Python

Tutorial for Programmers

Gerald Senarclens de Grancy

Opera Software
S:T Larsgatan 12
58224 Linköping, Sweden

April 21, 2009

Outline

Features

- Open Source
- Object oriented scripting/ programming language
- Type declaration not necessary
- Completely dynamic language
- Platform independent (Windows, Unix, Linux, Mac, Amiga, BeOS, Win CE, Dos, QNX, Psion Series 5, OpenVMS, VxWorks and other environments providing a Java virtual machine)
- Provides high level dynamic data types
- Simple and elegant syntax (very easy to learn)

Features

- Extensive standard libraries and third party modules
- Extensions and modules easily written in C, C++ (or Java (Jython) and .NET (IronPython))
- Embeddable within applications as a scripting interface

Features

- Extensive standard libraries and third party modules
- Extensions and modules easily written in C, C++ (or Java (Jython) and .NET (IronPython))
- Embeddable within applications as a scripting interface

$\Rightarrow$ Huge Community

Features

- Extensive standard libraries and third party modules
- Extensions and modules easily written in C, C++ (or Java (Jython) and .NET (IronPython))
- Embeddable within applications as a scripting interface

$\Rightarrow$ Huge Community

$\Rightarrow$ Opera's official scripting language

Development Tools and Environments

- Standard Python Software
  - Python Shell ("play" around, test statements and get help)
  - Python Debugger
- GNU Emacs Python mode
- IDLE
- IPython
- Many different IDEs and GUI designers

Launching a program

```
$ python program.py
```

Launching a program

```
$ python program.py
```

as shell script

```
#!/usr/bin/env python
$ chmod u+x program.py
```

Launching a program

```
$ python program.py
```

as shell script

```
#!/usr/bin/env python
$ chmod u+x program.py
```

by importing it (in the python shell or another program)

```
import program
```

Launching a program

```
$ python program.py
```

as shell script

```
#!/usr/bin/env python
$ chmod u+x program.py
```

by importing it (in the python shell or another program)

```
import program
```

or using an integrated development environment

Flow Control

Truth tests: empty enumerations and 0 are considered False:

```
[], {}, (), 0, 0.0, None, False
```

Blocks are delimited by indentation and started by a colon

if

if test:
    suite
[**elif** test:
    suite]
[**else:**
    suite]

Flow Control

while

while test:
    suite
[**else:**
    suite]

Flow Control

while

while test:
    suite
[**else:**
    suite]

for

for elem **in** sequence:
    suite
[**else:**
    suite]

Flow Control

while

while test:
    suite
[**else:**
    suite]

for

for elem **in** sequence:
    suite
[**else:**
    suite]

range(...)

```
range([start,] stop[, step]) -> list of integers
```

Flow Control

- `pass` (empty statement, like ";" in Java)
- `break` (continues execution after the innermost loop)
- `continue` (jumps to the next iteration of the innermost loop)
- `return` (leaves the current function returning a value or `Null`)
- no `switch case`
  can be done w/ repeated elifs or by indexing dictionaries

Advanced Data Types

string (immutable)

"Python's", r'Python"s raw', u"""multiline unicode"""
special letters: %, \n, \t, \\,...
strings are concatenated automatically
basic methods on string s

```
s.strip(), s.lstrip()
s.find(sub [, start [, end]])
s.rfind(sub [, start [, end]])
s.replace(old, new [, maxtimes]
s.capitalize(), s.swapcase(), s.islower()
...
```

Advanced Data Types

tuple (immutable)

```
(), (3), (1, 6, 2), ('a', 4, 'string', 34.876, (3, 1))
```

Advanced Data Types

tuple (immutable)

```
(), (3), (1, 6, 2), ('a', 4, 'string', 34.876, (3, 1))
```

list (mutable)

```
[], [3], [1, 6, 2], ['a', 4, 'string', [[1], (3, 1)], (1, [2])]
```

basic methonds on list l

```
l.append(element), l.insert(i, element)
l.remove(element)
l.index(element), l.sort([function])
l.count()
l.pop([i])
...
```

Advanced Data Types

sequence operations

concatenation +
repetition *
indexing `seq[i]` (`l[0]`, `s[4]`)
slicing `seq[start:end]` (`t[1:3]`, `l[:]`, `s[:-3]`)

```
elem in seq, elem not in S
for elem in seq
len(seq)
min(seq), max(seq)
```

Advanced Data Types

dict(ionary) (mutable)

```
{}, {'spam': 1, 'egg': 5}, {'d': 'Ed':1, 'l':[1, 2] }
```

basic methods on dictionary d

```
d.keys(), d.values(), d.has_key(key)
d.copy(), d.clear()
d.update(dictionary)
d.get(key, [default]), d.setdefault(key, [default])
d.popitem()
...
```

Advanced Data Types

file

```
# read
file = open('filename', 'r')
file.read(), file.read(n)
file.readline(), file.readlines(), file.xreadlines()
# write
file = open('filename', 'w')
file.write(string), file.writelines(stringlist)
# basic methods on file f
file.closed, file.mode, file.name
file.close(), file.tell()...
```

Advanced Language Features

functions

```
def myFunction(arg1, arg2):
  return arg1, arg2
def myFunction2(arg='default value'):
  print arg
def myFunction3(*args):
  for arg in args:
      print arg
def myfunction4(**args):
      keys = args.keys()
      for key in keys:
        print args[key]
```

Advanced Language Features

object oriented features

```
class MyClass([Base1, Base2, ...]):
  static = 'salut'
  def __init__(self, arg):
      self.property = arg
  def __repr__(self):
      reprStr = 'property = ' + str(self.property) + '\n'
      reprStr += 'static = ' + str(self.static)
      return reprString
  def setProperty(self, newValue):
      self.property = newValue
  def getProperty(self):
      return self.property
myObject = MyClass(arg)
```

Advanced Language Features

exceptions

```
class MyError(Exception):
    def __init__(self, value):
        self.value = value
try:
    raise MyError, "no good"
except MyError:
    print "MyError was caught"
else:
    print "no exception occurred"
finally:
    print "this always happens"
```

Libraries

- `os` (operating system interaction)
  - concerning platform independence use eg. `os.sep`, `os.linesep`
  - the following command gives access to a system shell `os.system('shell command')`
- `sys` (everything about the Python system itself (eg. `sys.argv`)
- `pickle` (create portable serialized representations of Python objects) `pickle.dump(myObject, file)`, `object = pickle.load(file)`
- `string`, `re` (regular expressions), ...
- GUI: Tkinter, wxwidgets, GTK, QT, ...
  Java Swing (w/ Jython it's possible to access the complete Java API + all Java libraries)

for more information use the built-in help(...) function

Further Reading

- `http://www.python.org`
- `http://www.python.org/doc/` (official Python documentation)
- `http://www.python.org/doc/current/tut/tut.html` (the Python Tutorial)
- Newsgroups: comp.lang.python, comp.lang.python.announce
- ...