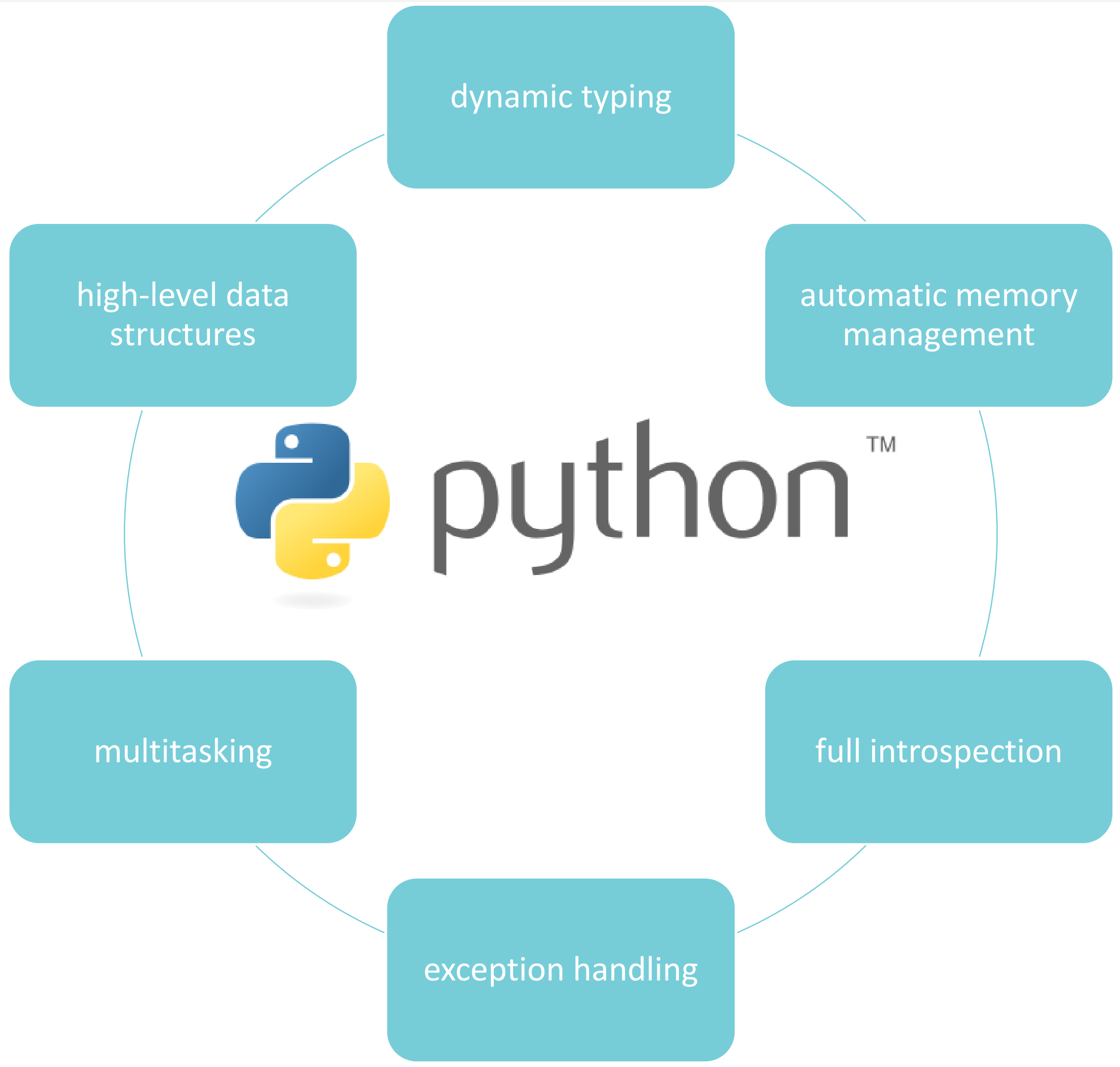
Python Programming Language Foundation

Session I

What is Python?

What is Python?



Capabilities

- Interpreted
- Interactive
- Object-Oriented
- Portable
- Extendable

History

- was conceived in the late 1980s
- successor to ABC programming language
- **Guido van Rossum** creator and «Benevolent dictator for life » of the Python

