

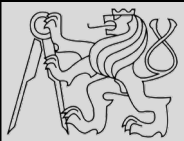
Lecture 1 – Introduction, Variables, Expressions, Statements

<https://cw.fel.cvut.cz/wiki/courses/be5b33prg/start>

Michal Reinštein

Czech Technical University in Prague,
Faculty of Electrical Engineering, Dept. of Cybernetics,
Center for Machine Perception

<http://cmp.felk.cvut.cz/~reinsmic/>
reinstein.michal@fel.cvut.cz



LECTURES – Michal Reinstein, Ph.D.

reinstein.michal@fel.cvut.cz

<http://cmp.felk.cvut.cz/~reinsmic>

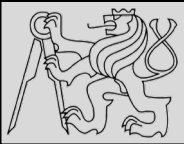


LABS – Ahmet Iscen, Ph.D.

ahmet.iscen@cmp.felk.cvut.cz

<http://cmp.felk.cvut.cz/~iscenahm/>





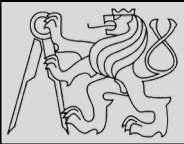
THE GOAL!



m p

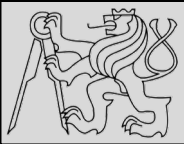
3

- Develop skills with Python **fundamentals**
- Learn to **recognize and write** "good" Python
- Gain experience with **practical Python** tasks
- Understand **when** to choose Python (**or not!**)

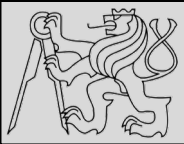


Think like a computer scientist

- Combines:
 - mathematics (**formal language to denote ideas**)
 - engineering (**analysis, synthesis, systems, tradeoffs**)
 - natural science (**observe, hypothesis, test predictions**)
- Problem solving!
 - formulate problems
 - think about solutions
 - implement solutions clearly & accurately



- Problem formulation (**input / output**)
- Formalism (**math?**)
- Algorithm (**the idea!**)
- Implementation (**engineering**)
- Testing (**are we good?**)



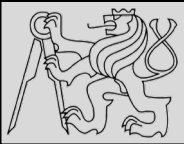
Problem formulation

Find a pair of numbers from a given list of N integers (both **sorted** and **unsorted**) such that their sum is exactly as given (in our case 8).

Examples

[1, 2, 3, 9] where $SUM = 8$... negative case

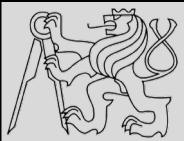
[1, 2, 4, 4] where $SUM = 8$... positive case



1. Solution for sorted list: **quadratic** complexity using **exhaustive search**
2. Solution for sorted list: **$n \cdot \log(n)$** complexity using unidirectional **binary search** (halving the interval) for the complement
3. Solution for sorted list: **linear** complexity using **comparing lower and upper bound** such that if $< \text{SUM}$ increase lower and if $> \text{SUM}$ decrease upper index (*smallest possible sum first two, largest possible sum last two*)
4. Solution for unsorted list: build list of previously visited complements and compare for a match while iterating
5. Final touch – edge cases, empty list



date	week	lect.	topic
05.10.2018	1.	MR	Introduction, the way of program. Variables, expressions, and statements. to be specified
12.10.2018	2.	MR	Program flow, conditionals, simple loops, simple data types to be specified
19.10.2018	3.	MR	Program structure, functions to be specified
26.10.2018	4.	MR	Compound data types, traversals to be specified
02.11.2018	5.	MR	Modules, namespaces to be specified
09.11.2018	6.	MR	Collections: sets, dictionaries, named tuples to be specified
16.11.2018	7.	MR	Files, I/O reading, writing data. to be specified
23.11.2018	8.	MR	MID-TERM TEST during the lecture! Clean code and how to write it PDF Clean code and how to write it PEP8
30.11.2018	9.	MR	Debugging - practical examples to be specified
07.12.2018	10.	MR	Objects, classes I to be specified
14.12.2018	11.	MR	Objects, classes II to be specified
21.12.2018	12.	MR	Iterators, generators to be specified Itertools by example
04.01.2019	13.	MR	END-OF-TERM TEST during the lecture! Python by example 1000 python questions
11.01.2019	14.	MR	Testing programs. Unit tests. Exceptions. to be specified

