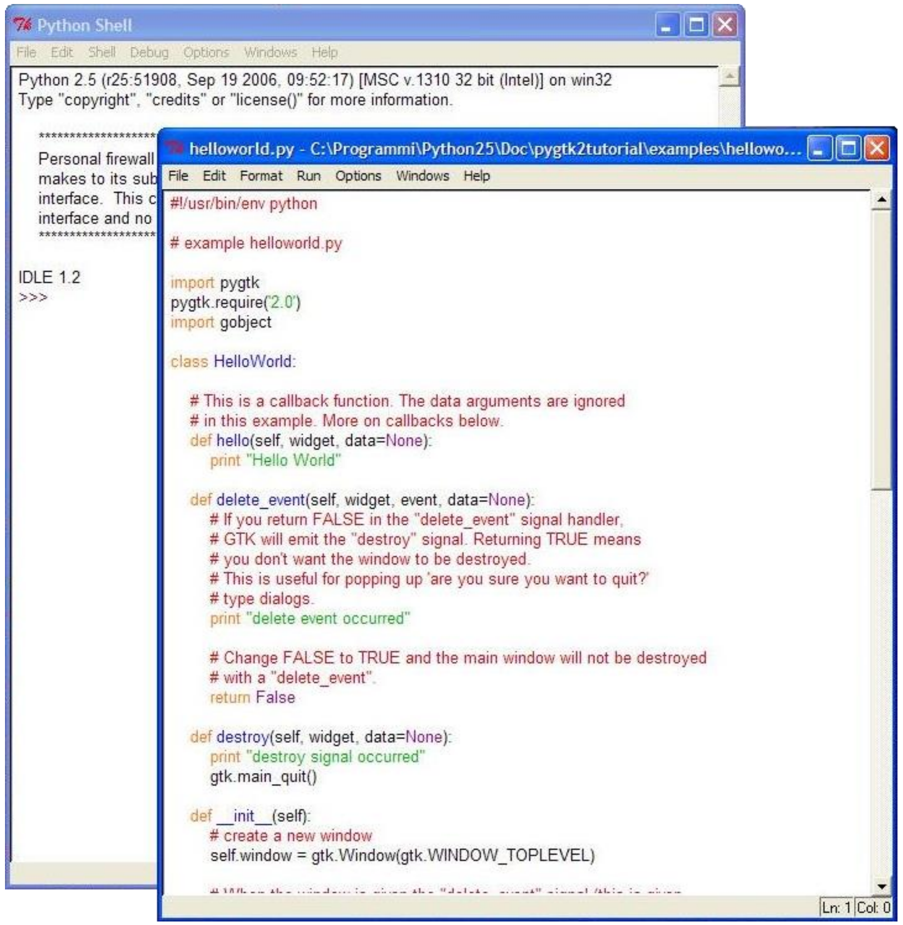
Getting started with Python

What is Python?
Python is a general-purpose programming language

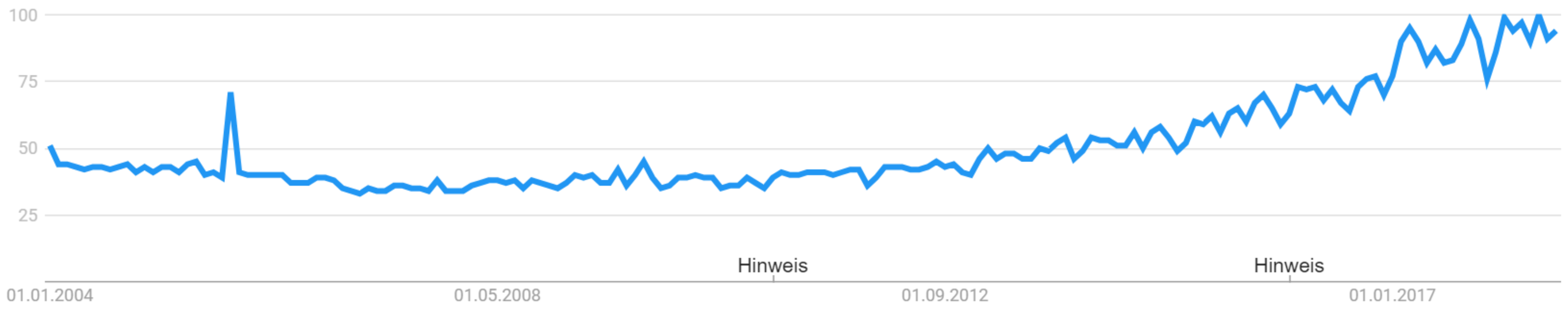
- Python is a high-level programming language.
- Python comes with the Integrated Development Environment "**IDLE**" that interprets the Python language and provides two ways to perform analyses:
  1. Python Shell window
  2. Multi-window text editor
- ➢ However, separate windows for the Python shell, text editing window, and graphic device can get confusing.