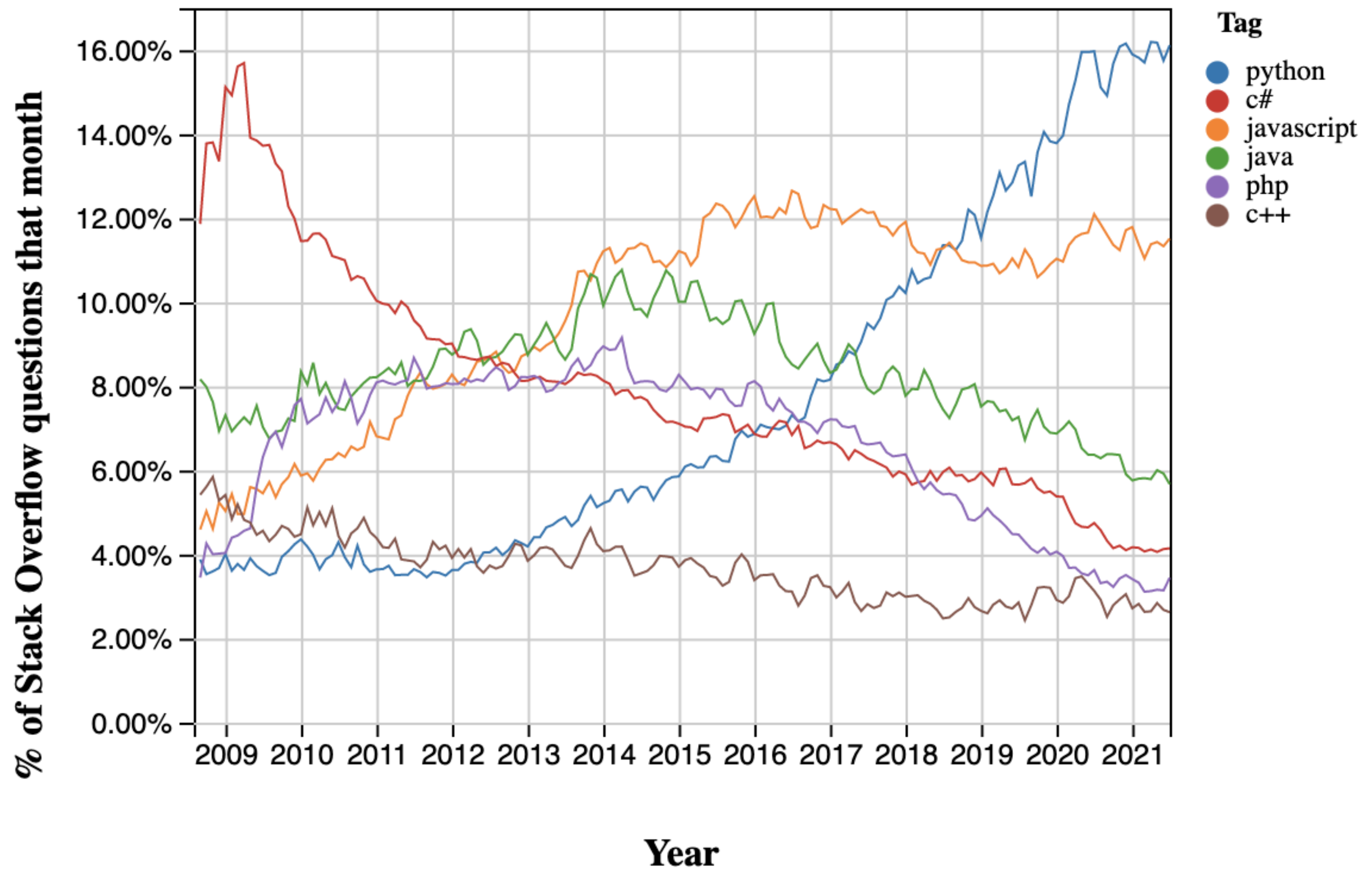
History

In July 2018, Van Rossum announced that he would be stepping down from the position of "Benevolent dictator for life" (BDFL) of the Python programming language.

… I would like to remove myself entirely from the decision process. I'll still be there for a while as an ordinary core dev, and I'll still be available to mentor people -- possibly more available. But I'm basically giving myself a permanent vacation from being BDFL, and you all will be on your own. …

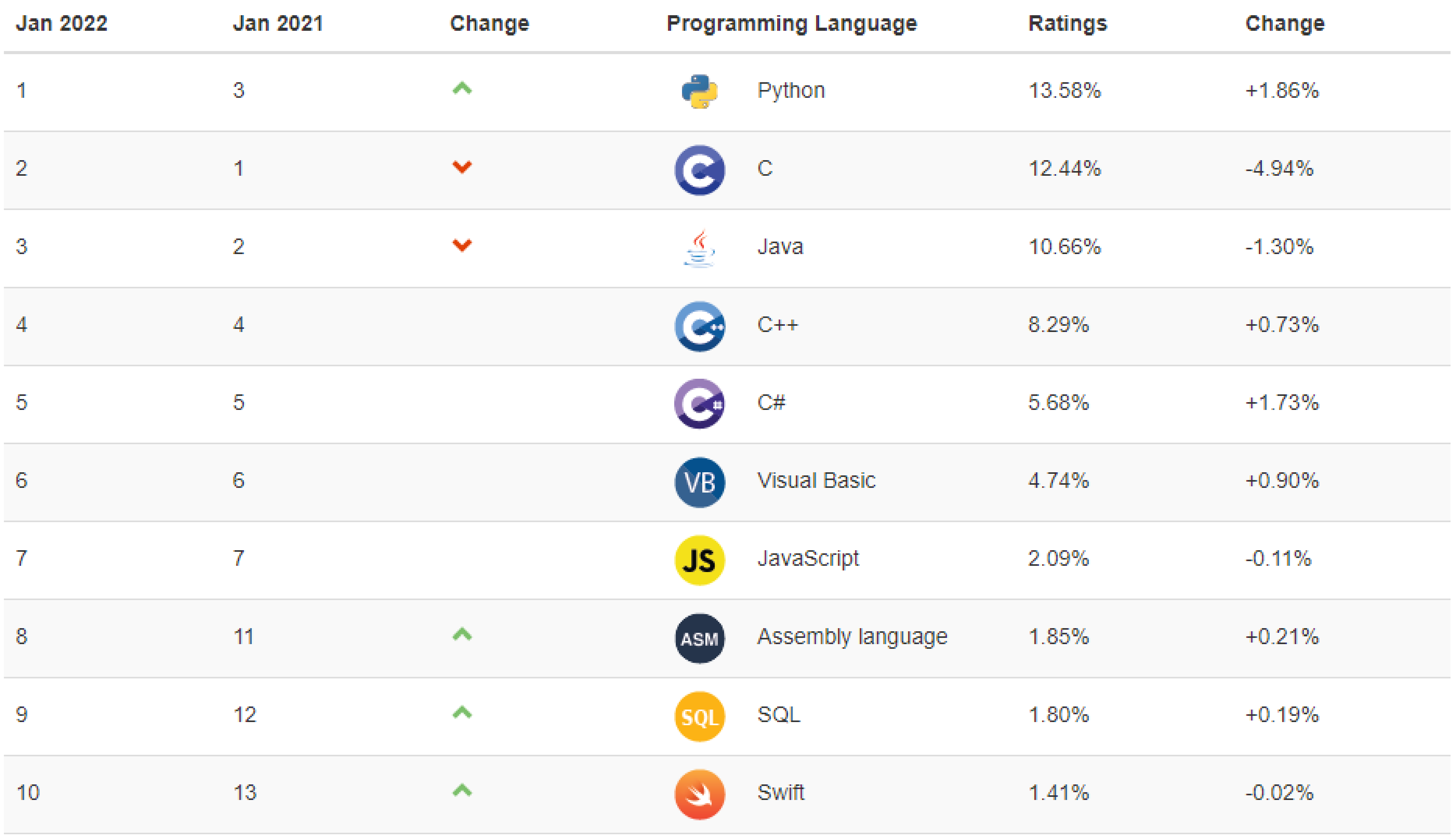
Mail from Guido Van Rossum: https://mail.python.org/pipermail/python-committers/2018-July/005664.html

Stack Overflow stats



Source: https://insights.stackoverflow.com/trends

Tiobe index



Jan 2022	Jan 2021	Change	Programming Language	Ratings	Change
1	3	▲	Python	13.58%	+1.86%
2	1	▼	C	12.44%	-4.94%
3	2	▼	Java	10.66%	-1.30%
4	4		C++	8.29%	+0.73%
5	5		C#	5.68%	+1.73%
6	6		Visual Basic	4.74%	+0.90%
7	7		JavaScript	2.09%	-0.11%
8	11	▲	Assembly language	1.85%	+0.21%
9	12	▲	SQL	1.80%	+0.19%
10	13	▲	Swift	1.41%	-0.02%