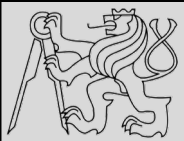


Grading

Points: 50 homework (mostly coding), 20 mid-term tests during the term, 30 final exam. *At least 30 points (out of 70) are needed before going to the final exam (in order to obtain "zapocet"). At least 10 points (out of 30) are needed to pass the exam.* Extra points for discussions and bonus homework count to the total sum of 70 points.

A	B	C	D	E	F
100-91	90-81	80-71	70-61	60-51	50-0

- Lectures and computer labs
- Home works
- Programming tests (2x) during the lectures
- Final exam test
- Extra points: *activity, finding bugs, errors ...*
- Automatic evaluation & plagiarism detection



PLAGIARISM WARNING

https://cw.fel.cvut.cz/wiki/help/common/plagiarism_cheating

Rules

These are the general rules for most courses using this CourseWare and UploadSystem. If you have any doubts, ask a lecturer.

Table of Contents
-Rules
-What is a plagiarism
-Practical notes

1. Code or text (from hereafter, only code) for each assignment has to be done independently. In programming, it is normal to (re)use someone else's code, however, for study purposes this is undesirable. ¹⁾ If you are to use someone else's code, it will be specifically allowed in the assignment.
2. Turned in work are checked for plagiarism. Any code, which is either a complete copy or which after simple mechanical syntax modification is identical to code of someone else, is a plagiarism.
3. In case you turn in a plagiarism, you still have to turn in a original solution in original time, however, you receive zero points.
4. In case you turn in copied code repeatedly²⁾, you are automatically classified with an F.
5. Lecturer does not guarantee he will be able to differentiate between original work and plagiarism.
6. Student who does not agree with his evaluation or classification can proceed according to [ČVUT Study and Examination process](#).

What is a plagiarism

[Plagiarism](#) is a text, code, illustration, method or idea taken from another's work without citing your sources.

Everyone who uses someone work or results of another, has to indicate he did so and reference the original author. Otherwise it is plagiarism. For study reasons, using other's work is also forbidden in some courses, even if it was properly labeled.

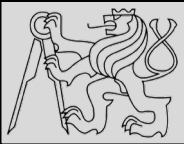
Practical notes

In most assignments, you will primarily have to avoid using someone else's code. If you find a code (i.e. from a colleague or on the internet), which solves your assignment, try to wait for a day or two, before starting your own work, to limit the volume of copied work.

Students, we strongly appeal to your professional honor. Copying and Cheating are serious offences against academic ethics. If you are at your wit's end, ask lecturers for help. They will gladly provide you with the directions to the solution.



¹⁾ If you study someone else's code, try to wait for at least 24 hours, before you start writing your own implementation.

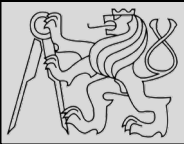
²⁾ This means even across your courses, not only in the same course



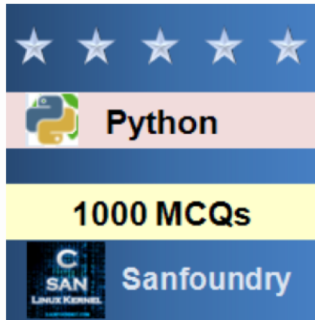
Exams

Multiple choice test, **no materials** as well as **no devices** allowed (*papers will be provided, only own pen is necessary*). Any use of materials, devices or cooperation during the exam will be awarded with 0 points (fail). The content of the final exam will be based on the content of:

1. Lectures (not limited but including the slides released after each lecture)
2. Exercises during the labs
3. Relevant chapters of the  [Wentworth2012](#) book (links to relevant chapters can be found at the bottom of lecture slides)
4. **Collection of Python multiple-choice question to practice for the exam**
 <http://www.sanfoundry.com/1000-python-questions-answers/>



Python Questions and Answers



Our 1000+ Python questions and answers focuses on all areas of Python subject covering 100+ topics in Python. These topics are chosen from a collection of most authoritative and best reference books on Python. One should spend 1 hour daily for 2-3 months to learn and assimilate Python comprehensively. This way of systematic learning will prepare anyone easily towards Python interviews, online tests, examinations and certifications.

