Lecture 1 – Introduction. Variables, expressions

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

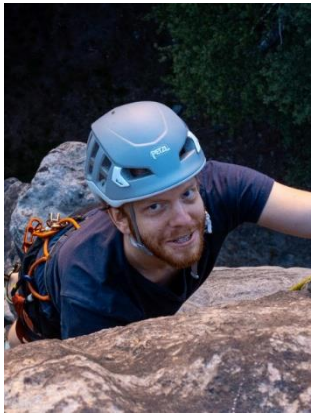
Tomas Jenicek

Czech Technical University in Prague,
Faculty of Electrical Engineering, Dept. of Cybernetics,
Center for Machine Perception
http://cmp.felk.cvut.cz/~jenicto2/
tomas.jenicek@fel.cvut.cz

LECTURES – Tomas Jenicek

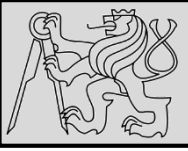
tomas.jenicek@fel.cvut.cz

http://cmp.felk.cvut.cz/~jenicto2



LABS – Milan Nemy

nemymila@fel.cvut.cz



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 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 SOLVING!

- 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 (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

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

EXAMPLE – THE PROBLEM!

1. Solution for sorted list: **quadratic** complexity using **exhaustive search**
2. Solution for sorted list: **n*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 < SUM increase lower and if > 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 (*hash table with constant time for look-up*)
5. Final touch – edge cases, empty list

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

COURSE ADMINISTRATION

week	date	topic	materials
1.	24.09.2021	Introduction. Variables, expressions.	
2.	01.10.2021	Primitive data types, program flow	
3.	08.10.2021	Program structure, functions	
4.	15.10.2021	Sequence data types, traversals	
5.	22.10.2021	Collections (sets, dictionaries), iterators	
6.	29.10.2021	Modules, namespaces, conventions	
7.	05.11.2021	Mid-term test	
8.	12.11.2021	Filesystem, file reading and writing	
9.	19.11.2021	Debugging, code testing, exceptions	
10.	26.11.2021	Objects, classes I	
11.	03.12.2021	Objects, classes II	
12.	10.12.2021	End-of-term test	
13.	17.12.2021	Advanced concepts	
14.	07.01.2022	Revision for the exam	

Grading

Points: 50 homework (mostly coding), 20 tests during the term (2 tests, 10 points each), 30 final exam.

At least 30 points (out of 70) and regular lab attendance 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 final exam. To pass the course and get a grade, "zapocet" must be obtained, exam passed and at least 51 points gained in total (see the table below). It is possible to get additional up to 20 points for extra activity during the semester, such as completing a bonus homework.

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

F means fail.

- Lectures and computer labs
- Home works
- 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

Plagiarism

It is required that all work you submit in this course is original and your own. It is not allowed to copy homework solutions from other students or from the internet, to provide your homework solutions to other students, or to publish them on the internet. You may freely discuss your solutions with other students, bu code sharing is prohibited. See plagiarism_cheating for more details.

It is your responsibility that you do not share your code. In case of discovery, the person who provided the code is punished as well. Sufficient evidence of plagiarism is even when a student is unable to explain how his code works.

There are very strict punishments with regard to plagiarism and cheating during tests and exams. The first discovered plagiarism/cheating leads to zero points from the assignment/test. In case of an assignment, it is further necessary to submit a new, original, solution for zero points. The second occurrence means an F from the course and any subsequent plagiarism/cheating leads to disciplinary actions at the faculty level. It is important to note that every discovered plagiarism/cheating gets into your record – the plagiarism/cheating occurrences are counted cumulatively across all courses during your studies.

