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

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

LABS – Tomas Petricek

- petrito1@fel.cvut.cz
- http://cmp.felk.cvut.cz/~petrito1

THE GOAL!

- 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)**,
  - engineering **(analysis, synthesis, systems)**,
  - science **(observe, hypothesis, predictions)**

- Problem solving!
  - Formulate problems
  - Think about solutions
  - Implement solutions clearly and accurately

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

EXAMPLE – THE PROBLEM!

Problem formulation

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

Examples

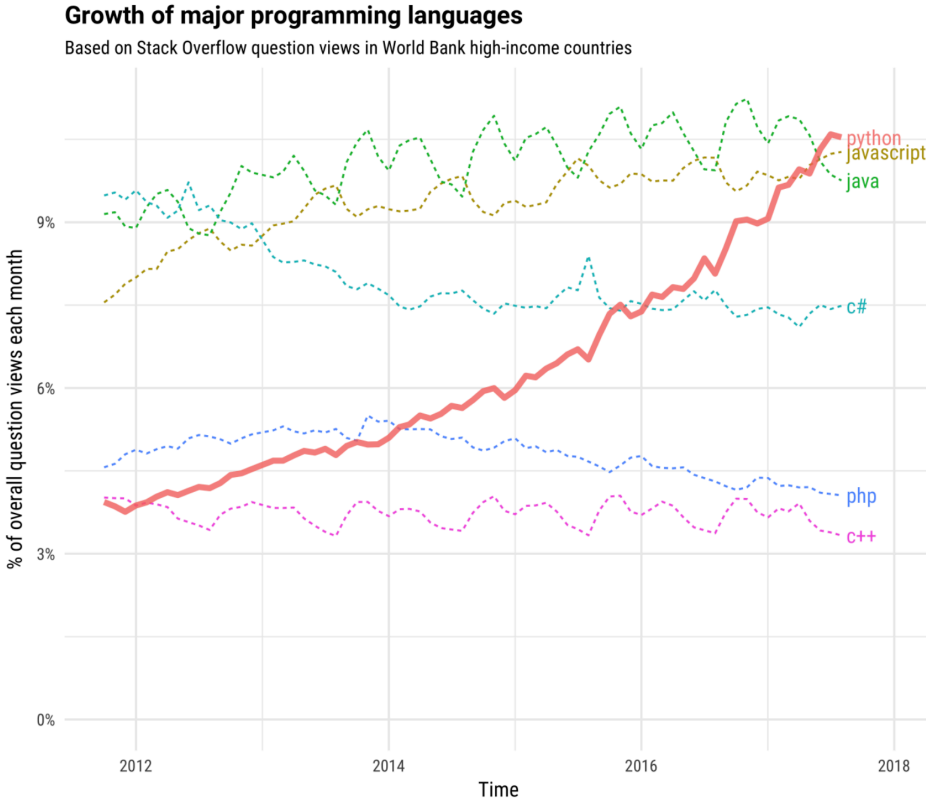
[1, 2, 3, 9] where SUM = 8 … negative case
[1, 2, 4, 4] where SUM = 8 … positive case