Python Shell
File Edit Shell Debug Options Windows Help
helloworld.py - C:\Programmi\Python25\Doc\pygtk2tutorial\examples\hellowo...
File Edit Format Run Options Windows Help
Ln: 1 Col: 0

```
Python 2.5 (r25:51908, Sep 19 2006, 09:52:17) [MSC v.1310 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

    ********************
    Personal firewall
    makes to its sub
    interface.  This c
    interface and no
    ********************

IDLE 1.2
>>>
```

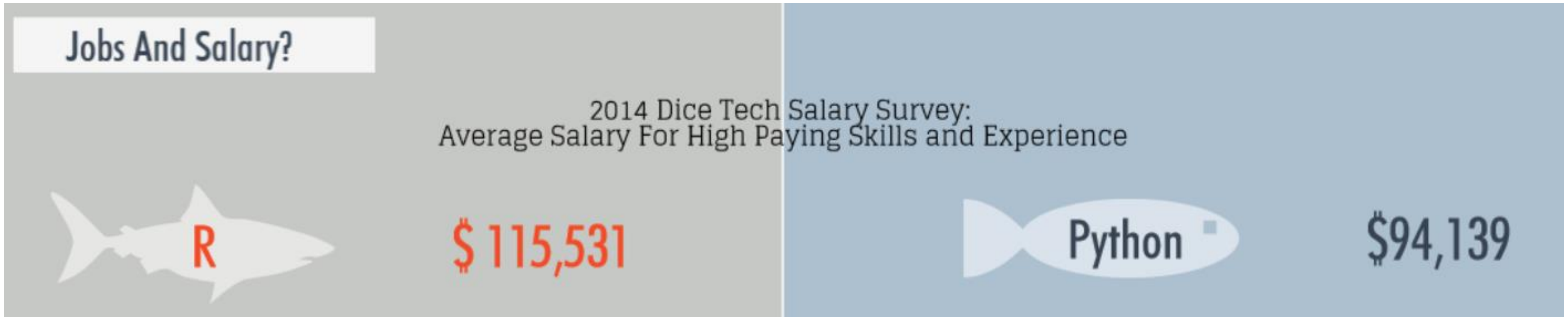
```
#!/usr/bin/env python

# example helloworld.py

import pygtk
pygtk.require('2.0')
import gobject

class HelloWorld:

    # This is a callback function. The data arguments are ignored
    # in this example. More on callbacks below.
    def hello(self, widget, data=None):
        print "Hello World"

    def delete_event(self, widget, event, data=None):
        # If you return FALSE in the "delete_event" signal handler,
        # GTK will emit the "destroy" signal. Returning TRUE means
        # you don't want the window to be destroyed.
        # This is useful for popping up 'are you sure you want to quit?'
        # type dialogs.
        print "delete event occurred"

        # Change FALSE to TRUE and the main window will not be destroyed
        # with a "delete_event".
        return False

    def destroy(self, widget, data=None):
        print "destroy signal occurred"
        gtk.main_quit()

    def __init__(self):
        # create a new window
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
```