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

Exams and Tests

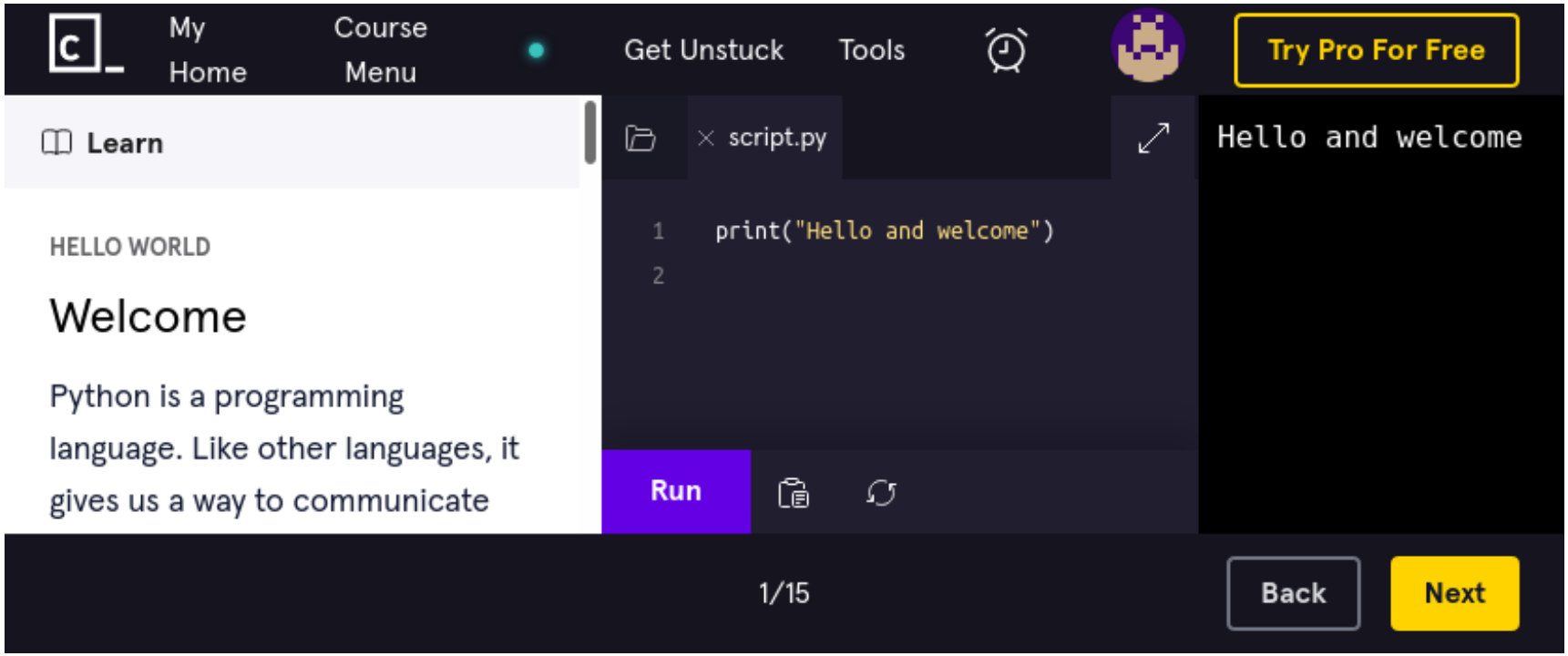
There will be two tests during the semester (mid-term and end-of-term) and a final exam during the exam period. The format of both the exam and the mid-term/end-of-term tests will be specified during the semester.

The content of the exam / test will be based on the content of:

1. Lectures before the date of the exam / test (not limited but including the slides released after each lecture)
2. Exercises and home-works practiced before the date of the exam / test
3. Relevant chapters of the Wentworth2012 book
4. Collection of Python multiple-choice question to practice for the exam http://www.sanfoundry.com/1000-python-questions-answers/ related to the content of the lectures

Form of tests will be specified

Additional Online Resources



https://cw.fel.cvut.cz/wiki/courses/be5b33prg/resources

WHY PYTHON?

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

Source: https://www.techrepublic.com/google-amp/article/why-python-is-so-popular-with-developers-3-reasons-the-language-has-exploded/

WHY PYTHON?

Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		C	11.83%	-4.12%
2	3	↑	Python	11.67%	+1.20%
3	2	↓	Java	11.12%	-2.37%
4	4		C++	7.13%	+0.01%
5	5		C#	5.78%	+1.20%
6	6		Visual Basic	4.62%	+0.50%
7	7		JavaScript	2.55%	+0.01%
8	14	⇈	Assembly language	2.42%	+1.12%
9	8	↓	PHP	1.85%	-0.64%
10	10		SQL	1.80%	+0.04%

September 2020: Python enters the TIOBE index **top 3** for the first time in 2018 and holds top 3 position ever since https://www.tiobe.com/tiobe-index/

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

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

WHY PYTHON?