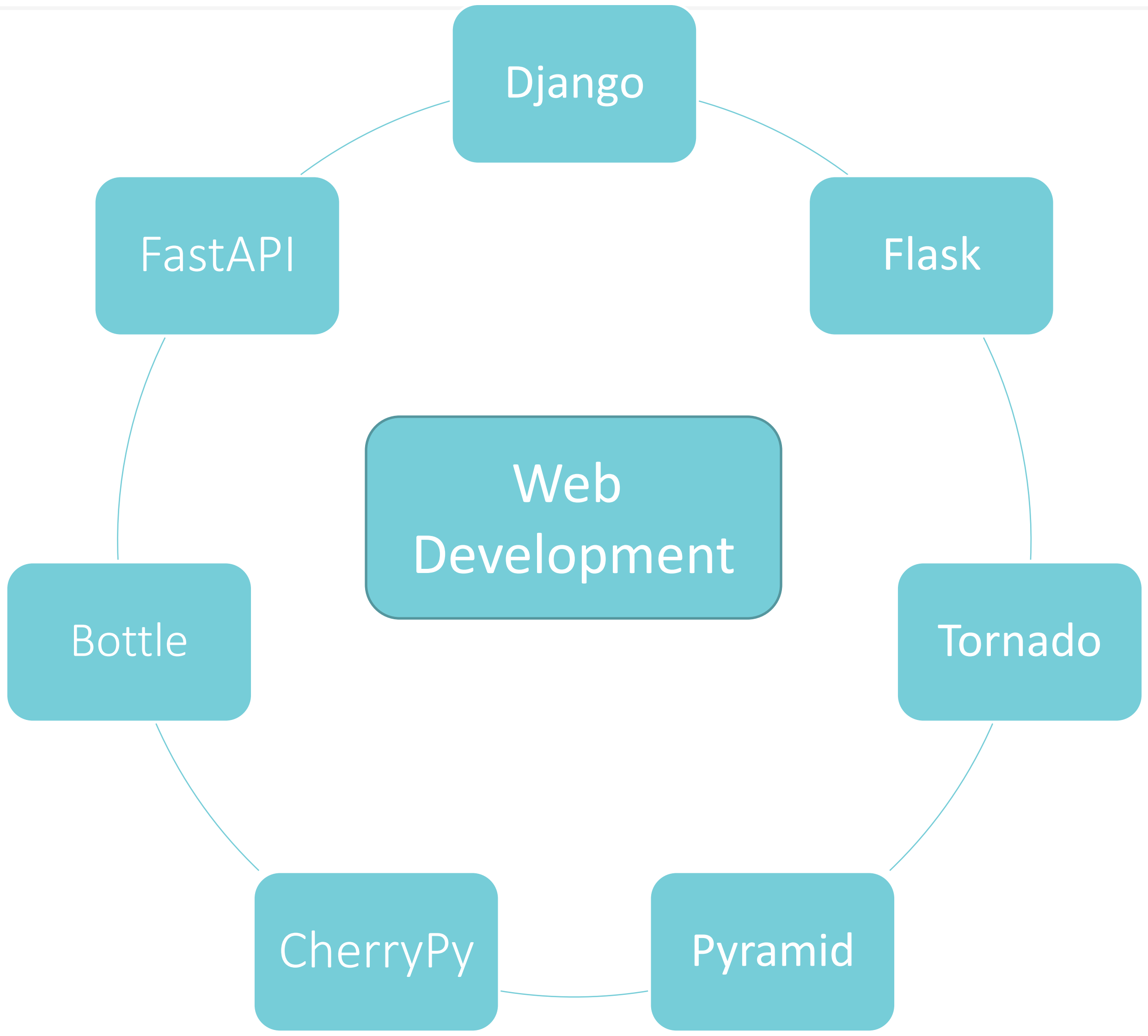
Source: https://www.tiobe.com/tiobe-index/

What can I do with Python?

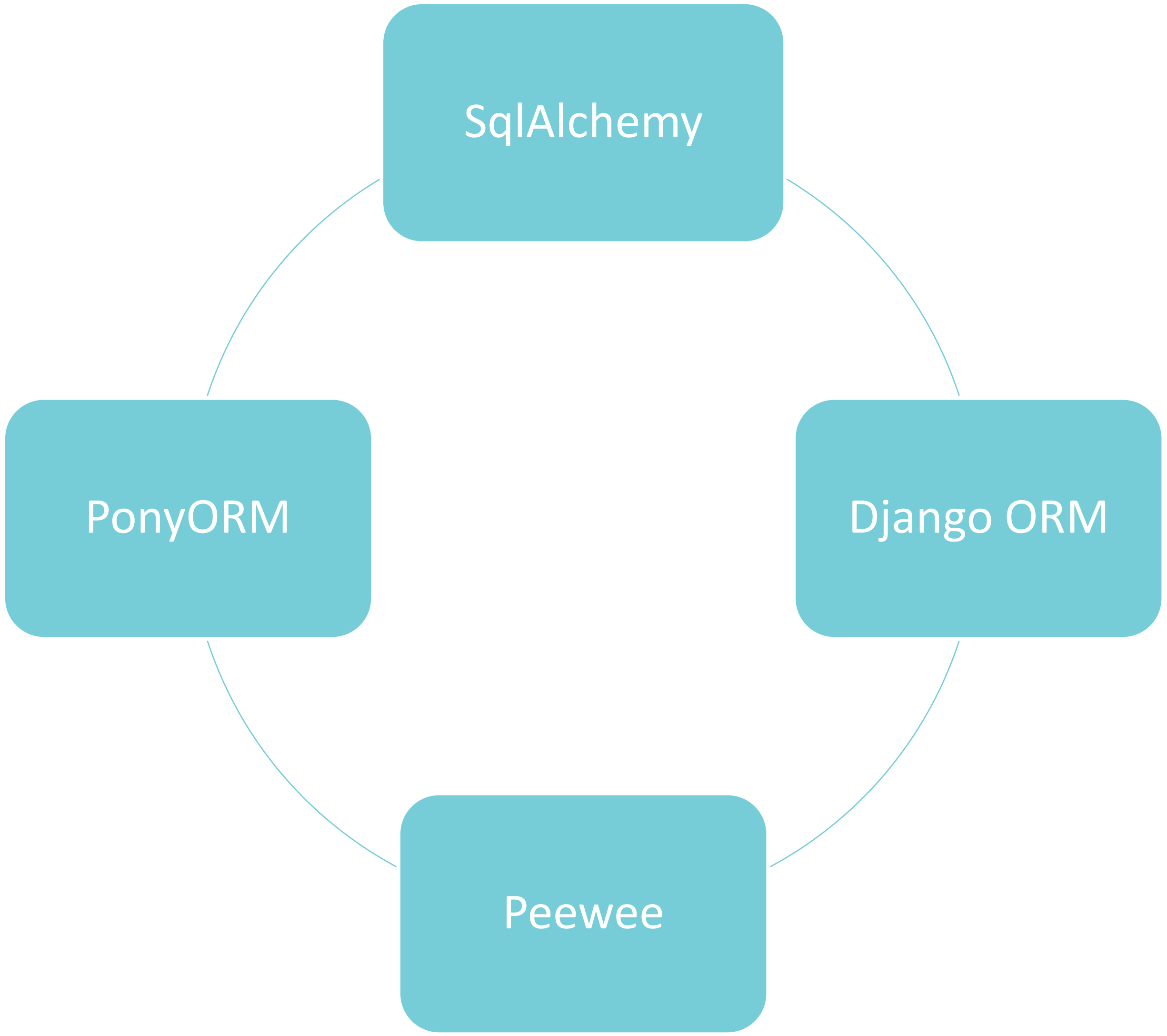
What can I do with Python?

