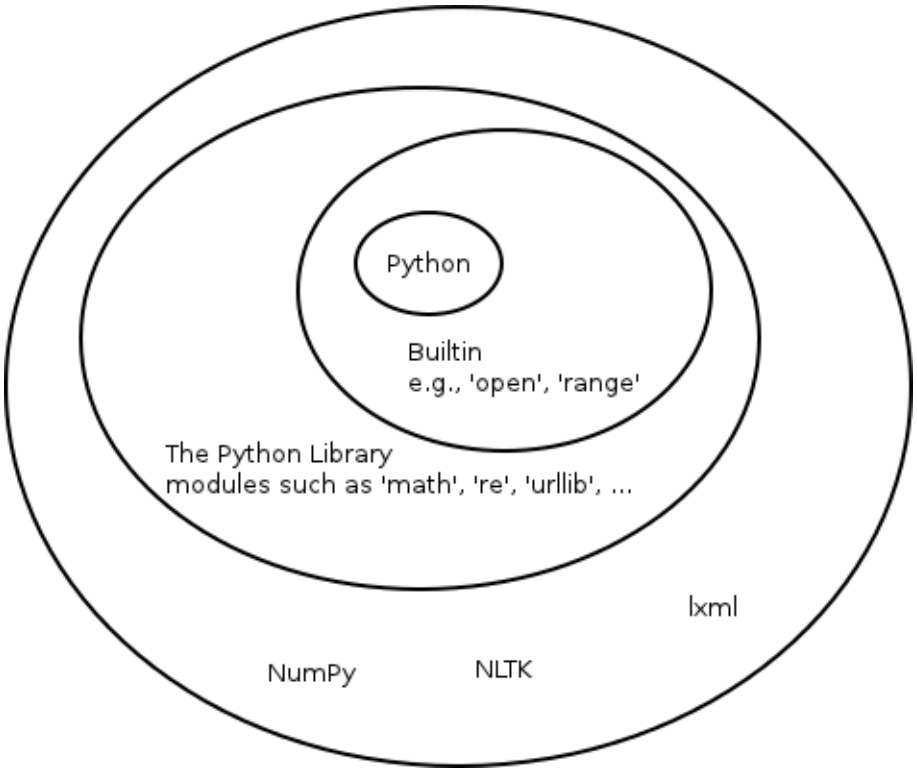
Python programming — installation

Finn Årup Nielsen

DTU Compute
Technical University of Denmark

August 31, 2014

Overview of Python parts



Apart from Python "itself":

Builtins that include basic operations: file loading, construction of types, etc. They are available in a special module available with (available with `>>> dir(__builtins__)`). They are automatically loaded when you start python

The Python Standard Library: bundled modules, such as "copy", "string", "re", "urllib". Write `import` *module name* to use them.

Other libraries, such as NLTK, NetworkX, lxml and NumPy. You need to install them and write `import` *module name* to use them.

Installation of Python

Python can be installed from:

http://www.python.org/download/

Latest Python 2 (Python 2.7) are most relevant, but if you have Python 3 that is ok, since most data mining packages are now also running under that version.

On Linux and Mac there are already packages in the distribution so you need not install it from that page.

Installation on Debian-like systems

On an Debian/Ubuntu-like system it is straightforward, e.g., with:

```
aptitude search python
sudo aptitude install python
```

You can install further packages with, e.g.:

```
sudo aptitude install python-nltk spyder
```

This will setup the Linux distribution version of `nltk`, a Python package for natural language processing, and spyder, an integrated development environment for Python.