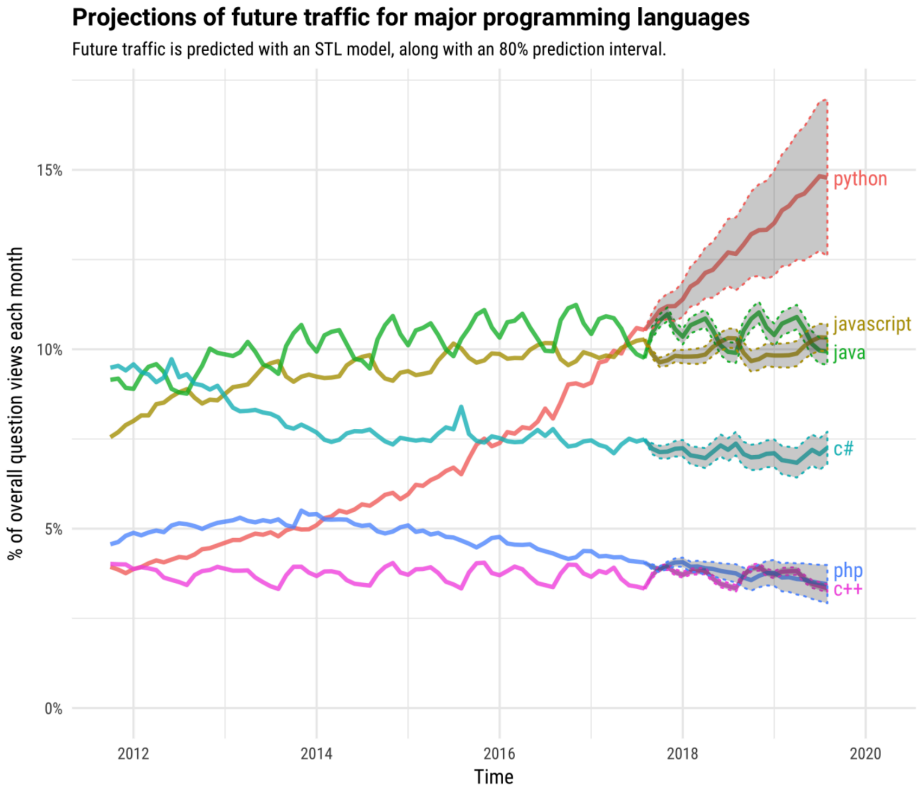
source: https://www.youtube.com/watch?v=XKu_SEDAykw

WHY PYTHON?

- Handy for engineers (**rapid prototyping**)
- Easy for beginners (**steep learning curve**)
- But strong for big apps: big data, AI …
  (https://www.tensorflow.org , https://www.scipy.org , http://scikit-learn.org/stable /, http://playground.arduino.cc/Interfacing/Python , …)
- Often used to command other programs
  (https://www.blender.org/manual/editors/python_console.html)
- Available for many platforms/operating systems (**large community**)

WHY PYTHON?





Stack Overflow https://stackoverflow.com/

a good friend of yours!

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

```
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!
>>>
```

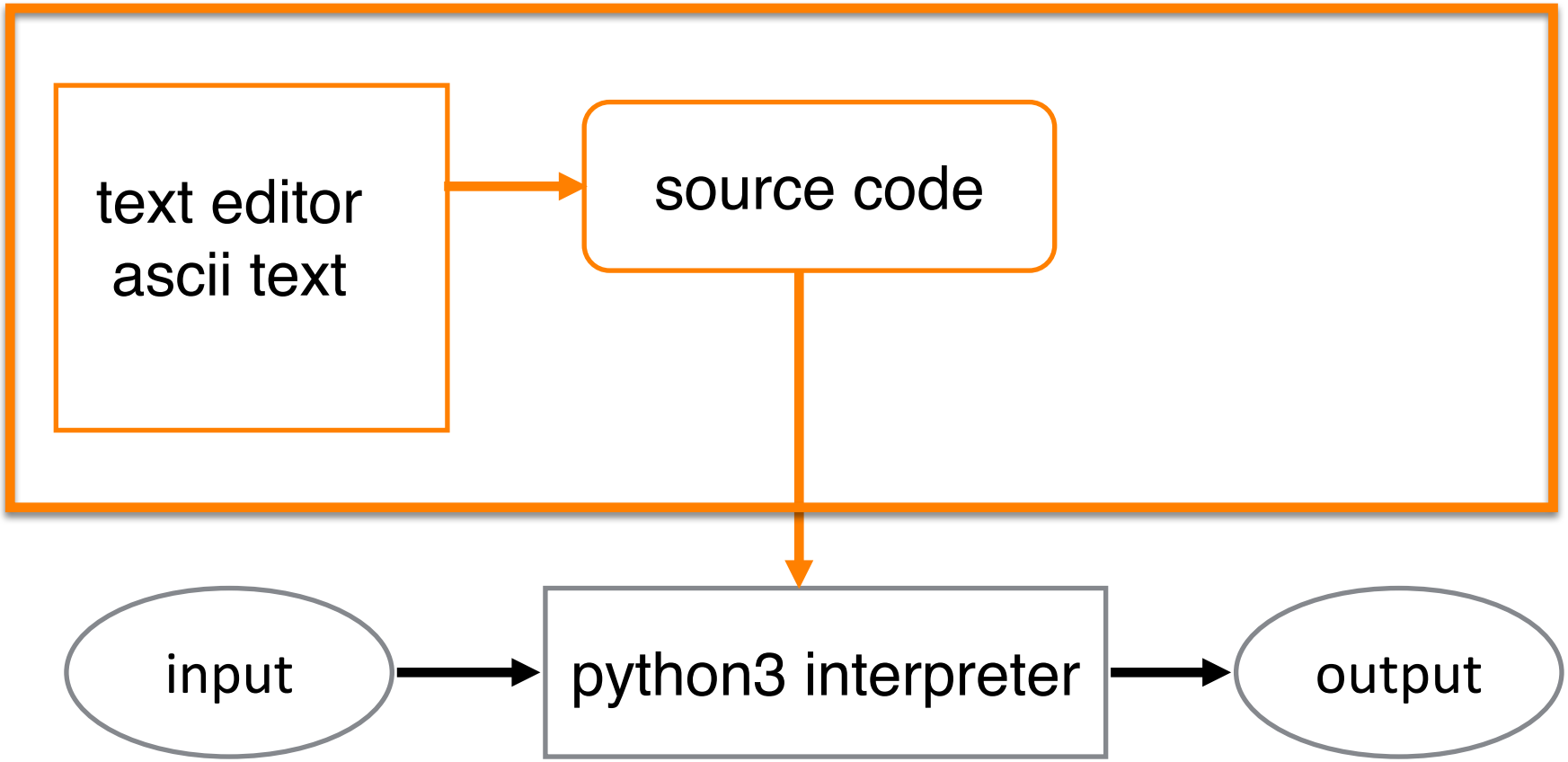
We will come back to the Zen of Python later …