Literally
Everything

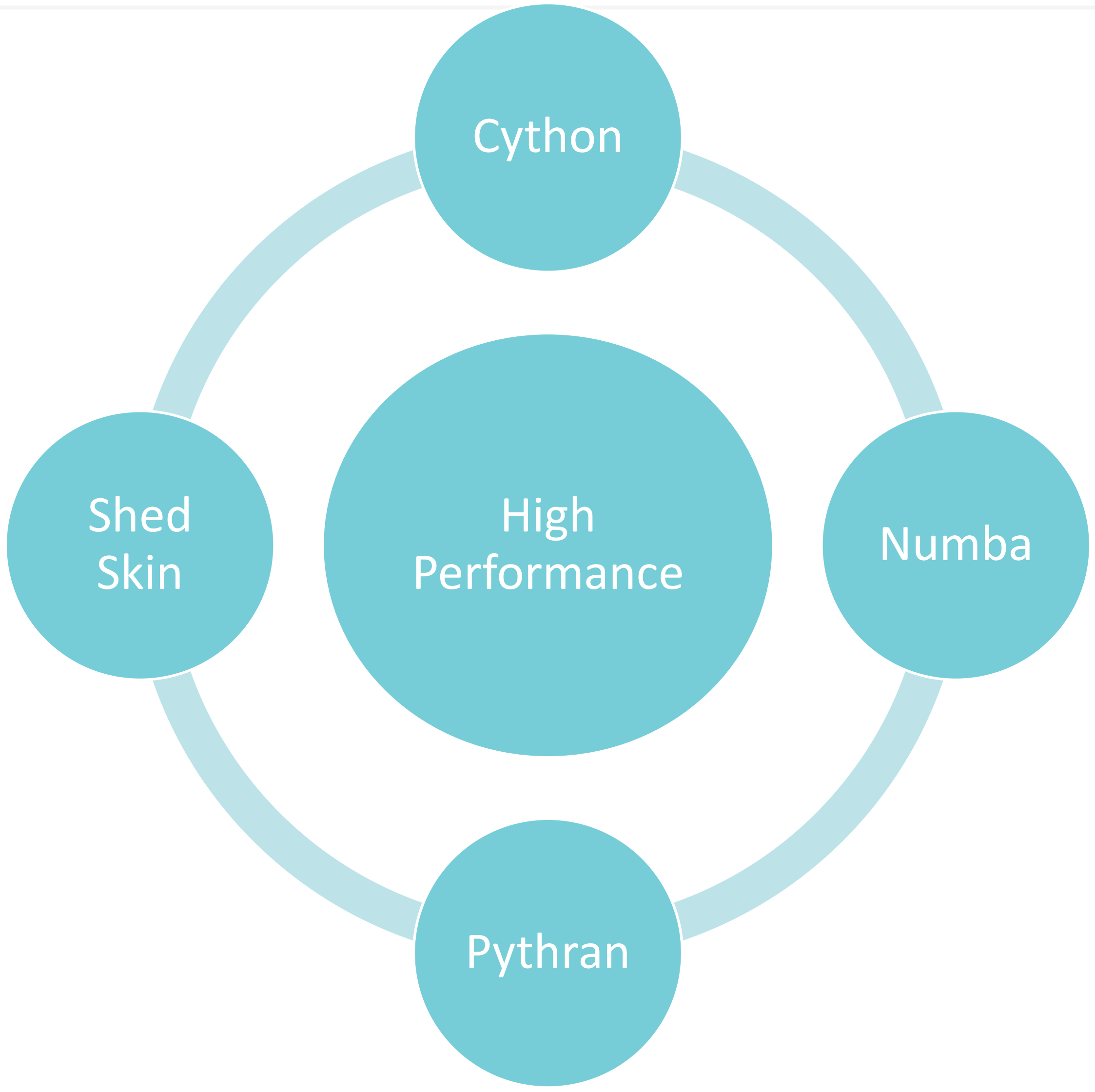
Web development



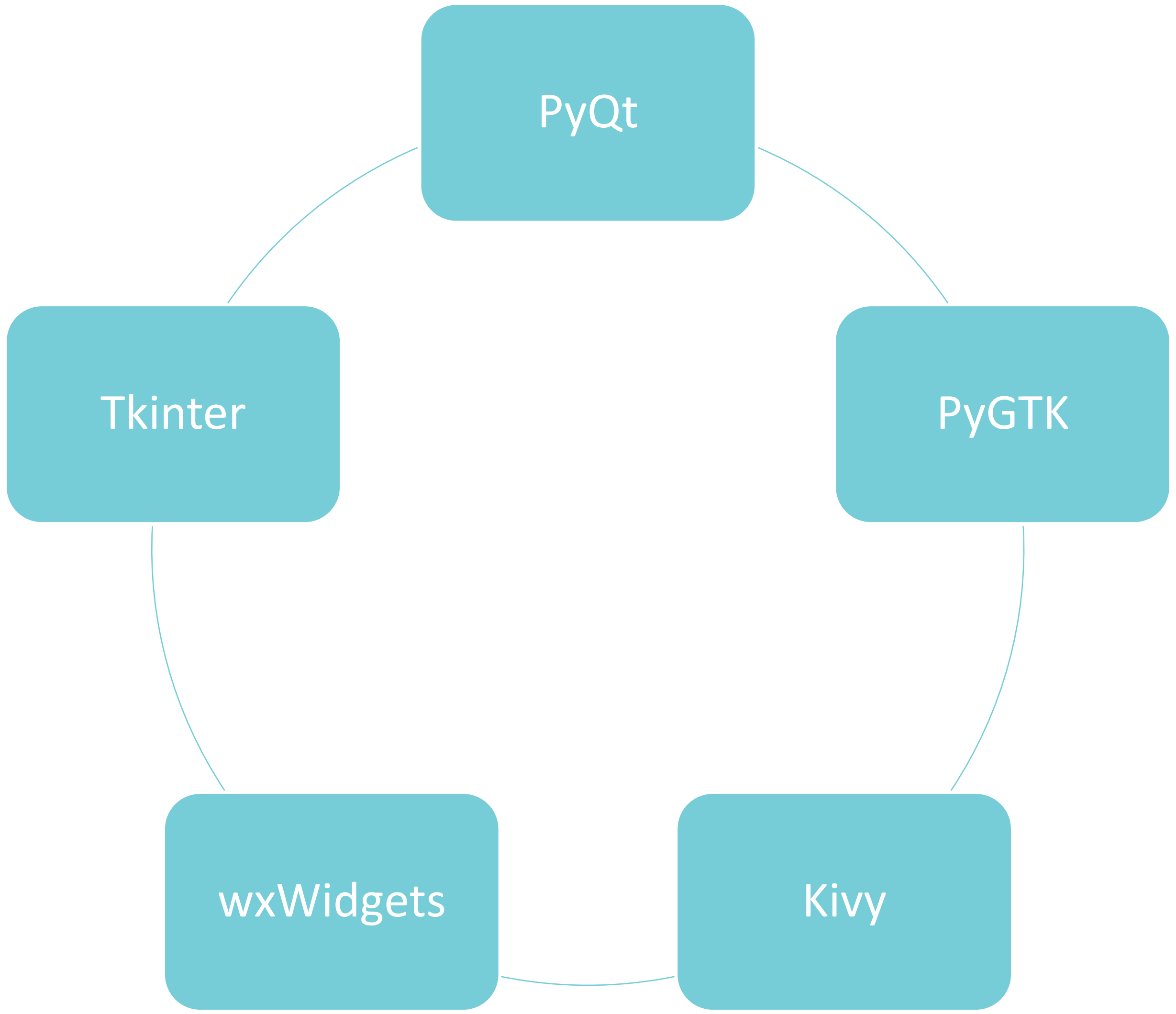
ORMs



High performance



GUI apps



Science

- Anaconda
- Scikit-learn
- NumPy
- Pandas
- Biopython
- SymPy
- Jupyter
- Tensorflow
- PySpark
- OpenCV

What IDEs can I use for Python development?

PyCharm

Visual Studio

Visual Studio

Vim / Vi

Atom

Sublime Text

Is there anything great written in Python?

Google

“Python has been an important part of Google since the beginning, and remains so as the system grows and evolves.
Today dozens of Google engineers use Python”

OpenStack

openstack®

“OpenStack, a cloud computing IaaS platform”

BitTorrent

“BitTorrent, original client, along with several derivatives”

Dropbox

“Dropbox, a web-based file hosting service”

Instagram

“Instagram's backend was written in Python.”

Reddit

“Reddit was originally written in Common Lisp,
but was rewritten in Python in 2005”

World of Tanks

“World of Tanks uses Python for most of its tasks”

List of Python software

https://en.wikipedia.org/wiki/List_of_Python_software

Python – multi-paradigm
dynamic programming language
with strong implicit typing

Static / dynamic typing

- **Static type** checks are performed without running the program. Typically, this is done as your program is compiled.
- **Dynamic typing** refers to when a language is more agnostic to data types until the program executes.

Examples:
Static: C, Java, C#;
Dynamic: Python, JavaScript, Ruby.

Static / dynamic typing

Dynamic / Python