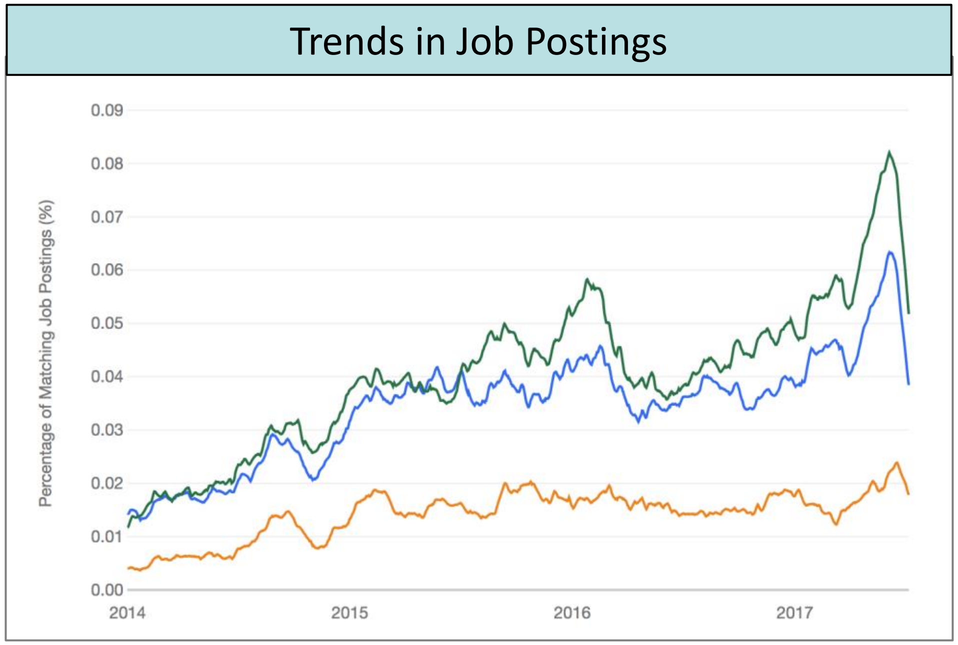
How popular is Python (vs. R)?
Python is a popular computing language for data analysis

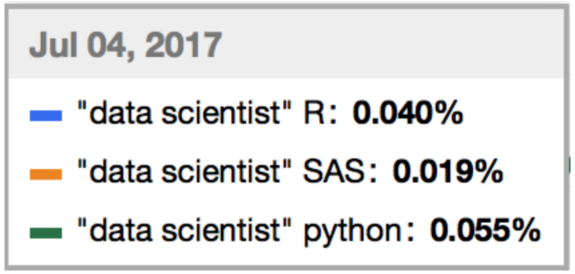


Google Trends (2018)

How popular is Python (vs. R)?
One of the highest paid skills on the job market

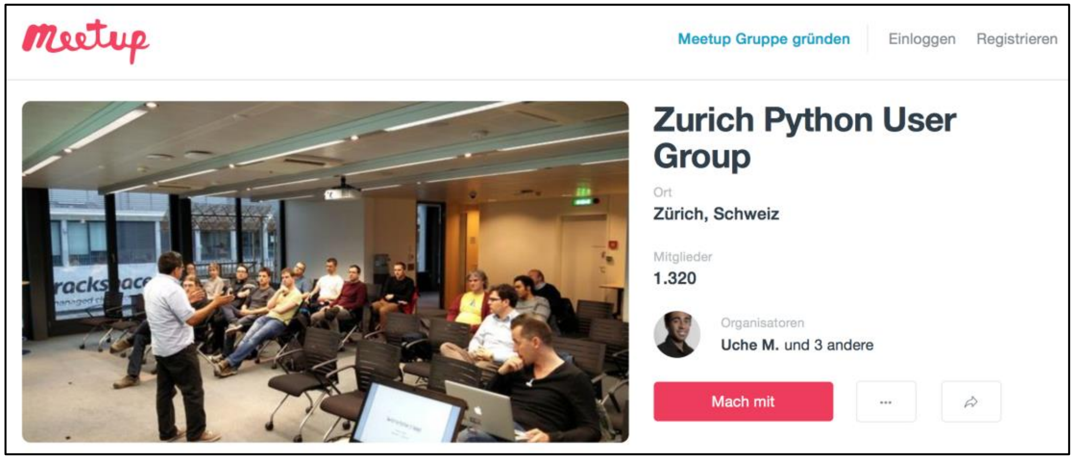

Jobs And Salary?
2014 Dice Tech Salary Survey:
Average Salary For High Paying Skills and Experience
R
$ 115,531
Python
$94,139





Indeed Job Trends (2017)

