroom of the lecture: *Monday* 14h-16h, IK-201

Requirements

- Attendance (max. 3 missed labs)
- Achieving at least 60% of the total possible points. You can collect points with 3 mid-term tests (2 practical + 1 theoretical). Each test has 33.3% weight.
- Solving at least 50% of the homeworks.

Practical tests: on computer. You get some exercises that you need to solve (write programs).

Theoretical test: in eLearning. Quiz-like test.

Grade scale:

% of points	Practical mark	
[0 - 59]%	1	fail
[60 - 69]%	2	pass
[70 - 79]%	3	
[80 - 89]%	4	
[90 - 100]%	5	

Requirements

If you fail, there will be an extra mid-term test (last chance), where you can try to improve your grade. However, in this case the best grade can be **maximum 2**.

This extra test will be a practical test on computer.

Bibliography

- *Guido van Rossum*: **Python Tutorial** (https://docs.python.org/3/download.html), 2026
  PDF version: https://docs.python.org/3.13/download.html (150 pages)
- *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** (https://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 [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 https://pypi.org (on January 16, 2026 there were **725,576** packages available; on August 14, 2025 there were **667,497** 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 3.14 . Since Python 2 (2.7) is still used, the key differences will be mentioned. Recommended version: Python 3.13+

Links

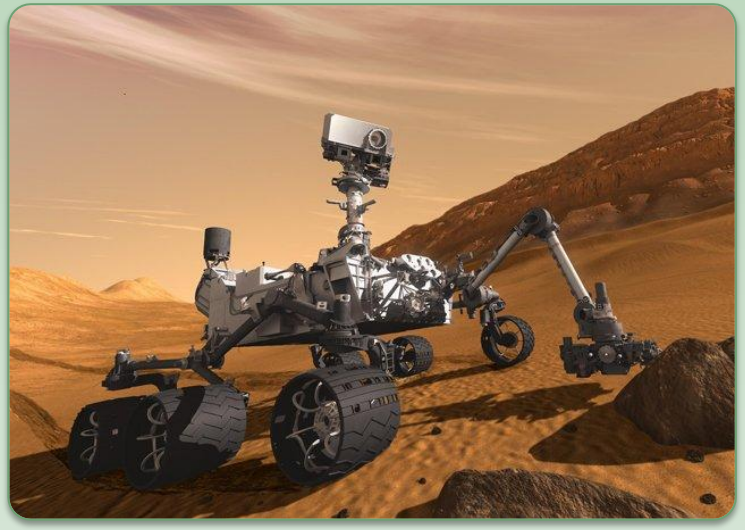
- Python HQ: https://www.python.org/
- Python documentation: https://docs.python.org/
- The Python Standard Library: https://docs.python.org/library/
- Python FAQ: https://docs.python.org/faq/general.html
- PEP 8 -- Style Guide for Python Code: https://www.python.org/dev/peps/pep-0008/

- 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 Programing
  - Simulation
  - Weather



Mars Curiosity (August 6, 2012)

Software: 2.5 million C lines

Log files were tested with Python scripts.

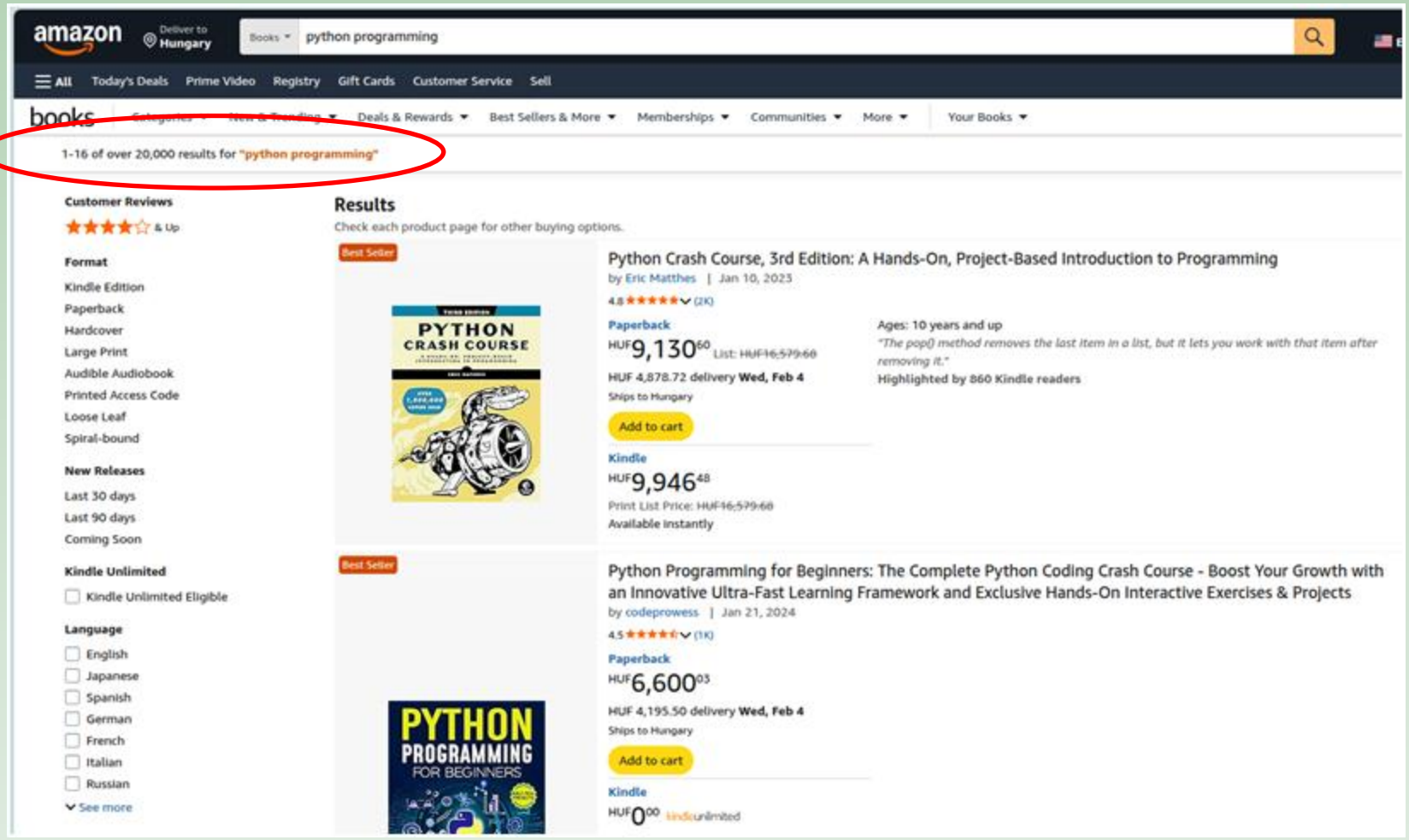
- Google (C, C++, Java, Python, Go, Dart, etc.)
  „Python where we can,
  C++ where we must”
  (link)