```
def mean( array ):
    result = 0
    for item in array:
        result += item
    return result / len(array)
```

Static / C

```
double mean( double array[], int size ) {
    double result = 0.0;
    for ( int i = 0; i < size; ++i )
    {
        result += array[i];
    }
    return result / size;
}
```

Strong / weak typing

- **Weakly-typed** language automatically converts the types according to its own set of rules when you perform certain operations using data of different types. Languages can make conversions between unrelated types *implicitly*;
- **Strongly-typed** languages *don’t allow implicit* conversions between unrelated types

Examples:
Strong: Java, Python, Haskell, Lisp;
Weak: C, JavaScript, Visual Basic, PHP

Strong / weak typing

Strong / Python

```
number = 42 + "666"
TypeError: Unsupported operand type(s)
for +: "int" and "str"

number = str(42) + "666"
"42666"

number = 42 + int("666")
708

number = "42" + "666"
"42666"
```

Weak / JavaScript

```
var number = 42 + "666"
"42666"
```

Explicit / implicit typing

- **Explicit** – generally it's manifestly adding type to our codebase. We have to know exactly what kind of type the value is.
- **Implicit** – means that the type is inferred by language type inference system which takes responsibility away from us of writing the types.

Examples:
Explicit: C++, D, C#
Implicit: Python, PHP, Lua, JavaScript

Explicit / implicit typing

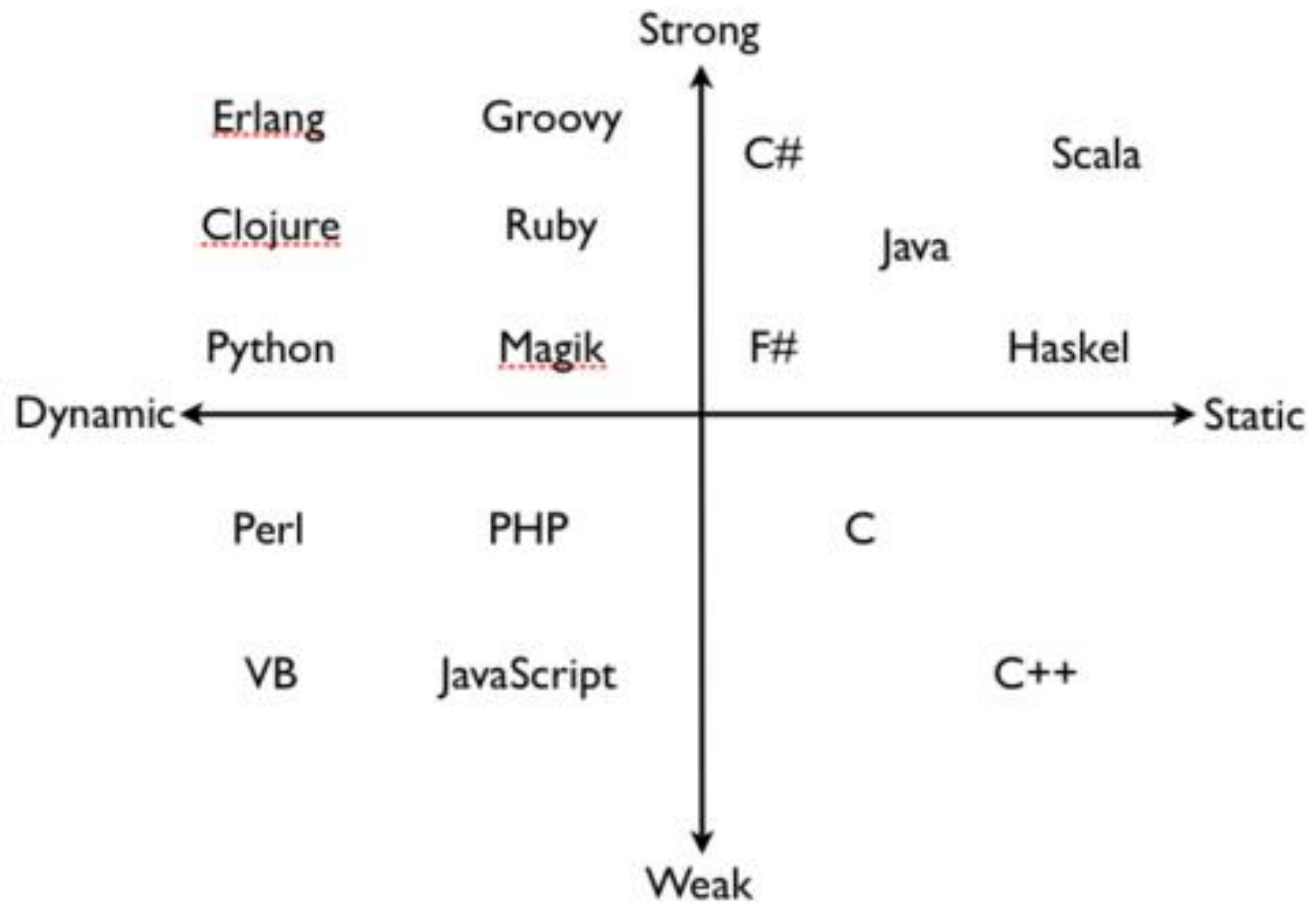
Implicit / Python

```python
number = 5
number = "5"
number = [1,2,3]
```