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

THE PROGRAM

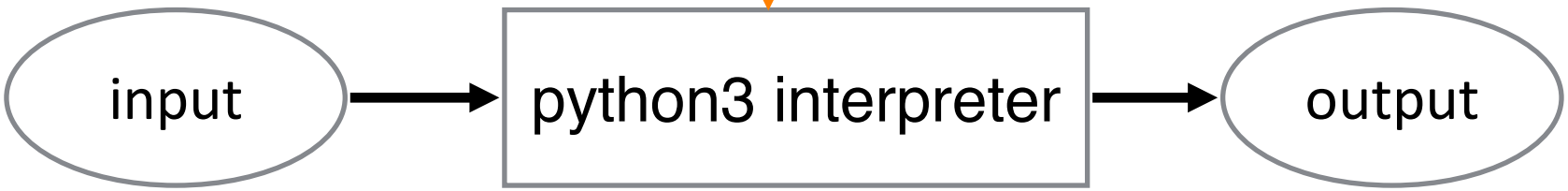
- **Program** is a sequence of instructions that specifies how to perform a computation.
- **Input** - get data from the keyboard, a file, device ..
- **Output** - display data on the screen or send data to a file or other device (client/server, local/remote).
- **Math** - perform mathematical operations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

OUR PROGRAM


text editor
ascii text
source code
input
python3 interpreter
output

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!
 >>>
```



WHAT IS PYTHON?