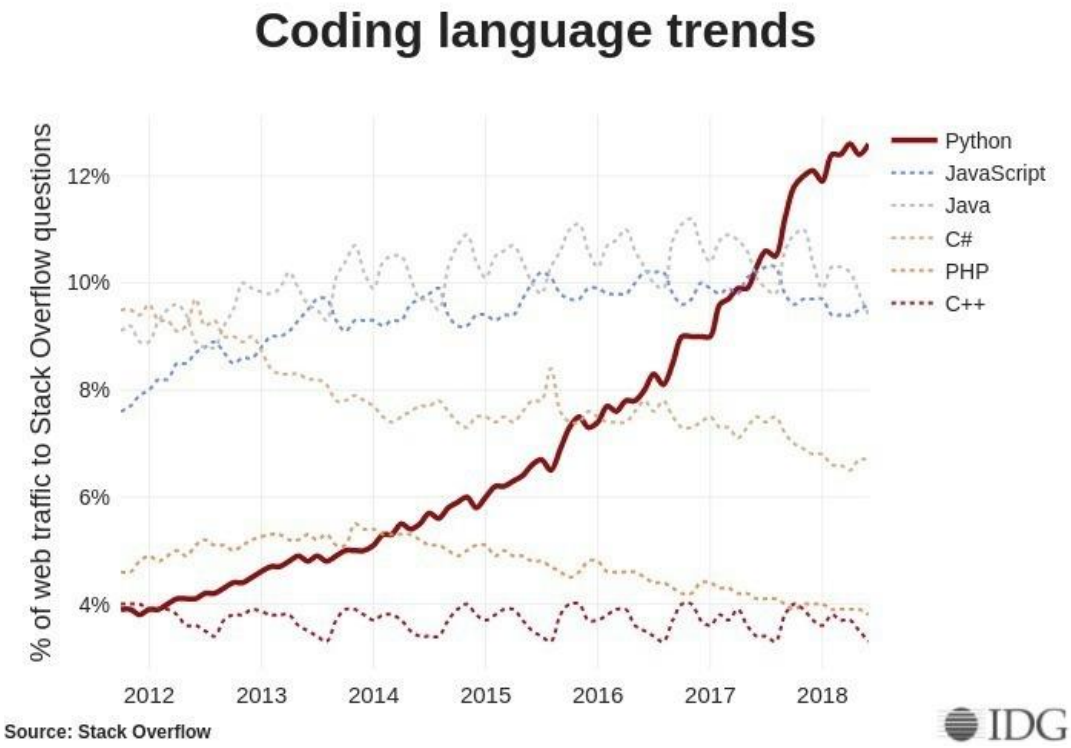
Companies using Python

Companies using python #1: Google: The company that needs no introduction, Google. Its video platform Youtube is all written on python!

Companies using python #2: Instagram: An image sharing platform was a simple language developed on Django (Python framework) before it was acquired by Facebook.

Companies using python #3: Netflix: The video streaming platform offer suggestions to its users constantly. Do you know what makes this possible. Yes, it's THE PYTHON!

Companies using python #4: Facebook: According to the official blog from facebook, 21% of facebook codebase is based on Python.