How popular is Python?
There are local Python user communities in Zurich



Meetup Gruppe gründen | Einloggen | Registrieren

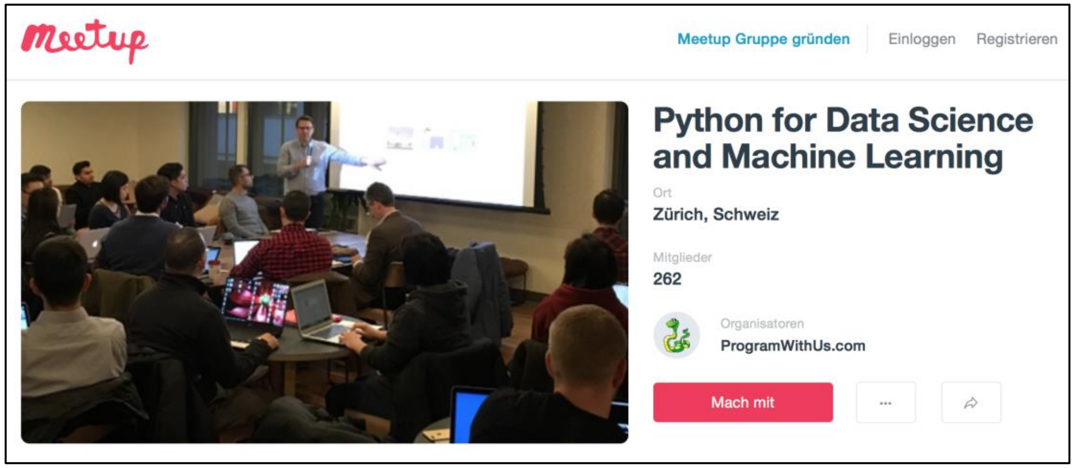
Zurich Python User Group

Ort
Zürich, Schweiz

Mitglieder
1.320

Organisatoren
Uche M. und 3 andere

Mach mit



Meetup Gruppe gründen | Einloggen | Registrieren

Python for Data Science and Machine Learning

Ort
Zürich, Schweiz

Mitglieder
262

Organisatoren
ProgramWithUs.com

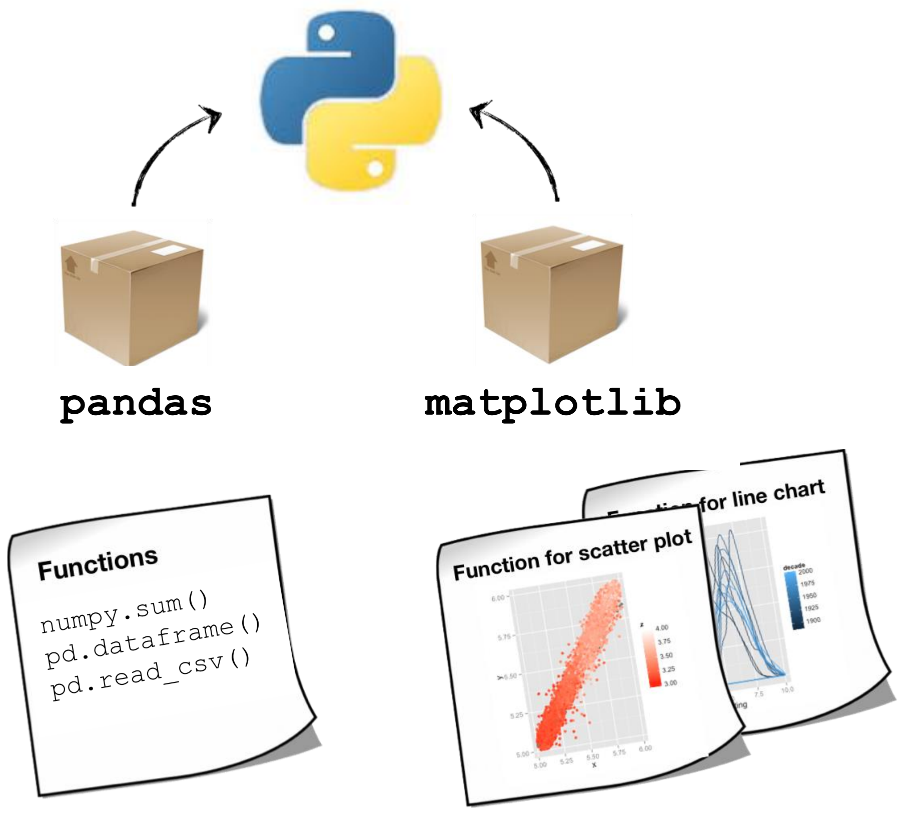
Mach mit

https://www.meetup.com/de-DE/Python-for-Data-Science-and-Machine-Learning-Zurich/ and https://www.meetup.com/de-DE/pyzurich/