Integrated Development Environment, IDE

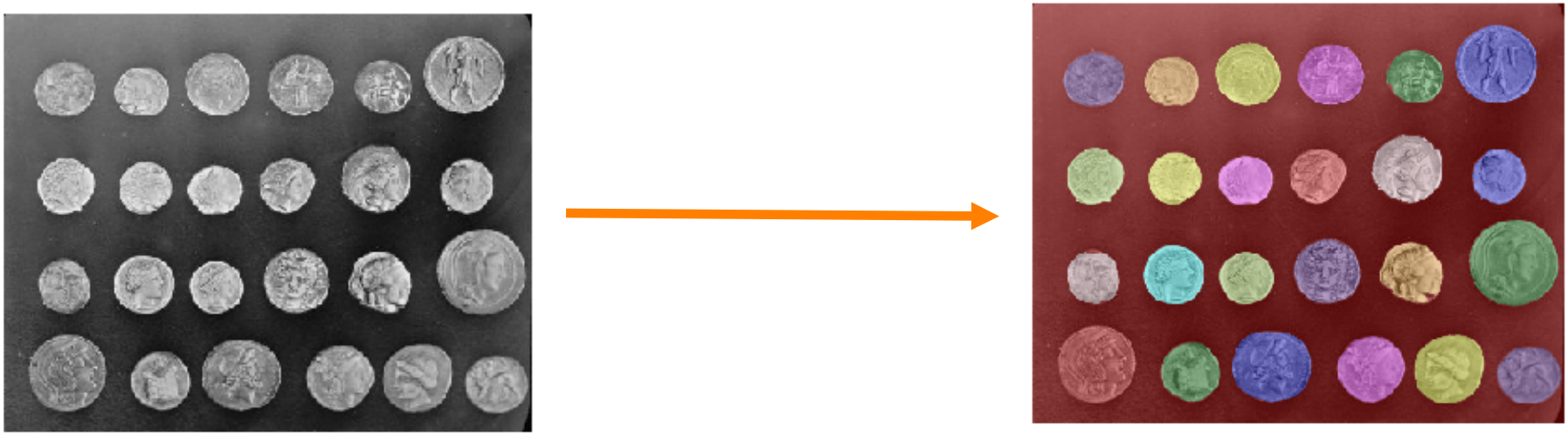
Python program, code, expressions

Python interpreter

Operating System
MS Win, Mac OSX, Linux

computer - hw

original slide by Tomas Svoboda, BE5B33PRG 2016/2017



IPYTHON – running python interpreter from web browser

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

Syntax errors

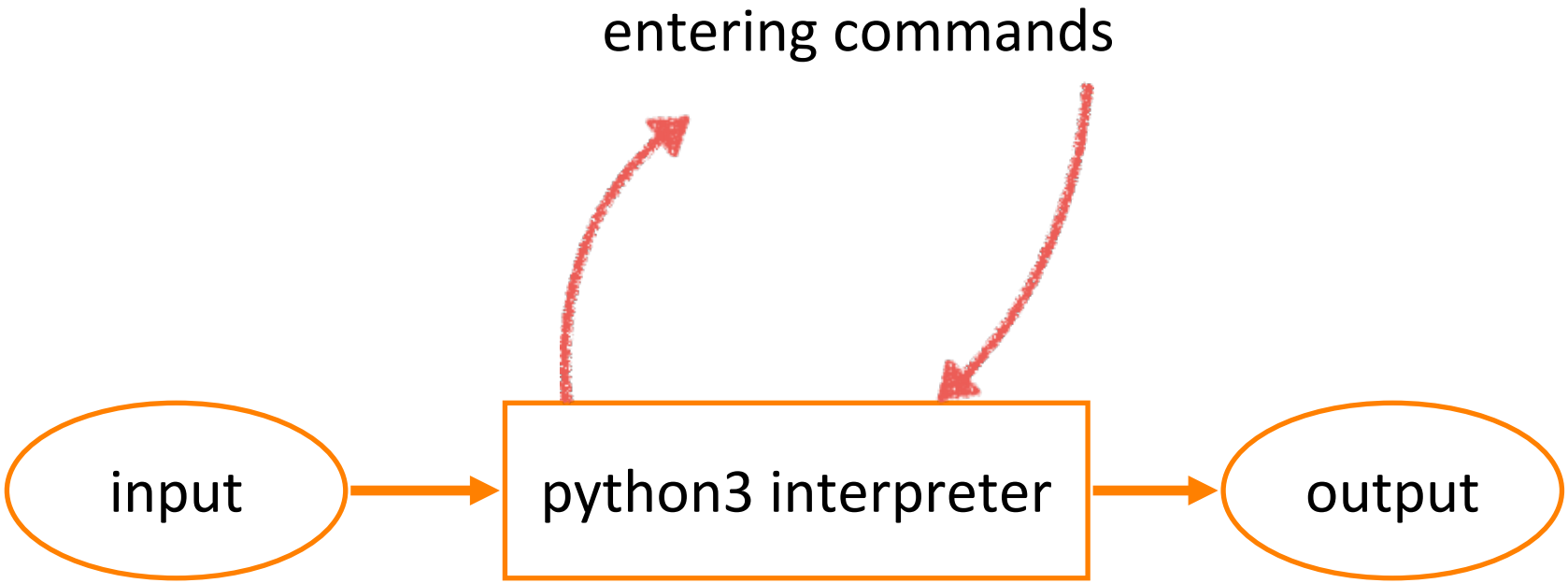
- Formal tokens & structure of the code must obey rules (IDE)
- Python executes only syntactically correct code

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



And now is the right time to actually explore the interpreter …

DATA TYPES