How popular?

TIOBE index (https://www.tiobe.com/tiobe-index/)

Jan 2026	Jan 2025	Change	Programming Language	Ratings	Change
1	1		Python	22.61%	-0.68%
2	4	↑	C	10.99%	+2.13%
3	3		Java	8.71%	-1.44%
4	2	↓	C++	8.67%	-1.62%
5	5		C#	7.39%	+2.94%
6	6		JavaScript	3.03%	-1.17%
7	9	↑	Visual Basic	2.41%	+0.04%
8	8		SQL	2.27%	-0.14%
9	11	↑	Delphi/Object Pascal	1.98%	+0.19%
10	18	⇈	R	1.82%	+0.61%
11	32	⇈	Perl	1.63%	+1.14%
12	10	↓	Fortran	1.61%	-0.42%
13	14	↑	Rust	1.51%	+0.34%
14	15	↑	MATLAB	1.40%	+0.34%
15	13	↓	PHP	1.38%	-0.00%
16	7	⇊	Go	1.24%	-1.37%
17	12	⇊	Scratch	1.24%	-0.31%
18	26	⇈	Ada	1.19%	+0.54%
19	17	↓	Assembly language	1.07%	+0.05%
20	25	⇈	Kotlin	0.97%	+0.23%

Literature



https://www.amazon.com/s?k=python+programming

(last checked on January 17, 2026)

Conferences

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

PyCon US 2025

- 118 videos
- https://www.youtube.com/playlist?list=PL2Uw4_HvXqvb98mQjN0-rYQjdDxJ_hcrs
- there are several *tutorials* among the presentations

PyCon US 2024

- 211 videos
- https://www.youtube.com/playlist?list=PL2Uw4_HvXqvYhjub9bw4uDAmNtprgAvlJ
- 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:

```
$ python3
Python 3.13.5 (main, Jun 21 2025, 09:35:00) [GCC 15.1.1 20250425] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Writing a script:

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

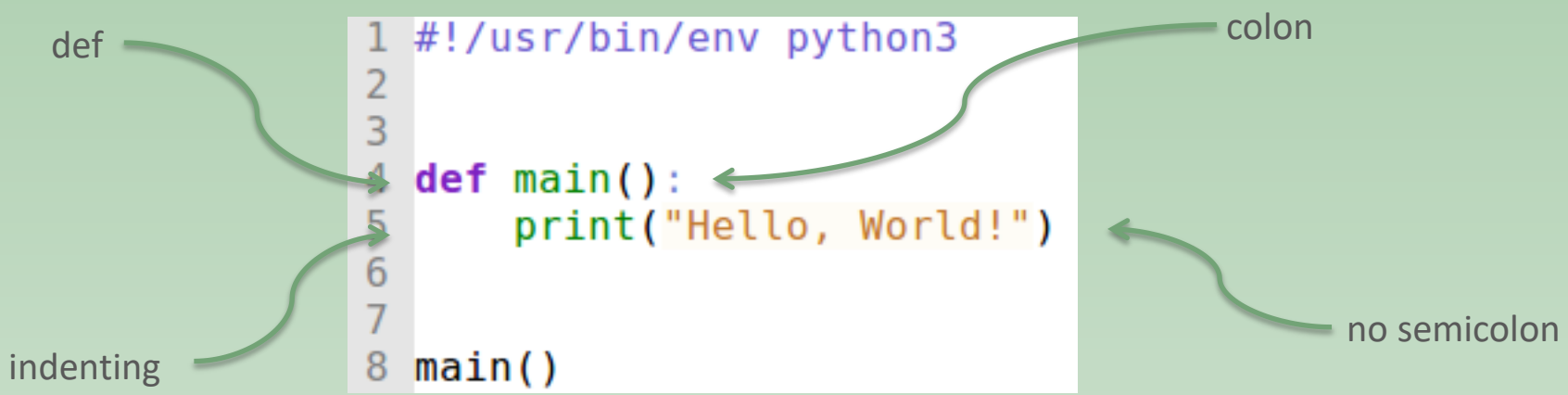
In Python 2, if you use accented characters, then you **must** add this line.
In Python 3, under Linux, this encoding is the default, thus this line can be omitted.

Using special characters:

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

```
>>> 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
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: https://bit.ly/3R0PN7G

```
>>> 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][5]…

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

Tips and tricks

Some useful aliases under Linux:

```
# ~/.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