Why Python?

Eine kurze Einführung

Gerald Senarclens de Grancy, gerald@senarclens.eu

Outline

1. Features
2. Tools
3. Basic Syntax
4. Sample Applications
5. Interfaces

Features

- Open Source
- Multi paradigm scripting/ programming language
- Platform independent
  - *Linux*
  - *Unix*
  - *Mac*
  - *Windows*
  - *Amiga*
  - *BeOS*
  - *Win CE*
  - *Dos*
  - *QNX*
  - *Psion Series 5*
  - *OpenVMS*
  - *VxWorks*
  - *environments providing a Java virtual machine*
  - *...*
- Simple and elegant syntax

- ***Very easy to learn***
- *Focus on task, not language*
- *Great for rapid prototyping and TDD*

- Batteries included
  - *Provides high level data types*
  - *Extensive standard library*
- Easily extensible - third party modules
  - *Numeric and symbolic calculations*
  - *Audio post processing*
  - *Image processing*
  - *System administration*
  - *Web development*
  - *GUI development*
  - *...*
- Huge Community

Tools

- Standard Python Software
  - *Python Shell (``play'' around, test statements and get help)*
  - *Python Debugger*
  - *IDLE*
- GNU Emacs Python mode
- IPython
  - *tab completion*
  - *help (?), shell interface (!), magic functions (%) - eg %edit, %cpaste*
- pdb - The Python Debugger
  - *h(elp)*
  - *l(ist) [first [,last]]*
  - *b(reak) ([file:]lineno | function) [, condition], cl(ear) [bpnumber [bpnumber...]]*
  - *run (restart)*
  - *s(tep), n(ext), r(eturn), c(ontinue)*
- nosetests

- *extends unittest to make testing easier*

- Many different IDEs and GUI designers
  - *Eclipse with PyDev*
  - *KDevelop with kdev-python*
  - *Eric*
  - *...*

Basic IO

Writing text to a console

spam.py

```
print("Spam and eggs")
```

Launching the program

```
$ python spam.py
```

Basic IO

Asking for input

dialog.py

```
#!/usr/bin/env python
food = raw_input("What do you want to eat? - ")
print("We don't have %s, we only have spam and eggs." % food)
```

Make the program executable and launch it as shell script

```
$ chmod u+x dialog.py
$ ./dialog.py
```

Basic IO

- Read a file
- Print its content

read-file.py

```
#!/usr/bin/env python
with open("data/history.txt", 'r') as hist_file:
    content = hist_file.read()
print(content)
```

Flow Control

- Blocks are delimited by indentation (4 spaces) and started by a colon
- Batteries included
- Example: throwing a coin

coin.py

```
#!/usr/bin/env python
from random import random
if random() >= 0.5:
    print("head")
else:
    print("tail")
```

High-level Types

- Python provides Files, Mappings, Sequences, Set types, ...
- Types offer operators and methods
- Example: sort the content of a file

sort-file.py

```
#!/usr/bin/env python
with open("data/history.txt", 'r') as hist_file:
    lines = hist_file.readlines()
lines.sort()
print(lines)
```

Loops

- `for elem in sequence` - iterate over elements in a sequence type
- Example: print a file line by line

for.py

```
#!/usr/bin/env python
with open("data/history.txt") as hist_file:
    lines = hist_file.readlines()
lines.sort()
for line in lines:
    print(line.strip())
```

Loops

- `while` - repeat a block as long as an expression is true
- Example: repeat lottery until you lose

while.py

```
#!/usr/bin/env python
from random import random
win_count = 0
while(random() >= 0.5):  # head
    win_count += 1
print("head won %d times" % win_count)
```

Functions

- Portion of code within a larger program
- Performs a specific task
- Relatively independent of the remaining code
- Example: sum a list of numbers

sum.py

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


def sum(*numbers):
    total = 0
    for num in numbers:
        total += num
    return total

print(sum(3, 5, 7.9))
```

Objects and Classes

- Python supports object oriented programming
- Objects are data structures
- Objects consist of data fields and methods and their interactions
- Example: pick a student from a list

random_student.py

```
#!/usr/bin/env python
from random import choice


class StudentPicker(object):
    def __init__(self, students):
        with open(students, 'r') as student_file:
            students = student_file.readlines()
        students = filter(None, students)
        students = [s.strip().split('\t') for s in students]
        self._students = ["%s %s, %s" % (s[0], s[1], s[2]) for s in students]
        self._initial_students = list(self._students)

    def pick_student(self):
        """
        Return a student and remove that student from the list.
        """
        try:
            student = choice(self._students)
            self._students.remove(student)
            return student
        except IndexError:
            return "no more students available; use 'reset' or 'quit'"

    def reset(self):
        self._students = list(self._initial_students)
```

Sample Applications

- Perform minor number crunching
- Example: matrix multiplication

matrix.py