```
>>> 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

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

VARIABLES

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**

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

VARIABLES

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**

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

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

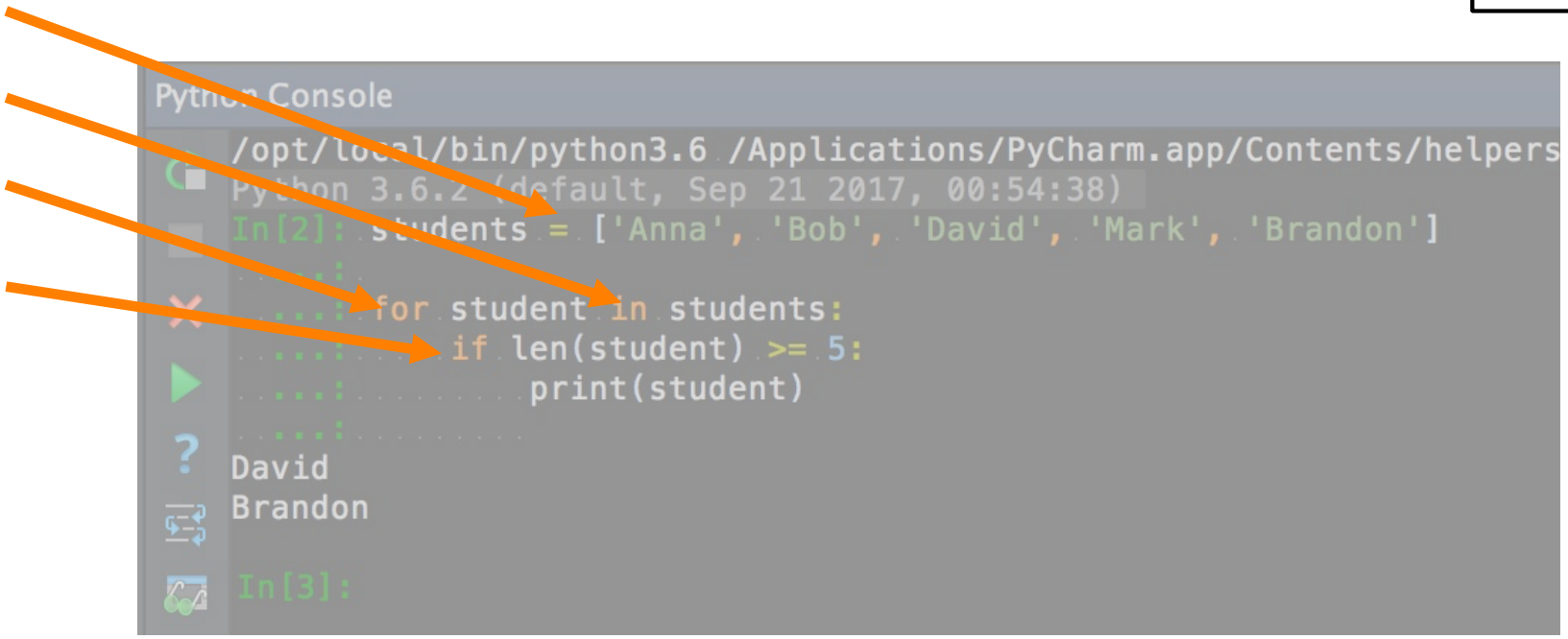
source http://openbookproject.net/thinkcs/python/english3e/variables_expressions_statements.html

