Module Sys for Python

Manuel Moreno

Bruno Ferreira

Analysis and demonstration about the module sys for the programming language Python

1.0.
Moduly Sys

Introduction

Sys module focus in allowing the user to utilize the standard input and output of the shell, but most importantly the user can now pass arguments to his Python script or program.
This happens because we a get access to variables that are maintained by the interpreter and we can use functions that interact with the said interpreter.
This presentation will show some of the usefull commands and programs of this module.

1.1.
Usefull commands

sys.stdin , sys.stdout
sys.stderr

The stdin and stdout are file objects used by the interpreter. These files are normal .txt documents that can be used by the scripts or the programs.

All the errors that are prompt by the interpreter go to the stderr so we can see which problems we encounter using these commands

Redirecting

```
#save std_out
stdout_s = sys.stdout

fd = open(“teste.txt” , ”w”)
sys.stdout = fd
print (“Print to file”)
# back to normal std_out
sys.stdout = stdout_s
```

sys.calltracing(func, args) and sys.settrace()

These commands allow Python to implement a debugger so we can see whats going on when there is a bug or an error that we cannot find. The command settrace(tracefunc) is a thread - specific function that supports the use of multithreads.

Call func(*args), while tracing is enabled. The tracing state is saved, and restored afterwards. This is intended to be called from a debugger from a checkpoint, to recursively debug some other code.

sys.argv

This is the most usefull command since it will be a bridge that communicates Python and other languages which can then communicate back through the shell to interact.
The idea of sys.argv is to allow you to pass arguments through to Python from the command line.

```
$ python argumentos.py -a -b -cc1 -d d1 arg1 arg2
Argument list:
['argumentos.py', '-a', '-b', '-c', 'c1', '-d', 'd1', 'arg1', 'arg2']
```

getopt module

In addition to sys.argv, if you use getopt module you can get the argument list grouped by options:

```
#previous script
optlist, argv = getopt.getopt(argv[1:], ‘abc:d:’)
print (optlist)
# [('-a', ''), ('-b', ''), ('-c', 'c1'), ('-d', 'd1')]
print (argv)
# ['arg1', 'arg2']
```