```
#!/usr/bin/env python
import numpy as np
A = np.array((0, 1, 0, 2, 1, 3, 1, 0, 1)).reshape((3, 3))
B = np.ones(9).reshape((3, 3))
print(np.dot(A, B))
```

Sample Applications

- Draw nice graphs
- Requires matplotlib and numpy
- Example: plot a probability density function (gaussian)

pdf.py

```
#!/usr/bin/env python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab

mu, sigma = 0.0, 1.0

fig = plt.figure()
ax = fig.add_subplot(111)
x = np.arange(-4, 4.1, 0.1)
y = mlab.normpdf(x, mu, sigma)
l = ax.plot(x, y, 'r-', linewidth=1)

ax.set_title(r'probability density function: $\mu=%f,\ \sigma=%f$'
    % (mu, sigma))
ax.set_xlim(-4, 4)
ax.set_ylim(0, 0.5)
# plt.show()
plt.savefig("data/density.svg")  # also supports pdf, png, ...
```

Interfaces

- Shell

shell.py

```
#!/usr/bin/env python
import cmd
import sys
from random_student import StudentPicker


class StudentPickerShell(cmd.Cmd):
    def __init__(self, students):
        cmd.Cmd.__init__(self)                # initialize the base class
        self.prompt = "ST> "
        self._picker = StudentPicker(students)

    def do_quit(self):
        sys.exit()

    def do_pick(self):
        print(self._picker.pick_student())

    def do_reset(self):
        self._picker.reset()

shell = StudentPickerShell("data/pioneers.tsv")
shell.cmdloop()
```

Interfaces

- QT

qt.py

```
#!/usr/bin/env python
import sys
from PyQt4 import QtGui, QtCore
from random_student import StudentPicker


class StudentPickerWindow(QtGui.QWidget):
    def __init__(self, students, parent=None):
        self._picker = StudentPicker(students)
        QtGui.QWidget.__init__(self, parent)
        self.setWindowTitle('Student Picker')

        pick = QtGui.QPushButton("Pick")
        reset = QtGui.QPushButton("Reset")
        textfield = QtGui.QLineEdit()
        self._textfield = textfield

        grid = QtGui.QGridLayout()
        grid.setSpacing(10)
        grid.addWidget(textfield, 1, 0)
        grid.addWidget(pick, 1, 1)
        grid.addWidget(reset, 1, 2)

        self.connect(pick, QtCore.SIGNAL('clicked()'), self.do_pick)
        self.connect(reset, QtCore.SIGNAL('clicked()'), self.do_reset)
        self.setLayout(grid)
        self.resize(500, 50)

    def do_pick(self):
        self._textfield.setText(self._picker.pick_student())

    def do_reset(self):
        self._picker.reset()
        self._textfield.setText("")

if __name__ == "__main__":
```

```
    app = QtGui.QApplication(sys.argv)
    window = StudentPickerWindow("data/pioneers.tsv")
    window.show()
    sys.exit(app.exec_())
```

Interfaces

- Web-interface
- Django-based
- Example: plot 2D functions

Further Reading

Guido van Rossum, Barry Warsaw
PEP 8 -- Style Guide for Python Code
http://www.python.org/dev/peps/pep-0008/

Mark Pilgrim
Dive Into Python 3 (2nd edition)
Apress (October 23, 2009)

Python Software Foundation
Python Documentation
http://docs.python.org/

The following list contains all external links in order of appearance in the presentation

- Gerald Senarclens de Grancy,: http://www.senarclens.eu/~gerald/
- Open Source: http://www.opensource.org/
- What: http://www.python.org/community/jobs/
- about: http://www.careerjet.at/
- Standard Python Software: http://www.python.org/download/
- Python Debugger: http://www.python.org/doc/current/lib/module-pdb.html
- IDLE: http://www.python.org/idle/
- GNU Emacs Python mode: http://www.python.org/emacs/python-mode/

- IPython: http://ipython.scipy.org/
- Eclipse: http://www.eclipse.org/
- PyDev: http://pydev.org/
- KDevelop: http://kdevelop.org/
- kdev-python: http://scummos.blogspot.com/
- Eric: http://eric-ide.python-projects.org/
- Files, Mappings, Sequences, Set types, ...:
  http://docs.python.org/library/stdtypes.html
- PEP 8 -- Style Guide for Python Code:
  http://www.python.org/dev/peps/pep-0008/
- Dive Into Python 3: http://getpython3.com/diveintopython3/
- Python Documentation: http://docs.python.org/
- help?: http://www.w3.org/Talks/Tools/Slidy2/help/help.html