COMMENTS

```
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

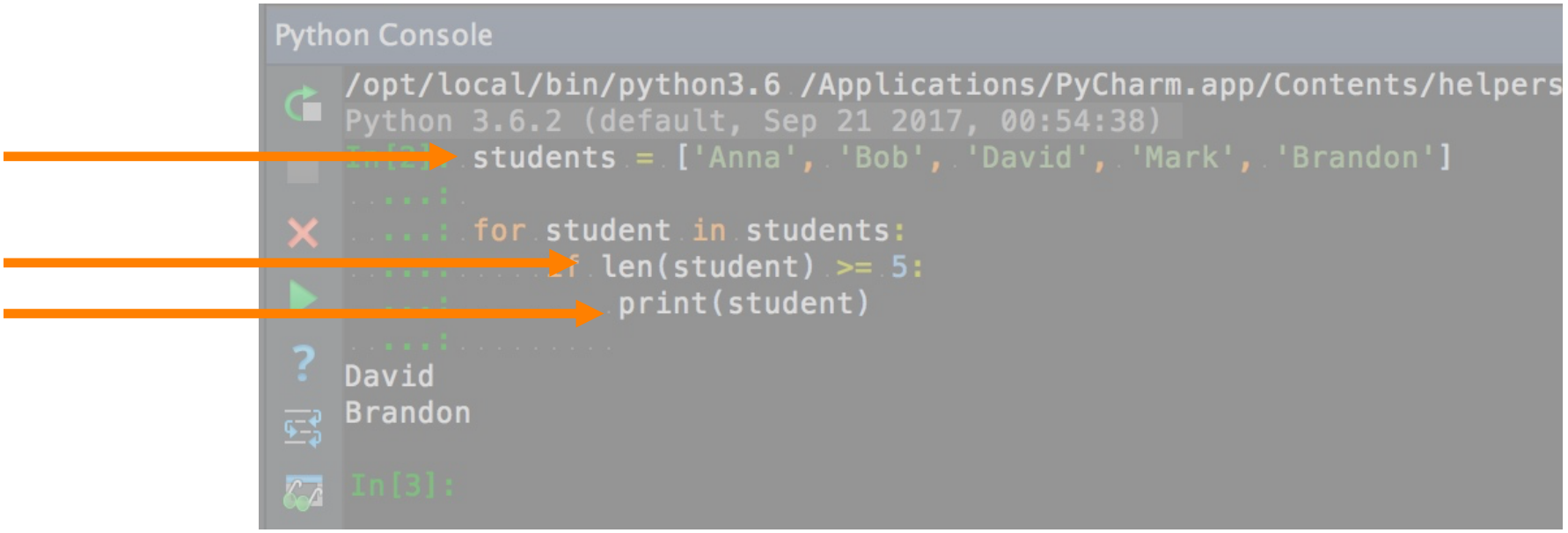
source http://openbookproject.net/thinkcs/python/english3e/variables_expressions_statements.html



- 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

EXPRESSIONS



- 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

date	week	lect.	topic
06.10.2017	1.	MR	Introduction, the way of program. Variables, expressions, and statements. lec01-intro.pdf
13.10.2017	2.	MR	Program flow, conditionals, simple loops, simple data types, lec02-programflow.pdf
20.10.2017	3.	MR	Program structure, functions, lec03-functions.pdf
27.10.2017	4.	MR	More complex data types, traversals, lec04-compound-types.pdf
03.11.2017	5.	MR	Iterations, loops. lec05-iteration.pdf (examples in PDF are for self-study). puzzle05.pdf
10.11.2017	6.	MR	Modules. Testing programs. lec06-modules.pdf, lec06-testing.pdf, Bonus task: Balls
17.11.2017	7.		**No lecture, Public Holiday**
24.11.2017	8.	MR	Clean code, how to write it
01.12.2017	9.	MR	I/O reading, writing data. Intro to SPAM filter (Handouts).
08.12.2017	10.	MR	Data collections: `set, dict, , ...` lec09-othercollections.pdf
15.12.2017	11.	MR	Data collections II `namedtuple`. Objects, classes I, lec10-classes-and-objects.pdf
22.12.2017	12.	MR	Objects, classes II, lec11-classes-and-objects-ii.pdf
05.01.2018	13.	MR	Handling exceptions, making code more robust. lec12-classes-and-objects-ii.pdf. Questions and answers lec13-questions-problems.pdf, score card and my_time codes
12.01.2017	14.	MR	Selected chapters from advanced programming. Questions and aswers.

- Lectures & computer labs
- Home works **50%**
- Programming tests during the term **20%**
- Final exam - programming **30%**
- Extra points: activity, finding bugs, errors … **10%**
- Automatic evaluation & plagiarism detection

original slide by Tomas Svoboda, BE5B33PRG 2016/2017

- http://openbookproject.net/thinkcs/python/english3e/
- https://cw.fel.cvut.cz/wiki/courses/be5b33prg/start
- https://stackoverflow.blog/2017/09/06/incredible-growth-python/
- http://stanfordpython.com/ CS41 course