Installation on Mac

On Mac an (old?) version might already be installed:

Start a terminal `Applications / Utilities / Terminal` and write

```
python
```

Or on a terminal, e.g., do:

```
port search python26
sudo port install python26
```

Or follow the installation instructions on MacPython:

http://homepages.cwi.nl/~jack/macpython/macpython-osx.html

Installation on Microsoft Windows

Besides installing from the Python download homepage there are different packages that assemble Python with libraries, editors, . . . ready for installation as one "unit":

Python(x,y): "Scientific-oriented Python Distribution based on Qt and Spyder" containing Python, numpy, pip, matplotlib, PyQt4 (for GUI development) and a lot of other packages.

winpython: "Portable Scientific Python 2/3 32/64bit Distribution for Windows"

With basic Python you need to be able to compile C-programs for C-code in the Python package, — unless you install precompiled version, see, e.g., http://www.lfd.uci.edu/~gohlke/pythonlibs/.

Commercial environments

ActivePython at http://www.activestate.com/activepython. There is a free "Community Edition".

Enthought Canopy, requires the request of an academic license

Anaconda. There is a "free enterprise-ready Python distribution".

Such system usually comes with interactive prompt, editor syntax highlighting, package manager, . . .

Community/academic version may lack, e.g., database modules and may not be free for commercial applications.

See a few further packages on the Python download page

Other "pythons"

bpython — Interactive shell with help and pastebin (mostly for Linux)

ipython — Enhanced interaction, mimicks Matlab. "`ipython -pylab`"

Eric and Spyder — Integrated development environments. Spyder especially for numerical Python.