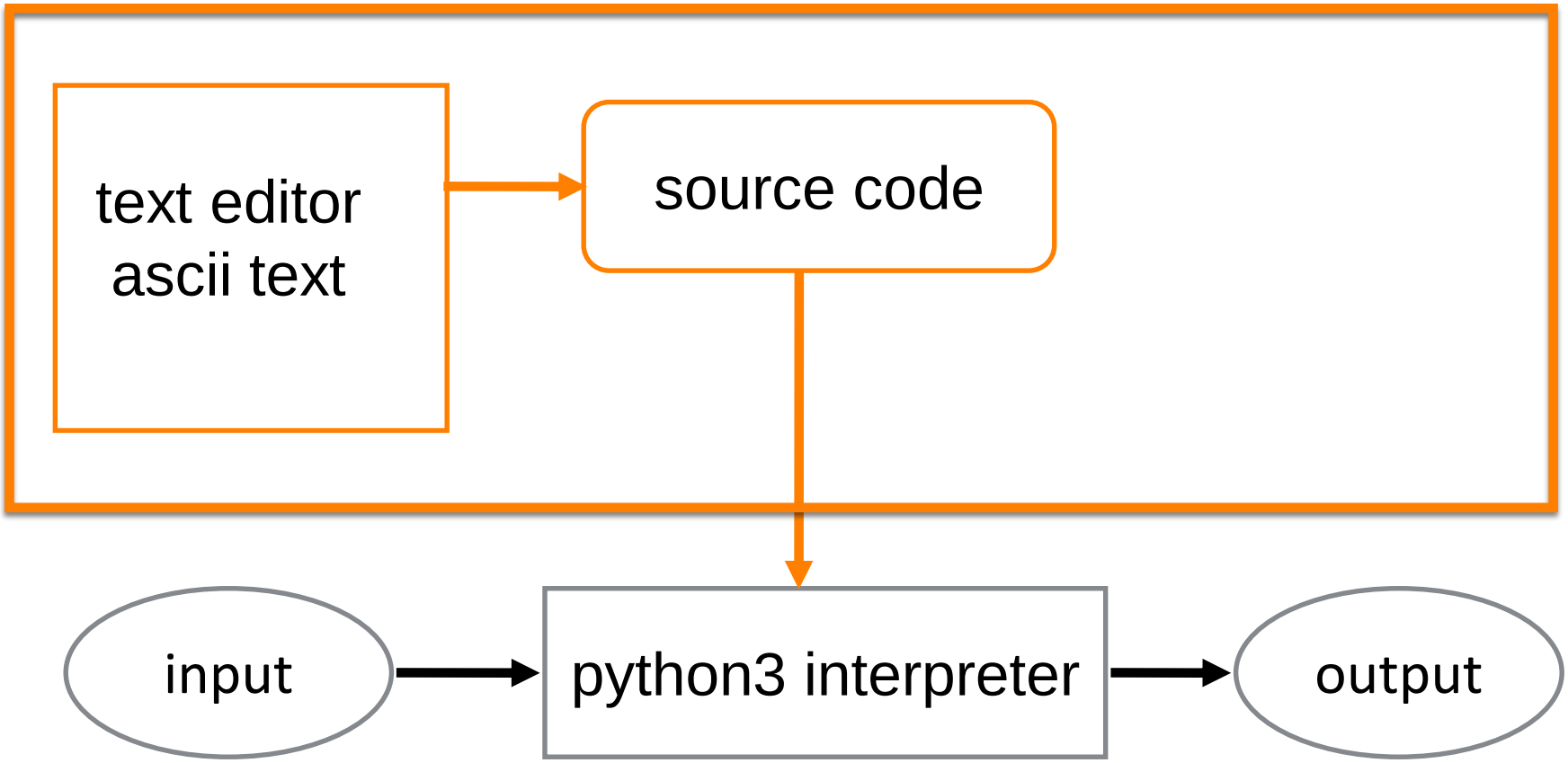
Stack Overflow https://stackoverflow.com/ a good friend of yours:
for **learning from others** but **not for copy-pasting others code**!

source: https://stackoverflow.blog/2017/09/06/incredible-growth-python/
source: https://hackernoon.com/future-of-python-language-bright-or-dull-uv41u3xwx