Explicit / C

```c
int number_i = 5;
char number_c = '5';
int number_a[] = {1,2,3};
```

Languages by typing strategy



How can I install Python?

Versions

Python 1.0
Jan 1994

- Python 1.5 - 31 Dec 1997
- Python 1.6 - 5 Sep 2000

Python 2.0
16 Oct 2000

- Python 2.1 - 17 Apr 2001
- Python 2.2 - 21 Dec 2001
- Python 2.3 - 29 Jul 2003
- Python 2.4 - 30 Nov 2004
- Python 2.5 - 19 Sep 2006
- Python 2.6 - 1 Oct 2008
- Python 2.7 - 03 Jul 2010

Python 3.0
3 Dec 2008

- Python 3.1 — 27 Jun 2009
- Python 3.2 — 20 Feb 2011
- Python 3.3 — 29 Sep 2012
- Python 3.4 — 16 Mar2014
- Python 3.5 — 13 Sep 2015
- Python 3.6 — 23 Dec 2016
- Python 3.7 — 27 Jun 2018
- Python 3.8 — 14 Oct 2019
- Python 3.9 — 05 Oct 2020
- Python 3.10 – 04 Oct 2021
- Python 3.11 – coming in Oct 2022

Environment

Python Official Website: http://www.python.org/

Python Documentation Website: www.python.org/doc/

How to install Python?

- http://www.python.org/download/
- ./configure
- make
- make install

Unix&Linux

- http://www.python.org/download/
- *python-XYZ.msi*

Windows

How to install Python?

```
user@user:~$ apt-get install python3
user@user:~$ apt-get install python3-pip
```

Interactive Python

Interactive Python

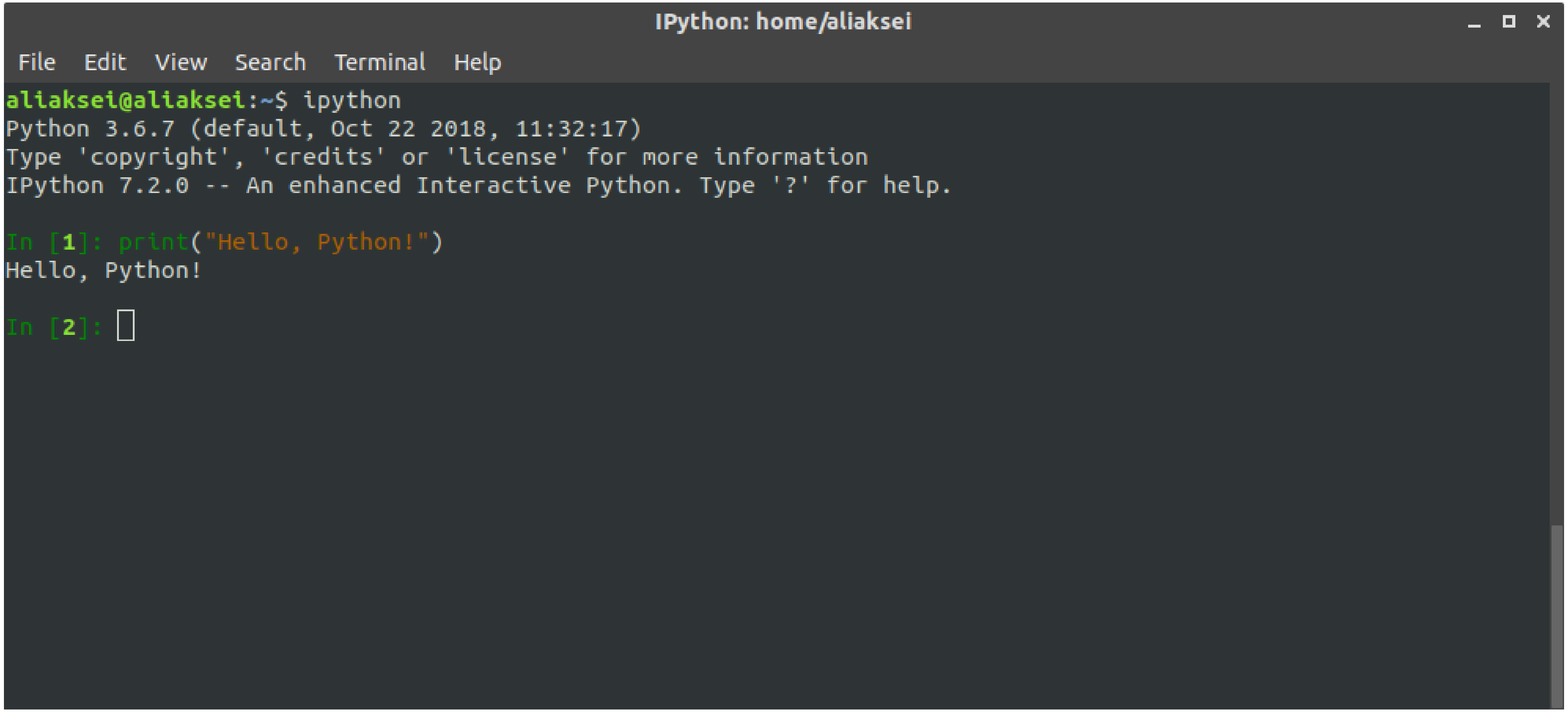
```
user@user:~$ python3
Python 3.6.5 (default, Apr  1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for
more information.

>>> print("Hello world!")
Hello world!
```

ipython

```
user@user:~$ pip install ipython
```

Documentation: https://ipython.org/

ipython



```
aliaksei@aliaksei:~$ ipython
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.2.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: print("Hello, Python!")
Hello, Python!

In [2]: 
```

Python scripts

```
#!/usr/bin/python
print("Hello, Python!")
```

```
user@user:~$ chmod +x test.py
user@user:~$ ./test.py
Or
user@user:~$ python ./test.py
```

Shebang

The #! syntax used in scripts to indicate an interpreter for execution under UNIX / Linux operating systems.