Why is Python so popular?
There are many reasons…

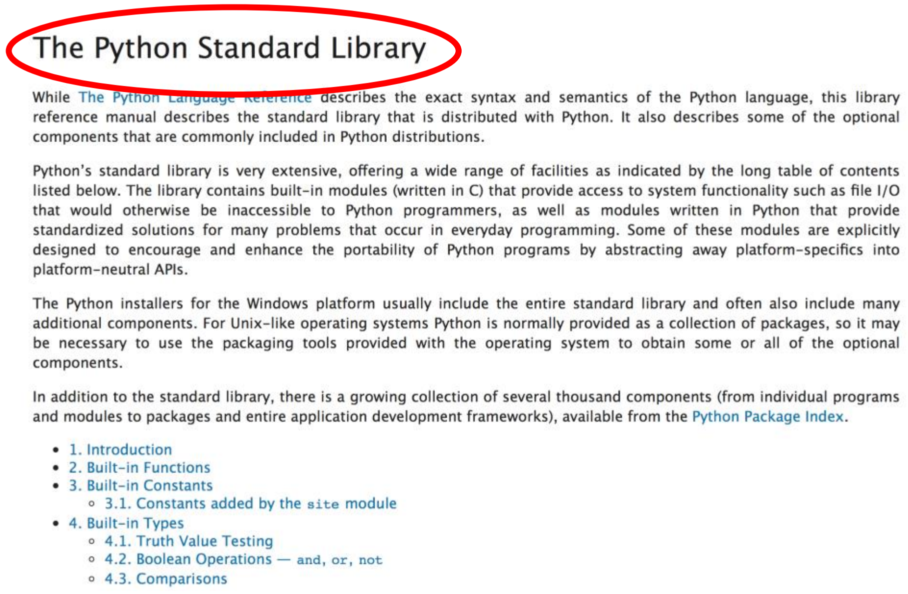
- Python is open source and free.
- Python code is relatively easy to read.
- Python runs on all operating systems (Windows, Mac OS X, Linux, Unix).
- Python is easily extensible via user-developed packages.
- Python is scalable.
- Analyses done using Python are reproducible.
- Using Python makes collaboration easier.
- Finding answers to questions is simple as the Python community is very helpful.

Why is Python so popular?
Python packages extend the functionality of Python



- **Packages** are collections of Python functions, classes, and data in the form of compiled code with a well-defined format.
- Most available packages are installed from the **Python Package Index (PyPI)** which is a repository of software for Python: https://pypi.python.org/pypi
- In total, more than 269’233 packages are available for the Python programming language (10/2020).

Why is Python so popular?
On the downside, not all packages are well documented



The Python Standard Library

While The Python Language Reference describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

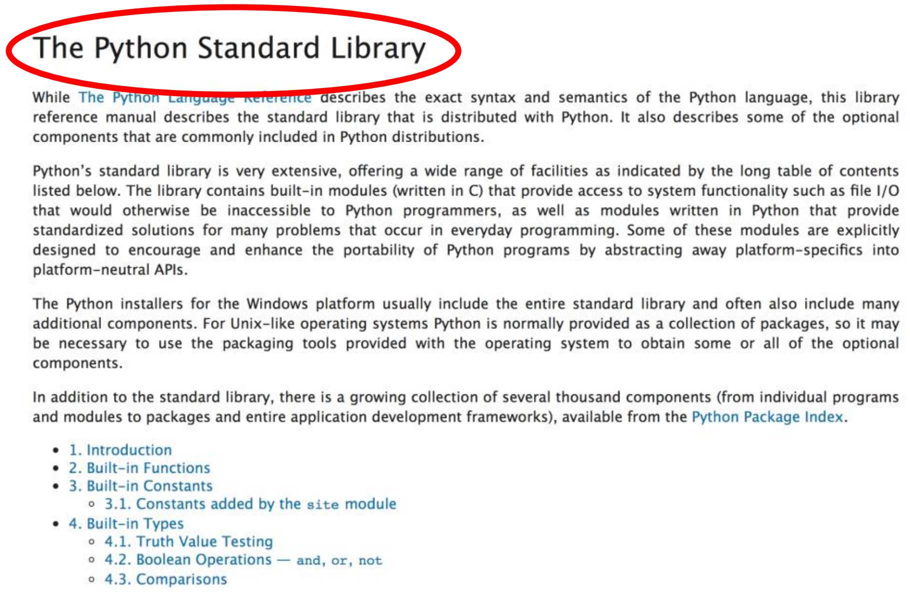
The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the Python Package Index.

