An introduction to Python for UNIX/C programmers

Guido van Rossum

Centrum voor Wiskunde en Informatica
Amsterdam

guido@cwi.nl

Python is...

- A VHLL (Very High Level Language)
- Simple
- Free
- Extensible
- Named after *Monty Python's Flying Circus*

The best of two worlds

Python provides the best of two worlds:

- dynamic scripting languages (Perl, Tcl)
- general programming (C/C++, Modula)

Hence it is usable for:

- scripts;
- rapid prototyping;
- serious software development;
- and as an extension language (embedded)

"Python is suitable for fast development of large programs, but also does well at throw-away shell coding" — Mark Lutz, XVT

It’s a scripting language

- Dynamic, interpretive, interactive
- No explicit compile or link steps
- No type declarations
- High level objects: lists, tuples, strings, associative arrays
- High-level operations: ‘in’, concatenation, etc.
- Automatic memory allocation/ deallocation (no ’pointers’)
- Programs can construct and execute program code using strings (exec/eval)
- Compact syntax
- Extensible in C/C++

It’s a general language

- Decent syntax:

```
while 0 <= y <= 5 and still_ok(x):
        if perfect(x, y): return x
        x, y = next_iteration(x, y)
```

- Modules:

```
import string
s = string.lower(s)
```

- Classes:

```
class Bag:
        def __init__(self): self.r = []
        def add(self, x): self.r.append(x)
```

- Exceptions:

```
try:
        x = 1/y
except ZeroDivisionError:
        print 'Sorry'
```

Extensibility

- Well-defined interface to and from C/ C++ functions and data
- Well-defined ways to add C/C++ modules to the system and language
- Dynamic loading of extensions (on some systems)
- Also usable as an *embedded language*, i.e. as extension language for large applications (like Lisp is for Emacs)

Interfaces

- Unix I/O and process system calls
- Sockets, TCP, UDP, Sun RPC
- FTP, NNTP, WWW, Gopher
- Window systems:
    - STDWIN (portable, not tied to X11)
    - Xt: Athena, Motif
    - SGI: GL, Forms
- Audio
- Images
- Many others library packages

Programming environment

- Interactive use: line editing, history
- Emacs editing mode
- Symbolic debugger (in Python; windowing version also available)
- Profiler (in Python)
- Stack trace on unhandled exception
- "Byte-compiled" modules managed automatically
- Dynamic loading of extension modules (on systems that support it, e.g. SunOS, SGI)

Distribution

Latest version: “0.9.9”; “1.0” soon

- Stable
  (new features released every 3-9 months)
- Not tied to Unix
  (Mac, DOS versions available)
- Complete: source, examples, tutorial, documentation
- Free, distributable, available via ftp
  (ftp.cwi.nl:/pub/python)
- Also on the NLUUG CD-ROM
- Mailing list: python-list-request@cwi.nl

Ancestry and influences

Eclectic design took ideas from:

- ABC (VHLL idea, VHL data types, statement syntax)
- C (tokens, expressions, 0-based indexing)
- C++ (classes, operator overloading)
- Modula-3 (modules, exceptions, methods)
- others (Icon, Algol-68, Pascal, fp, ...)

Example 1: directory listing

```
import os
names = os.listdir('.')
names.sort()
for n in names:
    if n[0] != '.':
        print n
```

Example 2: prime numbers

```
primes = [2]
print 2
i = 3
while len(primes) < 100:
    for p in primes:
        if i%p == 0 or p*p > i:
            break
    if i%p != 0:
        primes.append(i)
        print i
    i = i + 2
```

Example 3: phone database

```
import os, sys, string, regex

def main():
    pattern = string.join(sys.argv[1:])
    home = os.environ['HOME']
    filename = os.path.join(home,'.telbase')
    prog = regex.compile(pattern,
                         regex.casefold)
    f = open(filename, 'r')
    while 1:
        line = f.readline()
        if not line: break  # EOF
        if prog.search(line) >= 0:
            print string.strip(line)

main()
```

Example 4: *Motif* hello world

```
import sys, Xt, Xm

def main():
    toplevel = Xt.Initialize()
    button = toplevel.CreateManagedWidget(
                 'button', Xm.PushButton, {})
    button.labelString = 'Hello world'
    button.AddCallback(
                 'activateCallback', quit, 0)
    toplevel.RealizeWidget()
    Xt.MainLoop()

def quit(widget, clientdata, calldata):
    sys.exit(0)

main()
```

Example 5: classes

File Count.py:

```
class Counter:
    def __init__(self): # constructor
        self.value = 0
    def show(self):
        print self.value
    def incr(self, amount):
        self.value = self.value + amount
```

Usage:

```
import Count
my_counter = Count.Counter() # constructor
my_counter.show()
my_counter.incr(14)
my_counter.show()
```

Example 6: a C extension

```
/* File "demomodule.c" */

#include <stdlib.h>
#include <Py/Python.h>

static PyObject *demo_system(self, args)
    PyObject *self, *args;
{
    char *command;
    int sts;
    if (!Py_ParseArgs(args, "s", &command))
        return NULL;
    sts = system(command);
    return Py_BuildValue("i", sts);
}
```

(cont'd on next page)

A C extension (cont'd)

```
static PyMethodDef demo_methods[] = {
    {"system", demo_system},
    {NULL, NULL} /* Sentinel */
};

void PyInit_demo()
{
    (void) Py_InitModule("demo",
                         demo_methods);
}
```

Comparison to C

When to use C:

- massive data manipulations touching each byte/word of an array
- for utmost efficiency of production software

When to use Python:

- when code is mostly high-level
- for once-only programming
- when low-level operations can be mapped to built-in operations

Hint: use a hybrid approach!

Comparison to Perl

Disadvantages of Perl:

- steep learning curve
- encourages write-only code
- not well suited for large programs
- no nested data structures
- not easily embeddable

Disadvantages of Python:

- somewhat larger code (for readability)
- less powerful regular expressions
- less powerful output formatting

Comparison to Tcl

Tcl's main strengths are its extremely simple syntax and data types

This is also its main weakness; Tcl is not well suited for large programs

Tcl comes with "tk", an X-based GUI system

For GUI programming, python offers a choice:

- STDWIN (portable but too simplistic)
- Motif (X-based and complete but big and slow)
- Athena (X-based, keeps a middle ground)

Python in the real world

Interested companies:

- Sunrise Software (product!)
- Island Graphics
- XVT
- Angoss Software
- others...

Research projects using Python:

- MADE (multimedia; Bull, CWI et al.)
- U. of Virginia (virtual reality)
- CWI (hypermedia authoring)

The future

Release 1.0 scheduled for posting to comp.sources before end 1993

- bugs fixed
- compiles out-of-box on more platforms
- double-quoted string literals allowed
- functional programming support (lambda(), map(), bagof(), reduce())
- better naming conventions in C source
- improved speed and memory usage
- many minor improvements
- backwards compatible with 0.9.9