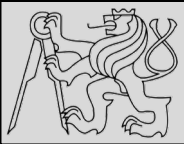
Highlights

- 1000+ Multiple Choice Questions & Answers in Python with explanations
- Every MCQ set focuses on a specific topic in Python Subject

Who should Practice these Python Questions?

- Anyone wishing to sharpen their knowledge of Python Subject
- Anyone preparing for aptitude test in Python
- Anyone preparing for interviews (campus/off-campus interviews, walk-in interview and company interviews)
- Anyone preparing for entrance examinations and other competitive examinations
- All – Experienced, Freshers and Students

SOURCE: <https://www.sanfoundry.com/1000-python-questions-answers/>



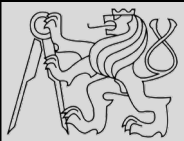
According to <https://www.techrepublic.com> ...

1. **Ease of learning** - one of the easiest programming languages to learn, known for high reliability and simple syntax (rapid prototyping, steep learning curve)

2. **The explosion of AI, machine learning, and data science in the enterprise**

(<https://www.tensorflow.org> , <https://www.scipy.org> , <http://scikit-learn.org/stable/> , <http://playground.arduino.cc/Interfacing/Python> , ...)

3. **Large developer community** - available for many operating systems, often used to command other programs



WHY PYTHON?



Sep 2018	Sep 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.436%	+4.75%
2	2		C	15.447%	+8.06%
3	5	▲	Python	7.653%	+4.67%
4	3	▼	C++	7.394%	+1.83%
5	8	▲	Visual Basic .NET	5.308%	+3.33%
6	4	▼	C#	3.295%	-1.48%
7	6	▼	PHP	2.775%	+0.57%
8	7	▼	JavaScript	2.131%	+0.11%
9	-	▲▲	SQL	2.062%	+2.06%
10	18	▲▲	Objective-C	1.509%	+0.00%
11	12	▲	Delphi/Object Pascal	1.292%	-0.49%
12	10	▼	Ruby	1.291%	-0.64%
13	16	▲	MATLAB	1.276%	-0.35%