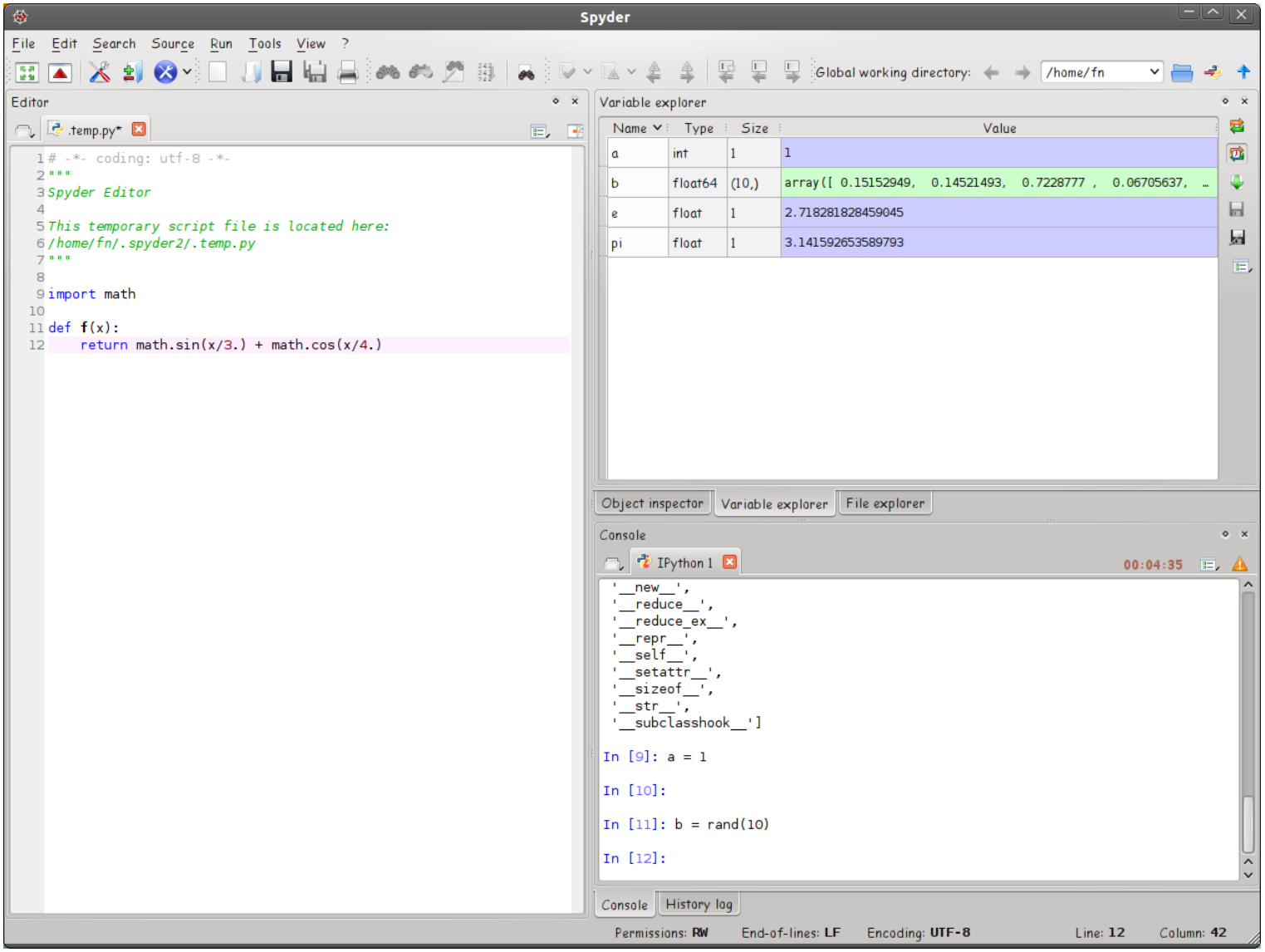
idle — Not so fancy GUI editor

PyPE — Python editor

Other programs to run a python program: IronPython (.Net), Jython (JVM) and PyPy

See also: http://wiki.python.org/moin/IntegratedDevelopmentEnvironments

Spyder


Spyder
File Edit Search Source Run Tools View ?
Global working directory: /home/fn
Editor
.temp.py*
-*- coding: utf-8 -*-
"""
Spyder Editor
This temporary script file is located here:
/home/fn/.spyder2/.temp.py
"""
import math
def f(x):
return math.sin(x/3.) + math.cos(x/4.)
Variable explorer
Name Type Size Value
a int 1 1
b float64 (10,) array([0.15152949, 0.14521493, 0.7228777 , 0.06705637, …
e float 1 2.718281828459045
pi float 1 3.141592653589793
Object inspector Variable explorer File explorer
Console
IPython 1
00:04:35
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__self__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__']
In [9]: a = 1
In [10]:
In [11]: b = rand(10)
In [12]:
Console History log
Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 12 Column: 42

. . . Spyder

Spyder has (or calls):

Editor with syntax highlighting, and indentation, interactive code analysis

Interactive Python shell (python or ipython)

Debugger and breakpoints via editor

Profiler to evaluate speed of the code

Style checking via pylint

IPython Notebook

Web-based cell-based ‘notebook’ interface.

Cell may be Python code, text output, plots, documention in e.g., Markdown.

