Python documentation

Documents some Python[1] tips and tricks

Contents

1 How to process command line arguments

Use the optparse[2] standard module.

To disable interspersed arguments (i.e to stop option processing at the first non-option), call `disable_interspersed_args` on your parser.

2 Return value of enumerate

```
for (index, value) in enumerate (sequence):
    # ...
```

[1] http://python.org/

[2] http://docs.python.org/library/optparse.html

3 Enumerate in reverse

Christophe Simonis suggests using the itertools library[3] , but I prefer the non-optimized version (simpler):

```
for (index,value) in reversed(list (enumerate (sequence))):
    doStuff ()
```

4 Convert a string to an integer

Use the built-in function `int (string, [base])`.

If you know that your string is going to be in a base lower than 11, you can use the `str.isdigit` function to check that your string indeed contains a number.

5 Convert an integer to a string

Use the built-in function `format`:

```
format (number, 'd') # For base 10
format (number, 'x') # For base 16
format (number, 'o') # For base 8
format (number, 'b') # For base 2
```

See the specifications[4] for more on `format`

6 Launch an external process

In general, the subprocess[5] standard module is to be used to manage external processes. However, I have coded two helper functions for common tasks.

6.1 If you are just interested in the return status

Use `svlib.servicerunner.Runner.callCmd (['your_cmd', 'your_arg1', ...])` . It throws an `UnexpectedStatusError` if the return status is not zero (or the one expected, see code for more)

6.2 If you want the output too

Use `svlib.servicerunner.Runner.callCmdGetOutput(['your_cmd', 'your_arg1',...])`.

The usage is similar to callCmd, but it returns a tuple `(returncode, stdoutString, stderrString)`. See code for more.

7 Change the current directory

Use `os.chdir(path)`

[3] http://christophe-simonis-at-tiny.blogspot.com/2008/08/python-reverse-enumerate.html

[4] http://docs.python.org/library/string.html#formatspec

[5] http://docs.python.org/library/subprocess.html

8 Correctly reading a shell command

Often, I want to transform a shell command given as a string, such as `cat /etc/foo.txt "file with space"` into a list of arguments to pass to e.g `callCmd`.

Of course `str.split` is not enough here, since it will just ignore quoted arguments. The solution is to use `shlex.split`.