September 2018: Python enters the TIOBE index top 3 for the first time

<https://www.tiobe.com/tiobe-index/>

<https://www.tiobe.com/tiobe-index/programming-languages-definition/>

source: <https://www.tiobe.com/tiobe-index/>

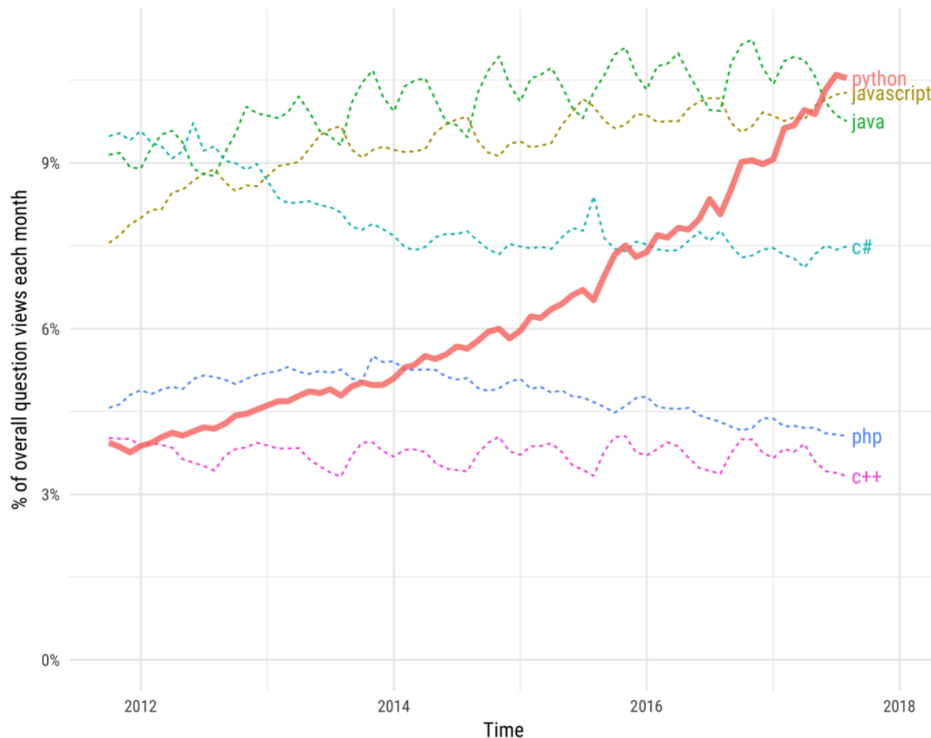


WHY PYTHON?



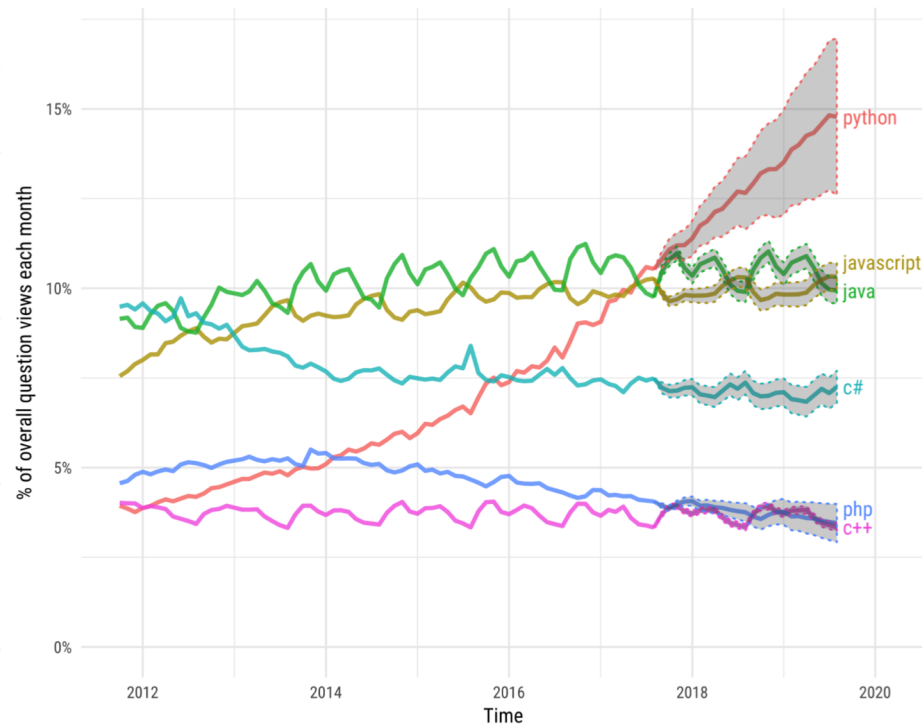
Growth of major programming languages

Based on Stack Overflow question views in World Bank high-income countries



Projections of future traffic for major programming languages

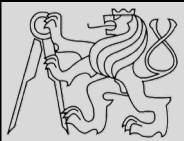
Future traffic is predicted with an STL model, along with an 80% prediction interval.