THE PROGRAM

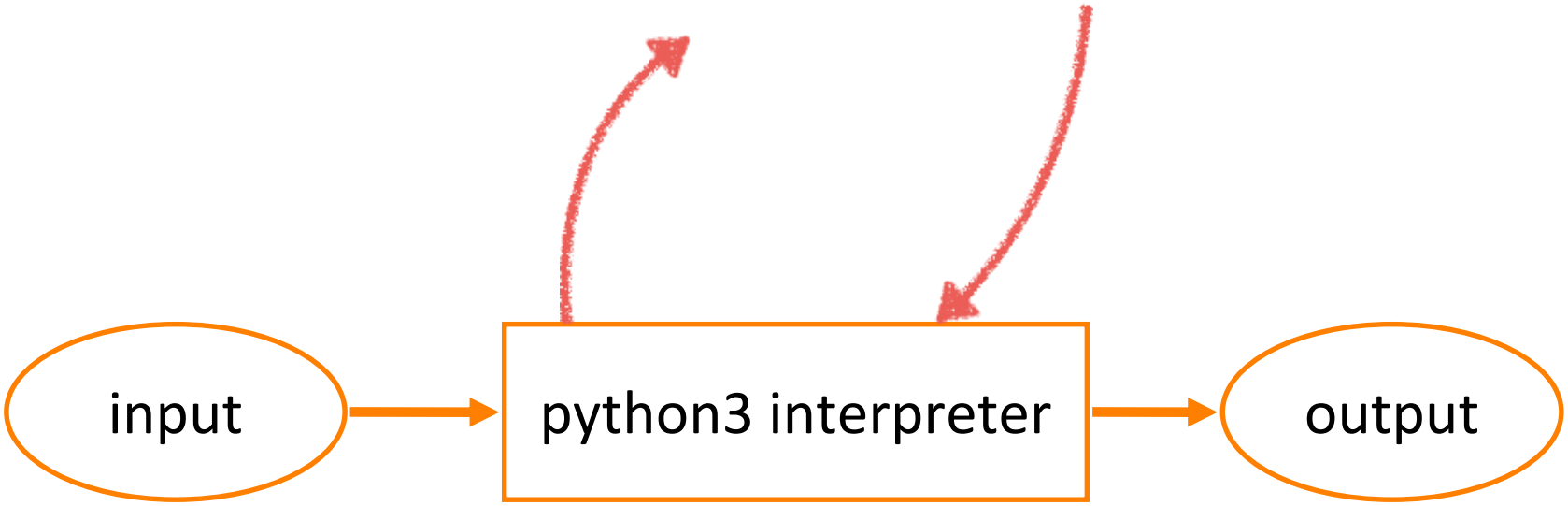
- **Program** is a sequence of instructions that specifies how to perform a computation.
- **Input** - get data from the keyboard, a file, device ..
- **Output** - display data on the screen or send data to a file or other device (client/server, local/remote).
- **Math** - perform mathematical operations (**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

Entering commands – in two modes:

1. Immediate mode using python console (quick testing)
2. Script mode using IDE or text editor (development)
3. IPython Notebook (presentation)



TIME TO CODE!

THE ZEN OF 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!
>>>
```

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

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

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

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

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

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**
  (variable name must begin with letter or underscore _)

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

VARIABLES

cannot begin with a **number**

this **$** is illegal character

class is reserved keyword

```
>>> 76trombones = "big parade"
SyntaxError: invalid syntax
>>> more$ = 1000000
SyntaxError: invalid syntax
>>> 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 _)