- 1. Introduction
- 2. Built-in Functions
- 3. Built-in Constants
  - 3.1. Constants added by the `site` module
- 4. Built-in Types
  - 4.1. Truth Value Testing
  - 4.2. Boolean Operations — `and`, `or`, `not`
  - 4.3. Comparisons

- Documentation for the Python standard library can be found here: https://dos.python.org/3/library/ .
- Documentation for other packages can easily be found in PyPI or Google. However, for many packages, documentation is fragmented and incomplete compared to R.
- Formats and conventions for documentation are not standardized. A good orientation is the „Python Style Guide" (https://www.python.org/dev/peps/pep-0008/).

Why is Python so popular?
On the downside, not all packages are well documented



The Python Standard Library

While The Python Language Reference describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

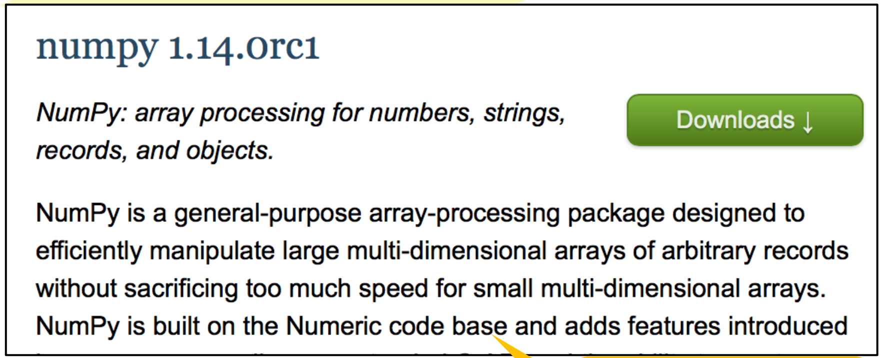
The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the Python Package Index.

- Documentation for the Python standard library can be found here: https://dos.python.org/3/library/ .
- Documentation for other packages can easily be found in PyPI or Google. However, for many packages, documentation is fragmented and incomplete compared to R.
- Formats and conventions for documentation are not standardized. A good orientation is the „Python Style Guide" (https://www.python.org/dev/peps/pep-0008/).

Basics of documentation in Python

1) Package documentation



numpy 1.14.0rc1

NumPy: array processing for numbers, strings, records, and objects.

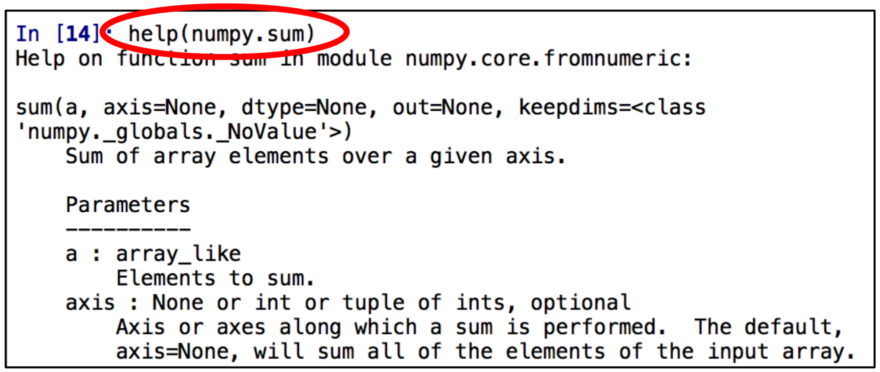
NumPy is a general-purpose array-processing package designed to efficiently manipulate large multi-dimensional arrays of arbitrary records without sacrificing too much speed for small multi-dimensional arrays. NumPy is built on the Numeric code base and adds features introduced