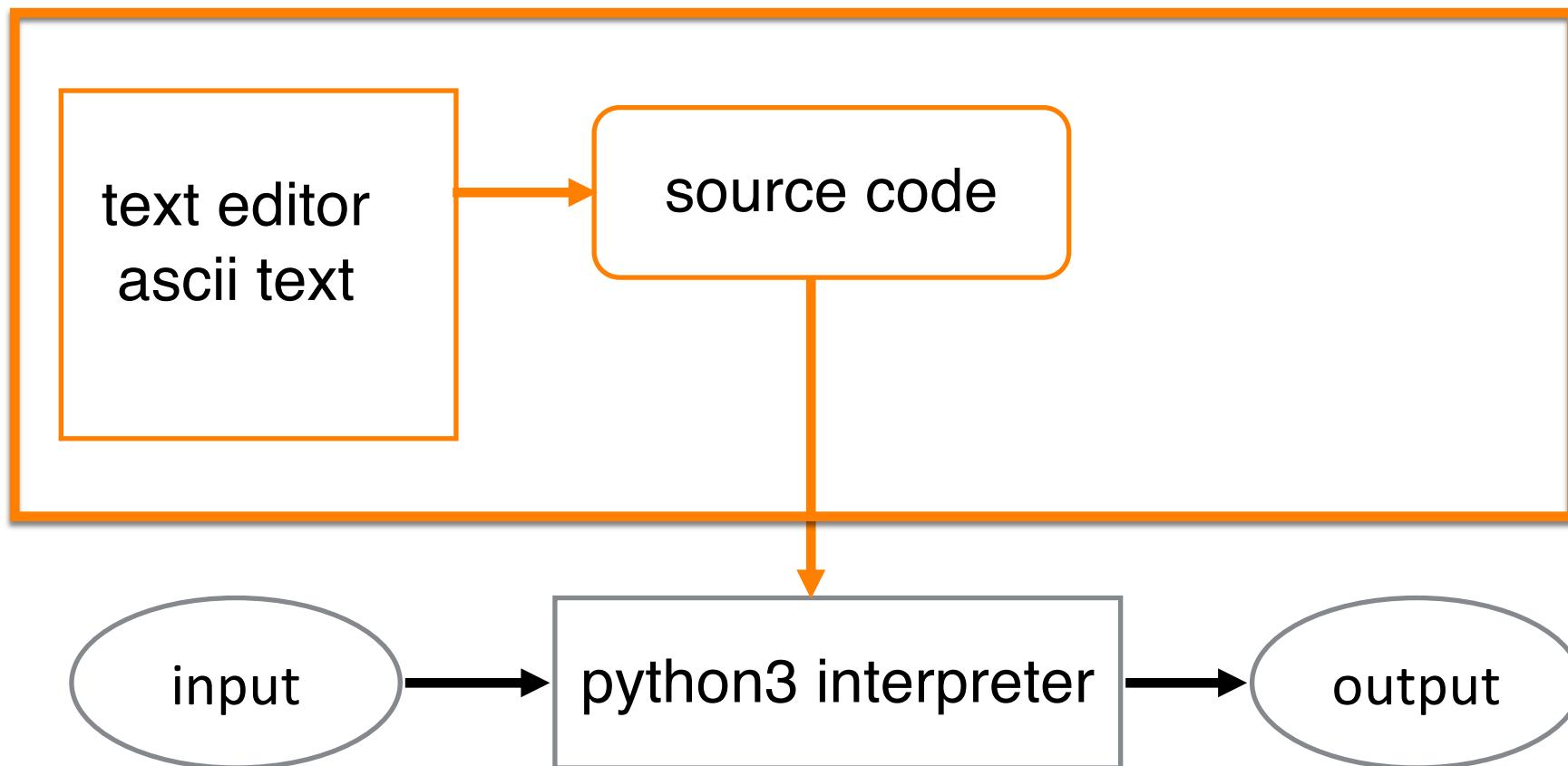
Stack Overflow <https://stackoverflow.com/> a good friend of yours!

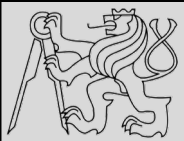
<https://stackoverflow.blog/2017/09/06/incredible-growth-python/>

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



- **Program** is a sequence of instructions that specifies how to perform a computation.
- **Input** - get data from the keyboard, a file, device ..
- **Output** - display data on the screen or send data to a file or other device (client/server, local/remote).
- **Math** - perform mathematical operations (**algorithms**)
- **Conditional execution** - Check for certain conditions and execute the appropriate sequence of statements.
- **Repetition** - Perform some action repeatedly