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

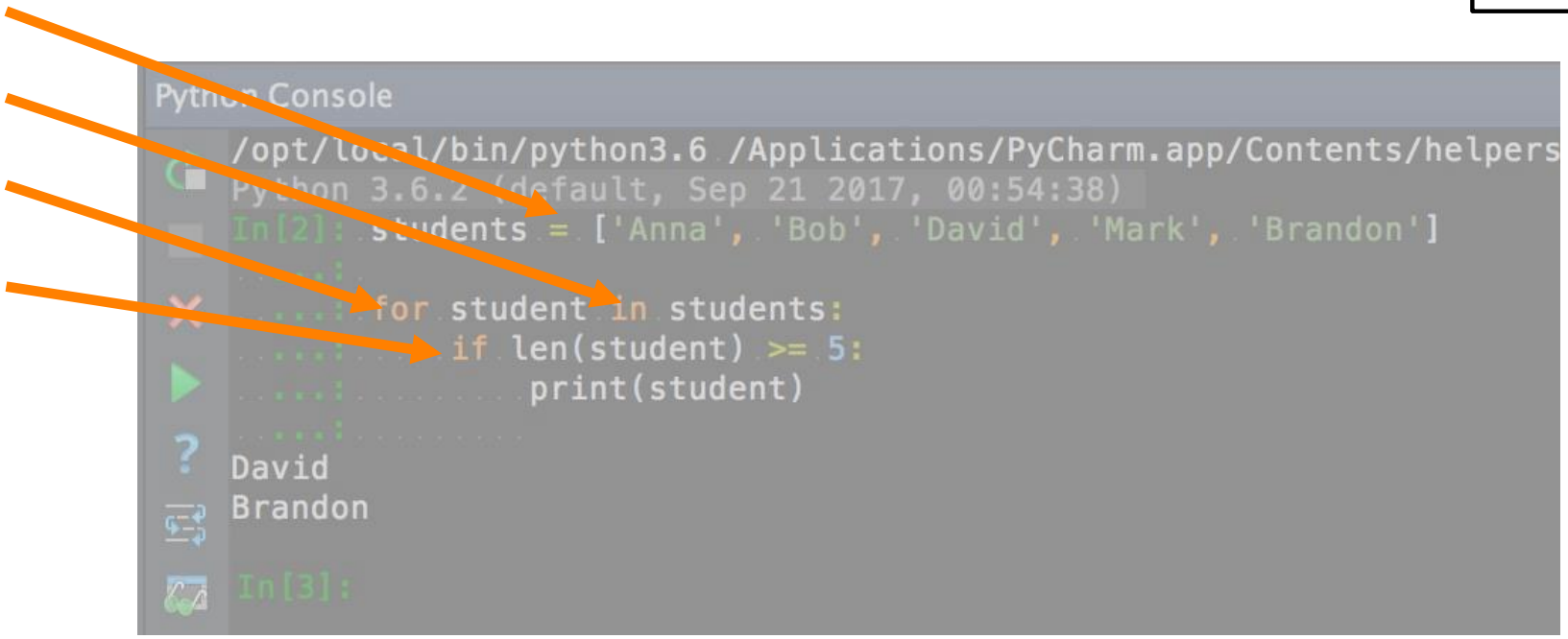
COMMENTS

```
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

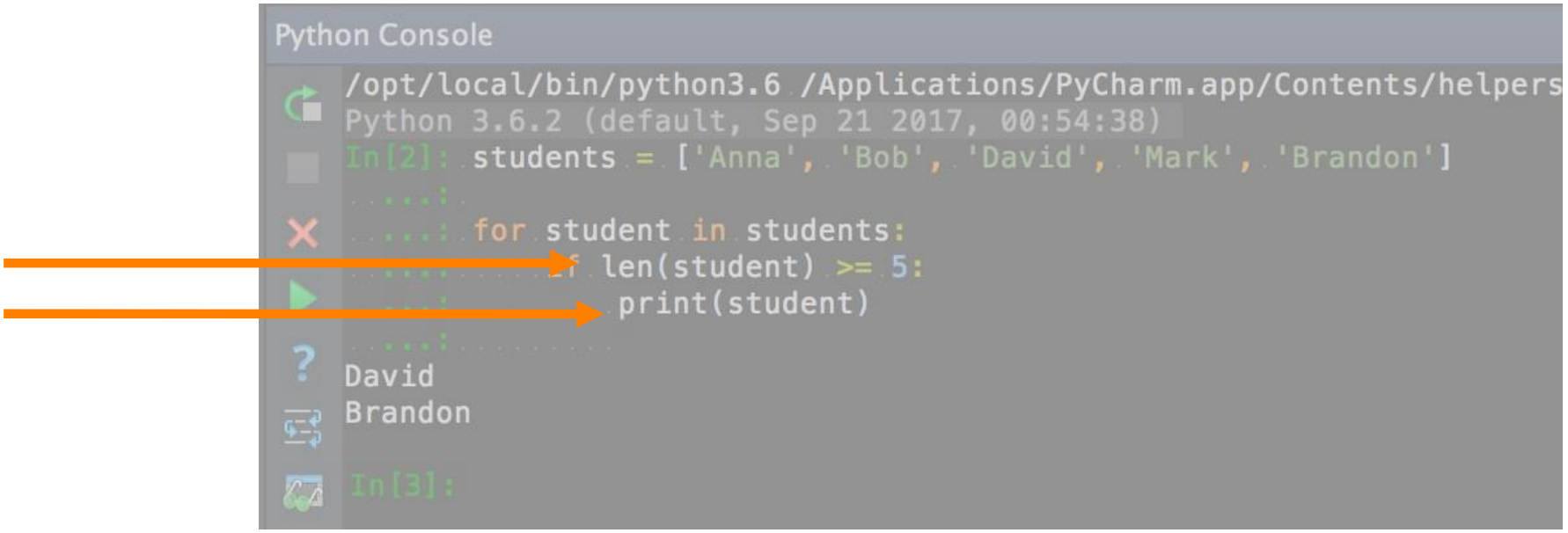
STATEMENTS



- 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

TIME FOR DEMOS!

(why we really use python)

REFERENCES

- 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/