Extract from a package description in PyPI

README file:

- Describes the purpose of the project or library. Main entry point for readers.
- Format: raw text or markup language
- Typically displayed as module description on the PyPI page ([https://pypi.python.org/)](https://pypi.python.org/)

2) Documentation of modules and functions



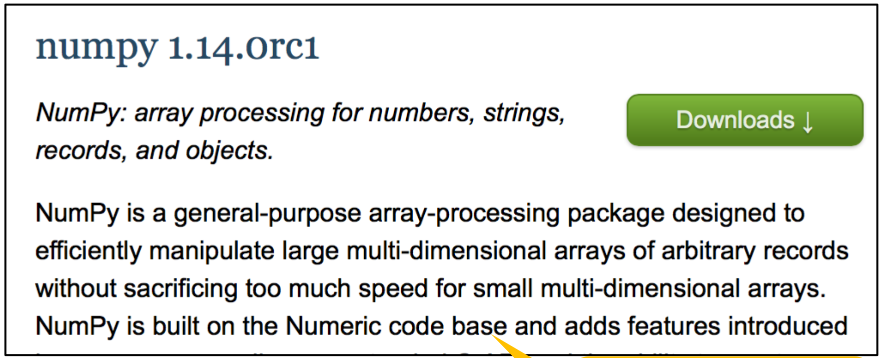
```
In [14]: help(numpy.sum)
Help on function sum in module numpy.core.fromnumeric:

sum(a, axis=None, dtype=None, out=None, keepdims=<class
'numpy._globals._NoValue'>)
    Sum of array elements over a given axis.

    Parameters
    ----------
    a : array_like
        Elements to sum.
    axis : None or int or tuple of ints, optional
        Axis or axes along which a sum is performed.  The default,
        axis=None, will sum all of the elements of the input array.
```

- All modules and functions should have a string literal describing itself (=docstring).
- They can be assessed via

  `help(packagename.functionname)`

Basics of documentation in Python

1) Package documentation


numpy 1.14.0rc1

NumPy: array processing for numbers, strings, records, and objects.

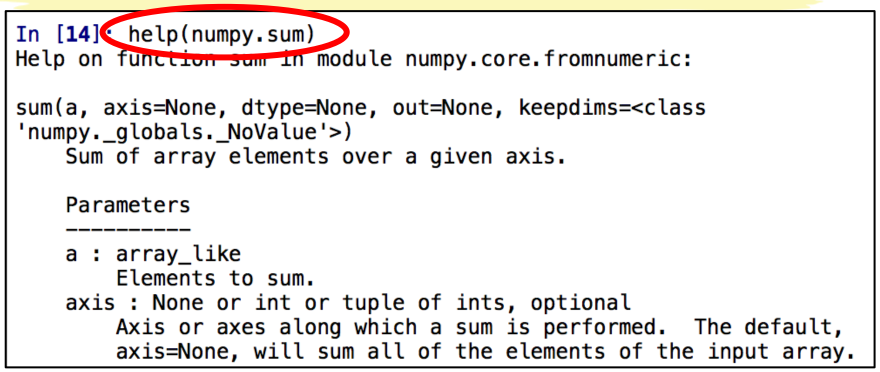
NumPy is a general-purpose array-processing package designed to efficiently manipulate large multi-dimensional arrays of arbitrary records without sacrificing too much speed for small multi-dimensional arrays. NumPy is built on the Numeric code base and adds features introduced

Extract from a package description in PyPI

README file:

- Describes the purpose of the project or library. Main entry point for readers.
- Format: raw text or markup language
- Typically displayed as module description on the PyPI page (https://pypi.python.org/)

2) Documentation of modules and functions


```
In [14]: help(numpy.sum)
Help on function sum in module numpy.core.fromnumeric:

sum(a, axis=None, dtype=None, out=None, keepdims=<class
'numpy._globals._NoValue'>)
    Sum of array elements over a given axis.

    Parameters
    ----------
    a : array_like
        Elements to sum.
    axis : None or int or tuple of ints, optional
        Axis or axes along which a sum is performed.  The default,
        axis=None, will sum all of the elements of the input array.
```