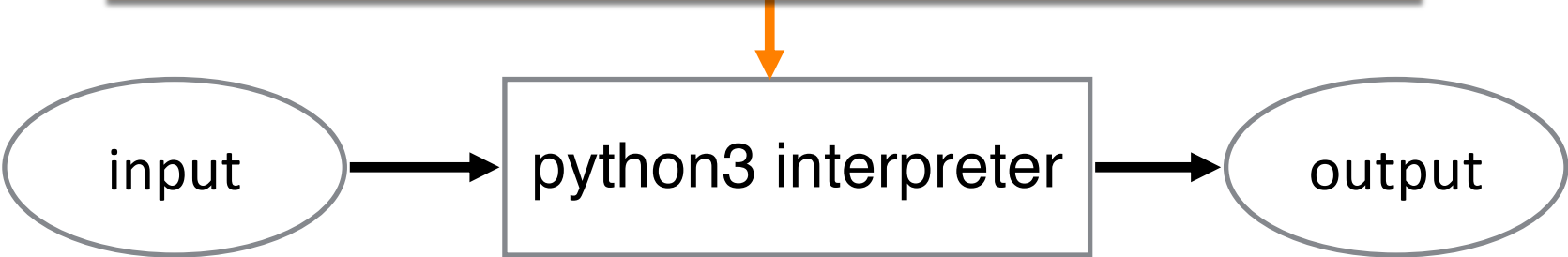


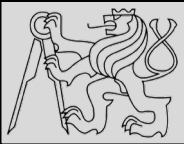


HELLO, WORLD!



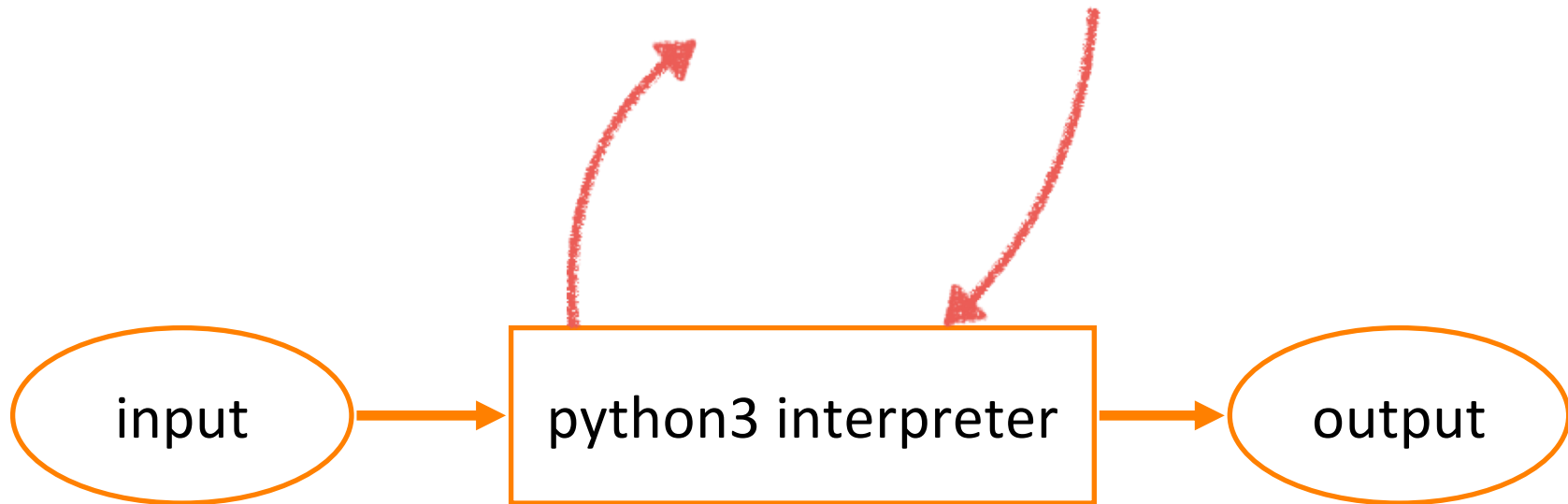
```
[michalreinstein@MacBook-Pro:~$~ $ python3
Python 3.6.2 (default, Sep 21 2017, 00:54:38)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> print("Hello, World!")
Hello, World!
>>>
```

