Python Crib-sheet

Iain Phillips

April 2000

Abstract

Summary of Python commands. Suitable for an experienced programmer, particularly one with knowledge of Perl and Unix/Linux. We ignore classes.

1 Invocation

To run the file `myprog`, type `python myprog`. Or if the first line is

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

and `myprog` is executable, then just type `myprog`.

2 Variables and assignment

```
name = 'Fred'
firstname,lastname = 'Fred','Bloggs'
```

There is no need to declare variables.

3 Control

```
if person == 'Alf':
  print person,ageof[person]
elif person != 'Sophie':
  print person
else:
  person = 'Joe'
```

Note indentation.

```
for x in ['a','b','c']:
  print x
for y in range(20):
  # 0..19
  print y
z = 5
while z>0:
  print z
  z = z-1
```

Breaking out of a loop:

```
for name in list:
  if name == 'Smith':
    break
  print name
```

4 Data

4.1 Strings

```
'abc'
```

Concatenation: `'abc'+'def'`
For multi-line strings use triple quotes.

```
'''This is a long sentence which goes
on beyond a single line.'''
```

Unlike in Perl, strings cannot contain variables:

```
name = 'Fred'
print 'My name is',name
# alternatively:
string = 'My name is '+name
print string
```

4.2 Lists

```
['a','b',1,2]
```

Indexing: `L[0]` (first element), `L[-1]` (last element)

Slices: `L[0:3]` from `L[0]` up to but not including `L[3]`

```
list = ['cat']
list.append('dog')
print len(list)
```

This will print 2.

4.3 Dictionaries

```
age = {}
age['Fred'] = 13
```

These are the same as hashes in Perl.

5 Types

Typing is dynamic. It is forbidden to form badly typed expressions. Unlike in Perl, one has to explicitly change types: Thus `3` is a number, and `'3'` is `3` as a character. `int('30')` is `30` (the number).

6 Sorting

Sorting is done in place for efficiency:

```
list = [1,4,3]
list.sort()
print list
```

NB it would be wrong to write

```
newlist = list.sort()
```

as `newlist` would then be the `none` object rather than the sorted list, which is still in `list`.

To sort a list by some attribute:

```
def byage(a,b):
  return cmp(ageof[a],ageof[b])
people = ['Alf','Joe','Karen']
ageof = {}
ageof['Alf'] = 12
ageof['Joe'] = 60
ageof['Karen'] = 40
people.sort(byage)
```

7 Functions

```
def myfunction(x,y):
  return x*y
print myfunction(2,3)
```

will print `6`.

Can have default arguments:

```
def fn(x,y=4):
  return fn(x*y)
fn(3)
```

will print `12`.

Functions can use variables from the main program, but if they can only modify them if first declared global:

```
name = 'Smith'
def titlename():
  global name
  name = 'Mr '+name
```

8 Modules

There are a number of built-in modules, the main ones being `sys`, `os`, `string`.

```
import os
os.mkdir('newdir',755)
time = os.popen('date').readline()
```

User-created modules should be in files named `*.py`. Suppose that the file `mymod.py` contains a function

```
def fn(a,b):
  print a+b
```

Then in another file we could have:

```
import mymod
mymod.fn('abc','def')
```

If `mymod.py` is in the same directory as the main program, then python will find it. Otherwise you will have to modify pythonpath. Suppose that your modules are stored in the directory `moddir`, with path `/.../moddir`.

```
import sys
sys.path.insert(0,'/../moddir')
import mymod
```

Or if you want to do this for all your programs, you can create a file `getmods` containing

```
import sys
sys.path.insert(0,'/../moddir')
```

Then start the main program with:

```
execfile('getmods')
import mymod
```

9 The string module

import string

```
list = string.digits
# so list = [0,1,2,3,4,5,6,7,8,9]
print string.join(list)
```

This will print `'0123456789'`.

```
str = 'cat dog canary'
list = string.split(str)
print string.strip('  cat dog  ')
```

This will print `cat dog` (without the leading and trailing spaces).

```
print string.replace('gala','a','b')
```

This will print `gblb`.

10 Input, Output and Files

To write to a new file:

```
file = open('newfile','w')
file.write('First line.\n')
file.write('Second line.\n')
file.close()
```

To read from an existing file:

```
file = open('existingfile')
for line in file.readlines():
  print line
```

To get access to the filename of the program:

```
import sys
filename = sys.argv[0]
```

11 CGI scripts

These will start with

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

If there is user input from a form (say there are name and age fields):

```
import cgi
form = cgi.FieldStorage()
name = form['name'].value
age = form['age'].value
```

If users have to give name and password:

```
user = os.environ['REMOTE_USER']
```

For debugging:

```
sys.stderr = sys.stdout
import traceback
try:
  ... (program)
except:
  print '<PRE>\n\n'
  traceback.print_exc()
```

This will print out on a web page whatever error message would normally appear at the command line.

12 Further information

See *Python Pocket Reference* by Mark Lutz (O'Reilly) or http://www.python.org.