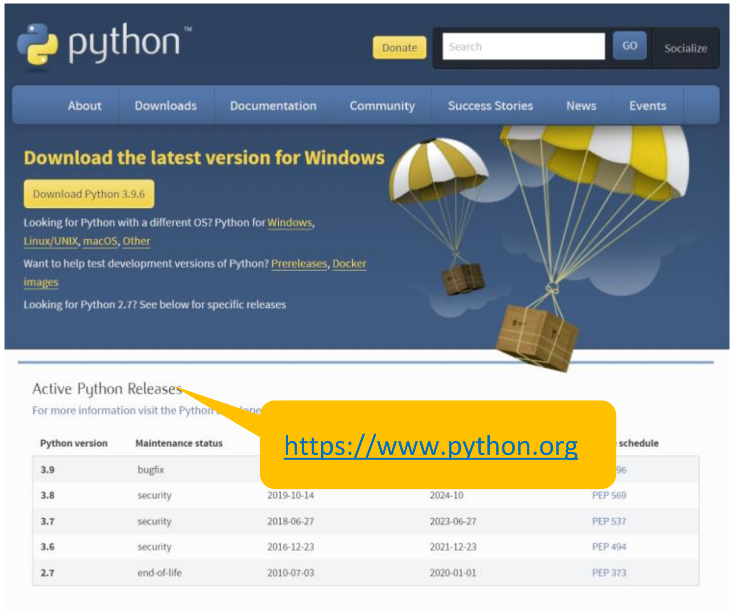
- All modules and functions should have a string literal describing itself (=docstring).
- They can be assessed via

  `help(packagename.functionname)`

http://docs.python-guide.org/en/latest/writing/documentation/

So how to use Python?

Distributions for local installation



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

Active Python Releases

For more information visit the Python

Python version	Maintenance status			schedule
3.9	bugfix	[illegible]	[illegible]	96
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
3.6	security	2016-12-23	2021-12-23	PEP 494
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

https://www.python.org

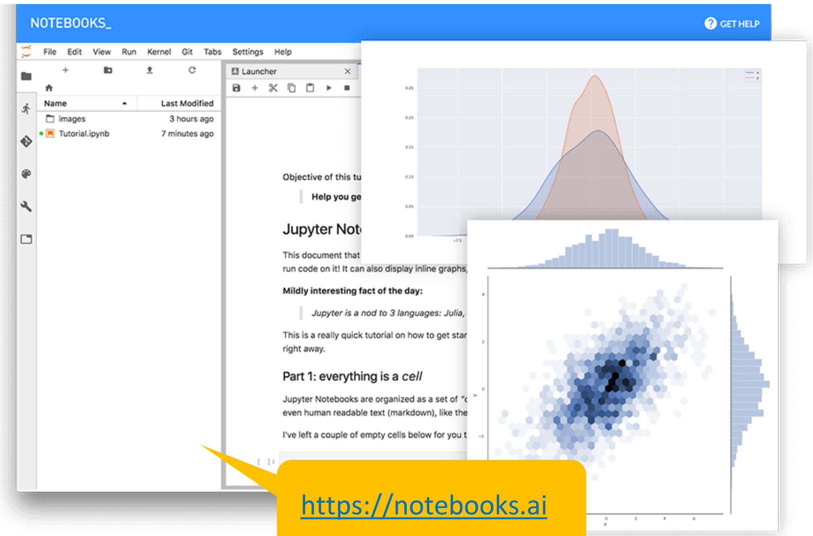
https://www.anaconda.com

ANACONDA®

Cloud services

https://colab.research.google.com

Google colab



Name	Last Modified
images	3 hours ago
Tutorial.ipynb	7 minutes ago

Objective of this tu

Help you ge

Jupyter Not

This document that
run code on it! It can also display inline graphs,

Mildly interesting fact of the day:

Jupyter is a nod to 3 languages: Julia,

This is a really quick tutorial on how to get star
right away.

Part 1: everything is a *cell*

Jupyter Notebooks are organized as a set of "c
even human readable text (markdown), like the

I've left a couple of empty cells below for you t

https://notebooks.ai

Getting started with Python