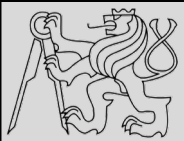




Entering commands – in two modes:

1. Immediate mode using python console
2. Script mode using IDE or text editor





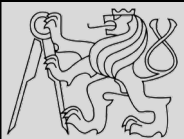
```

michalreinstein@MacBook-Pro:~$~ $ python3
Python 3.6.2 (default, Sep 21 2017, 00:54:38)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

https://artifex.org/~hblanks/talks/2011/pep20_by_example.html

source: http://artifex.org/~hblanks/talks/2011/pep20_by_example.html



WHAT IS PYTHON?



m p

21



Integrated Development Environment, IDE

Python program, code,
expressions

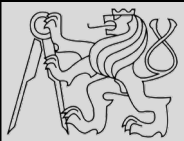
Python interpreter

Operating System
MS Win, Mac OSX, Linux

computer - hw



original slide by Tomas Svoboda, BE5B33PRG 2016/2017



Syntax errors

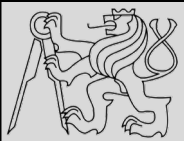
- Formal tokens & structure of the code must obey rules (IDE)
- Python executes only syntactically correct code

Runtime errors

- Discovered during runtime (program fails!)
- Exceptions – something exceptional happens (we can catch and handle exceptions!)

Semantic errors

- The meaning of the program (semantics) is wrong
- Program runs but does something different than we want



```
>>> type("Hello, World!")  
<class 'str'>  
>>> type(17)  
<class 'int'>
```

```
>>> type(3.2)  
<class 'float'>
```

```
>>> type("17")  
<class 'str'>  
>>> type("3.2")  
<class 'str'>
```

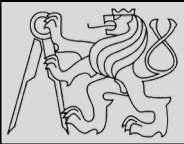
Strings in Python can be enclosed in either single quotes (') or double quotes ("), or three of each (''' or """)

```
>>> type('This is a string.')
```

```
<class 'str'>  
>>> type("And so is this.")  
<class 'str'>  
>>> type("""and this.""")  
<class 'str'>  
>>> type('''and even this...''')
```

```
<class 'str'>
```

- Integers (int) 1, 10, 124
- Strings (str) "Hello, World!"
- Float (float) 1.0, 9.999



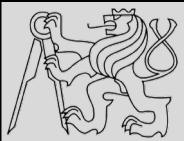
The **assignment statement** gives a value to a variable:

```
>>> message = "What's up, Doc?"
>>> n = 17
>>> pi = 3.14159

>>> message
'What's up, Doc?'
>>> n
17
>>> pi
3.14159
```

```
>>> day = "Thursday"
>>> day
'Thursday'
>>> day = "Friday"
>>> day
'Friday'
>>> day = 21
>>> day
21
```

- We use variables to **remember** things!
- Do not confuse **=** and **==** !
 - = is **assignment** token such that *name_of_variable = value*
 - == is operator to **test equality**
- Key property of a variable that we can change its value
- Naming convention: **with freedom comes responsibility!**
- Illegal name causes a **syntax error** (begin with letter or _)



cannot begin with a number



```
>>> 76trombones = "big parade"  
SyntaxError: invalid syntax
```

this \$ is illegal character



```
>>> more$ = 1000000
```

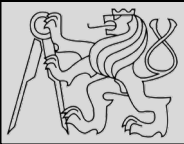
```
SyntaxError: invalid syntax
```

class is reserved keyword



```
>>> class = "Computer Science 101"  
SyntaxError: invalid syntax
```

- We use variables to **remember** things!
- Do not confuse = and == !
 - = is **assignment** token such that *name_of_variable = value*
 - == is operator to **test equality**
- Key property of a variable that we can change its value
- Naming convention: **with freedom comes responsibility!**
- Illegal name causes a **syntax error** (begin with letter or _)



and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with
yield	True	False	None		

- Python keywords have **special** purpose
- Always choose names **meaningful** to human readers
- Use **comments** to improve readability and clarity

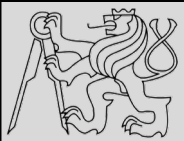
source http://openbookproject.net/thinkcs/python/english3e/variables_expressions_statements.html