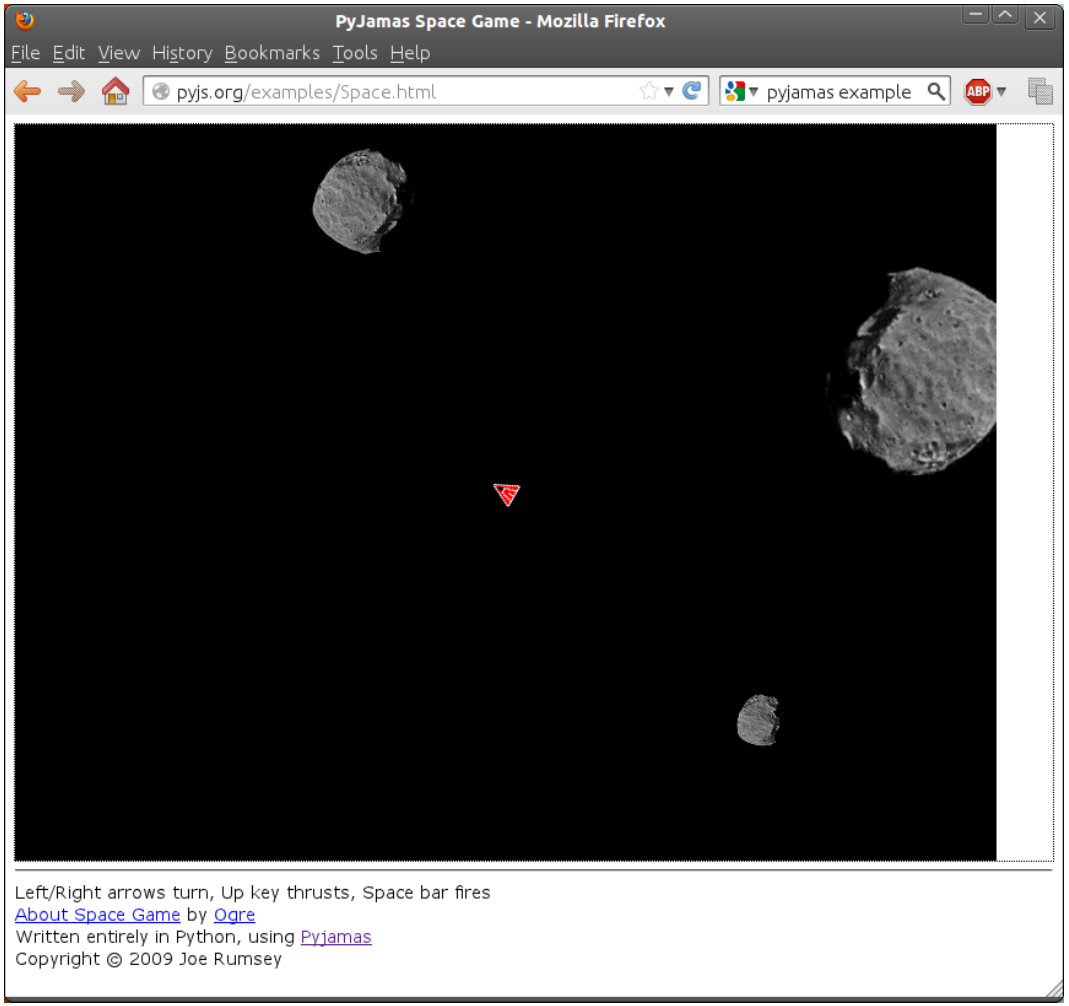
Individual code cells may be executed in random order.

IPython Notebook files are JSON file that may be shared on the Internet.

Great for interactive data mining.

Perhaps less interesting for traditional software development with reusable modules and testing.

Pyjamas



The Pyjamas tool converts a Python program into Javascript and HTML.

Asteroids game with canvas. The source code is available for Space.py

Other widgets: Buttons, labels, grids, . . .

See http://pyjs.org/examples/

Applications may also run as “desktop” via pyjd.

Packages on UNIX-derived systems

```
python-beautifulsoup
python-biopython
python-cherrypy
python-feedparser
python-imaging
python-networkx
python-nltk
python-nose
python-numpy
python-pysqlite2
python-scipy
python-simplejson
python-sklearn
```

and quite a number of other packages. . .

Installing packages outside distribution

`pip` (and `easy_install`) is a program that downloads package from central archive (or installs packages downloaded locally)

```
$ pip search nltk
nltk                      - Natural Language Toolkit
  INSTALLED: 2.0b8
  LATEST:    2.0b9
```

`pip install` manages dependencies to other packages.

To install pip you may need to use `easy_install`.

`pip` may uninstall. `easy_install` cannot do that.

PyPI

PyPI — the Python Package Index (the ‘cheeseshop’)

Central archive of distributed Python packages.

http://pypi.python.org/pypi

Used by `easy_install` and `pip`.

“There are currently 48140 packages here.” (of varying degree of quality)

Downloading packages from developers

Numpy:

http://sourceforge.net/projects/numpy/files/

NTLK:

http://nltk.org/install.html

. . . and so on.

This is only if you want to be on the blending edge.