Examples:

Bash: *#!/bin/bash*

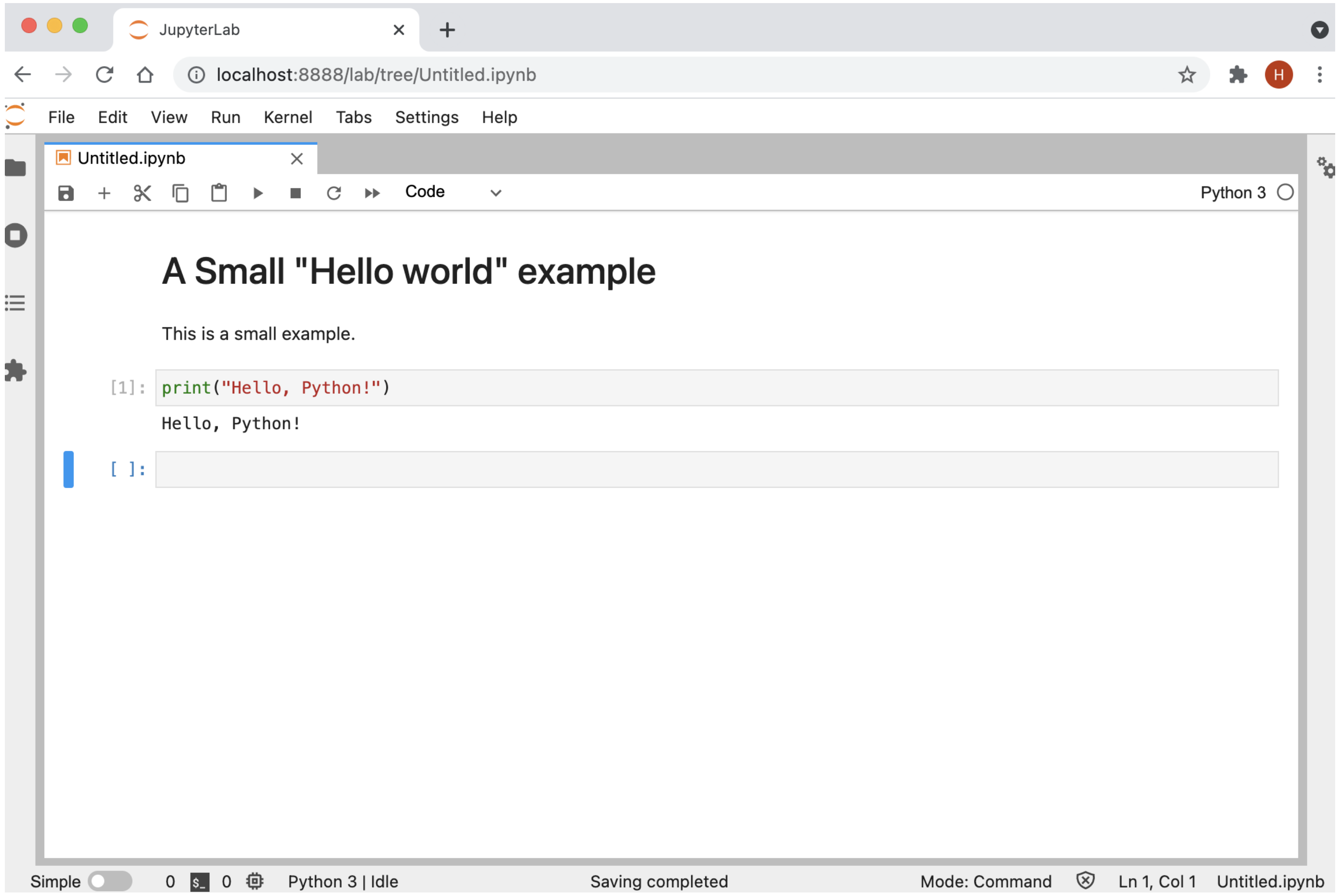
Python: *#!/usr/bin/python3*

Jupyter notebook

```
user@user:~$ pip install jupyterlab
```

Documentation: https://jupyter.org/

Jupyter notebook



A Small "Hello world" example

This is a small example.

```
[1]: print("Hello, Python!")
Hello, Python!
[ ]:
```

Python Syntax

Identifiers

- A-Z,a-z,_,0-9
- Case sensitive: EPAM != Epam

Key words

```
import keyword
print(keyword.kwlist)
```

Indentation

```python
if True:
    print("True")
else:
    print("False")
```

```python
if True:
   print("True")
    print("ERROR")
else:
       print("False")
```

Multiline statements

\ - line continuation character:

```
if a == True and \
   b == False
```

```
a = '1' + '2' + '3' + \
    '4' + '5'
```

Parentheses can be used:

```
if (a == True and
    b == False)
```

```
a = ('1' + '2' + '3' +
     '4' + '5')
```

Multiline statements

```
days = [
    'Monday',
    'Tuesday',
    'Wednesday',
    'Thursday',
    'Friday',
]
```

```
names = (
    'Jason',
    'Arnold',
    'Jean',
    'Ernie',
    'Julia',
)
```

```
values = {
    value_1,
    value_2,
    value_3,
    value_4,
    value_5,
}
```

Quotes

```python
word = 'word'

sentence = "This is a sentence."

paragraph = """This is a paragraph.
It is made up of multiple lines and sentences."""

inside_1 = "Quotation 1 \"inside\"."

inside_2 = 'Quotation 2 \'inside\'.'
```

Comments

```python
#!/usr/bin/python

# First comment
print("Hello, Python!") # second comment
```

Types of variables

- A **variable** works like *a reference to an object*. It doesn’t have a specific type.
- **Types** are specified for objects.

```
#!/usr/bin/python

variable = 100          # An integer assignment
print(variable)
variable = 1000.0       # A floating point
print(variable)
variable = "John"       # A string
print(variable)
```