```
1 #-----  
2 # This demo program shows off how elegant Python is!  
3 # Written by Joe Soap, December 2010.  
4 # Anyone may freely copy or modify this program.  
5 #-----  
6  
7 print("Hello, World!")      # Isn't this easy!
```

- Big & complex programs == difficult to read
- Comments and blank lines are for human readers only, ignored by the interpreter
- Use this token **#** to start a comment
- Use **blank lines** to make the code visually more appealing

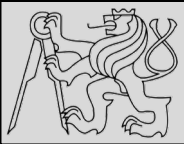
source http://openbookproject.net/thinkcs/python/english3e/variables_expressions_statements.html



```
Python Console
/opt/local/bin/python3.6 /Applications/PyCharm.app/Contents/helpers
Python 3.6.2 (default, Sep 21 2017, 00:54:38)
In[2]: students = ['Anna', 'Bob', 'David', 'Mark', 'Brandon']
.....:
.....: for student in students:
.....:     if len(student) >= 5:
.....:         print(student)
.....:
? David
Brandon
In[3]:
```

- Statement is an **instruction** executable in Python
- Statements **do not produce any results**
- So far only assignment statements =
- Statement examples: *for, in, if ...*

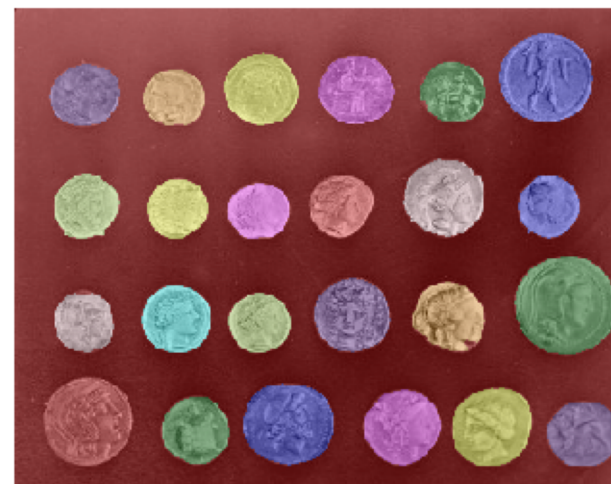
source http://openbookproject.net/thinkcs/python/english3e/variables_expressions_statements.html



```
Python Console
/opt/local/bin/python3.6 /Applications/PyCharm.app/Contents/helpers
Python 3.6.2 (default, Sep 21 2017, 00:54:38)
In[2]: students = ['Anna', 'Bob', 'David', 'Mark', 'Brandon']
.....:
.....: for student in students:
.....:     if len(student) >= 5:
.....:         print(student)
.....:
? David
Brandon
In[3]:
```

- Expression is a combination of **values**, **variables**, **operators**, and **calls** to functions
- Built-in Python functions: *len*, *type*, *print*
- Value by itself is an expression
- Expression **produces result** (right side of an assignment)

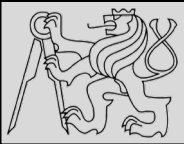
source http://openbookproject.net/thinkcs/python/english3e/variables_expressions_statements.html



IPYTHON – running python interpreter from web browser

http://scikit-image.org/docs/dev/auto_examples/xx_applications/plot_coins_segmentation.html

[Google Colab version](#)



- <https://cw.fel.cvut.cz/wiki/courses/be5b33prg/start>
- <http://openbookproject.net/thinkcs/python/english3e/>
- https://cw.fel.cvut.cz/wiki/courses/be5b33prg/tutorials/python#watching_and_listening
- <https://stackoverflow.blog/2017/09/06/incredible-growth-python/>
- <http://stanfordpython.com/>
- <https://www.sanfoundry.com/1000-python-questions-answers/>
- https://artifex.org/~hblanks/talks/2011/pep20_by_example.html