Using setup.py

More basic if everything else does not work:

Often packages contain a `setup.py` file which may install the package:

```
python setup.py install
```

You can create distribution package of your own by setting up a `setup.py` file in the directory your python program to distribute and then call:

```
python setup.py bdist
```

which generate (in this particular case) a tar.gz package:

```
dist/helloworld-1.0.linux-i686.tar.gz
```

setup.py example

```
from distutils.core import setup

setup(name='helloworld',
      version='1.0',
      description='Simple Hello, World program',
      author='Finn Aarup Nielsen',
      author_email='fn@imm.dtu.dk',
      url='http://www.imm.dtu.dk/~fn/',
      license='GPL',
      scripts=['helloworld.py'],
     )
```

The package contain a single script `helloworld.py`:

```
#!/usr/bin/env python
print("Hello, World\n")
```

Python packaging

Note there are several different Python packaging systems

distutils (the old simple way)

setuptools (recommended for building to Python Package Index, PyPI)

distribute (deprecated in newest versions, moved to setuptools 0.7)

distutils2

distlib

bento

...

virtualenv

`virtualenv` is a Python module that allows you to have multiple installation of Python on your computer, e.g., a production and a development installation.

This is important in some situation — such as web-serving — where you want to "freeze" the installation.

On Ubuntu install the python package:

```
aptitude install python-virtualenv
```

Create a new virtual environment:

```
python /usr/share/pyshared/virtualenv.py virtualenv1
```

You now have python in `./virtualenv1/bin/python`

. . . virtualenv

```
$ cd virtualenv1/
$ which python
/usr/bin/python
$ source bin/activate        # This will change the path
$ which python
/home/fn/virtualenv1/bin/python
```

With pip:

```
$ pip --environment virtualenv1 wikitools
```

The “wikitools” package now got into

```
~/virtualenv1/lib/python2.6/site-packages/wikitools/
```

. . . virtualenv

You can “freeze” the installation. Recording the modules and their version into a requirement file with `pip`

```
python /usr/share/pyshared/virtualenv.py --no-site-packages virtualenv1
cd virtualenv1
source bin/activate
pip freeze > requirements.txt
cat requirements.txt
```

The content of this requirements.txt file is now:

```
argparse==1.2.1
distribute==0.6.24
wsgiref==0.1.2
```

. . . virtualenv

Installation of a module in the virtual environment:

```
$ pip install simplejson
$ pip freeze > requirements.txt
$ cat requirements.txt
argparse==1.2.1
distribute==0.6.24
simplejson==3.3.0
wsgiref==0.1.2
$ python
>>> import simplejson
>>> simplejson.__version__
'3.3.0'
```

. . . virtualenv

Now move the code to a new virtual environment (`virtualenv2`):

```
$ deactivate            # get out of the virtual environment
$ cd
$ python /usr/share/pyshared/virtualenv.py --no-site-packages virtualenv2
$ cd virtualenv2
$ source bin/activate
$ pip install -r ../virtualenv1/requirements.txt
$ python
>>> import simplejson
>>> simplejson.__version__
'3.3.0'
```

Now the simplejson module is available in the second virtual environment.

Python in the cloud

Web sites that allows you to run a Python program from their computers:

Google App Engine (GAE), persistency with Googles approach.

Pythonanywhere, e.g., with Flask web framework and MySQL.

Others: Heroku, Plotly and (PiCloud and StarCluster).

For GAE you need to download an SDK, whereas for Pythonanywhere you can start coding when you have setup the account.

Summary

There are different layers of Python: the python language itself, the Standard Library, other libraries.

The easiest way of installation is probably in Linux with aptitude/apt-get

Install further libraries not in the distribution with pip.

There are a number of other Python environments: ipython (interactive), spyder (IDE), pypy (JIT, sandbox), . . .

Commercial Python environments could be considered: Enthought, ActivePython.

Consider virtualenv with pip for ‘real’ deployment.