`import this` or Python Zen

- *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!*

Materials for beginners

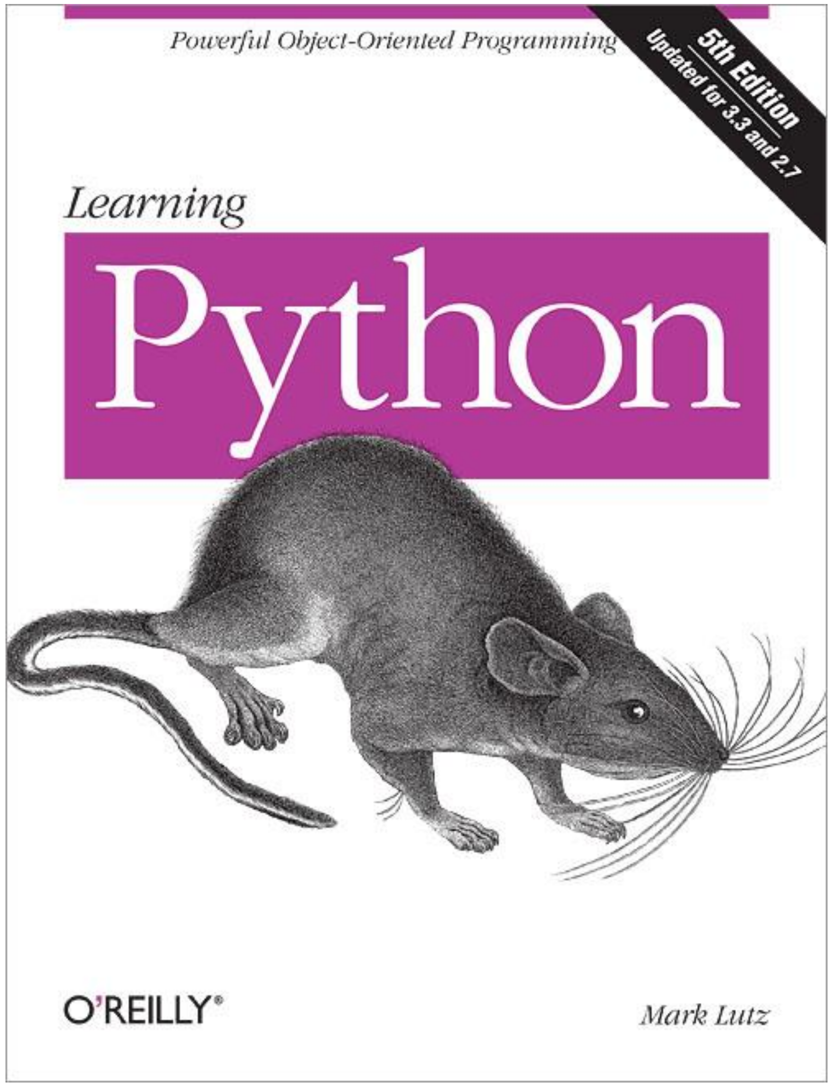
Python Tutorial

https://docs.python.org/3/tutorial/

Python Materials

https://wiki.python.org/moin/BeginnersGuide/Programmers

Materials for beginners



Powerful Object-Oriented Programming

5th Edition

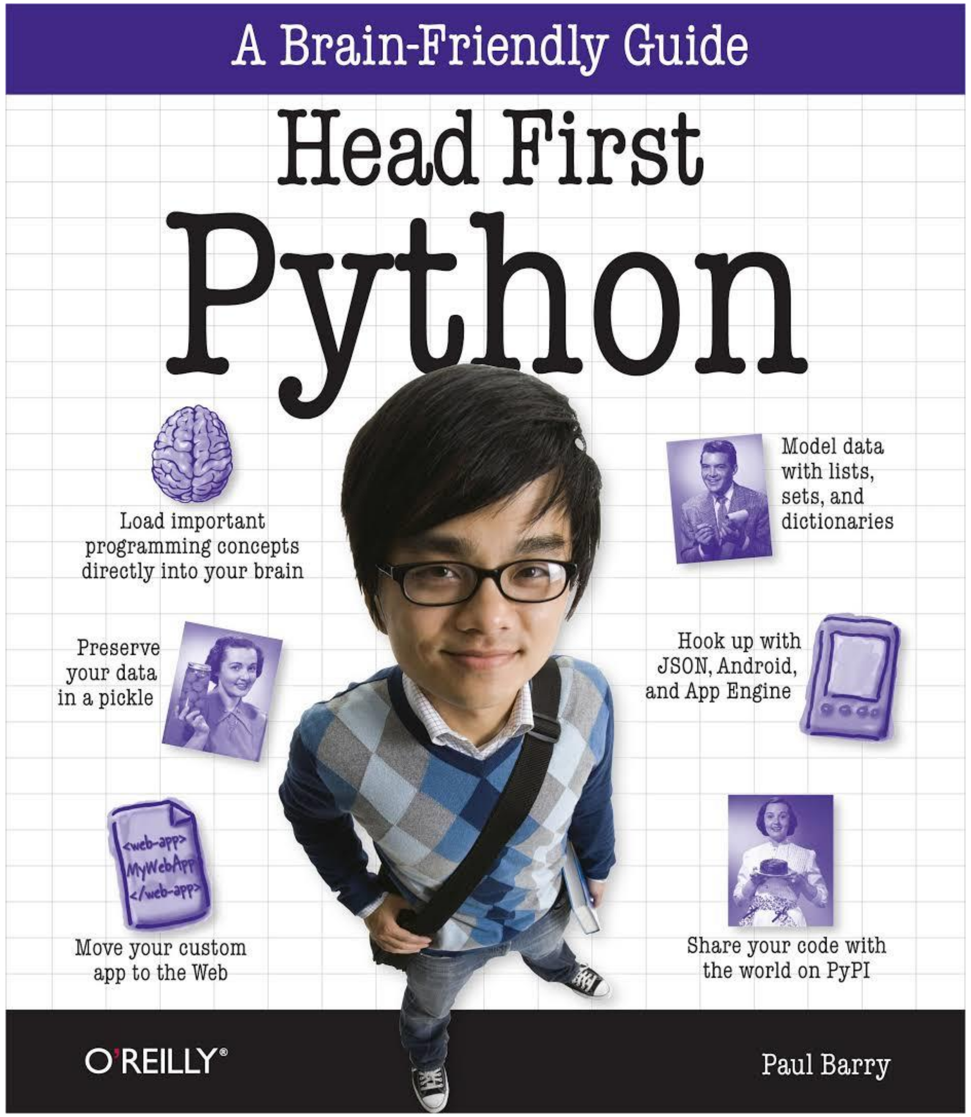
Updated for 3.3 and 2.7

Learning Python

O'REILLY®

Mark Lutz

Materials for beginners


A Brain-Friendly Guide

Head First Python

Load important programming concepts directly into your brain

Model data with lists, sets, and dictionaries

Preserve your data in a pickle

Hook up with JSON, Android, and App Engine

Move your custom app to the Web

Share your code with the world on PyPI

O'REILLY®

Paul Barry

Thanks for attention