

Librerías y Módulos



Contenido

Módulo OS.....3

Archivos y directorios3

Variables de entorno4

Módulo sys5

Nuestros propios módulos5

Introducción

Python cuenta con una innumerable cantidad de librerías para las más diversas tareas, desde acceder a información del sistema, conectarse a un sitio web, como vimos en el módulo anterior hasta para realizar tareas de *debugging* y muchas cosas más. Es por ello que en este módulo vamos a ver algunos de los módulos más importantes que tiene Python por defecto y que son necesarios para interactuar con el sistema en el cuál estamos ejecutando nuestros programas.

Módulo OS

El módulo OS de Python nos permite, según el sistema operativo que estemos utilizando, acceder a un montón de funcionalidades relacionadas con el manejo de directorios, archivos, procesos y muchas otras opciones. En esta sección veremos algunas de las funcionalidades que este módulo tiene para ofrecer. Si bien no lo vamos a cubrir en su completitud vamos a las más importantes.

Es importante entender que las acciones de los métodos y funciones con las que cuenta este módulo pueden variar según el sistema operativo. Como recordarán, Python es multiplataforma, es por ello que según el sistema operativo en el cuál se esté ejecutando algunas cosas pueden variar. En la documentación de Python¹, pueden encontrar una gran cantidad de información acerca de este módulo.

Archivos y directorios

Es común encontrarse con la necesidad de navegar a través de diferentes carpetas del sistema o contar con directorios específicos para almacenar información. Para lograr este cometido Python incluye algunas funciones dentro del módulo OS que permiten cambiar el directorio actual, decidir si se trata de un archivo o un directorio, decirnos en dónde estamos y mucho más.

Algunas de las funciones que veremos durante esta sección incluyen el listado de directorios, renombrando archivos, navegando a través del sistema y demás, si bien no las veremos todas, nos encargaremos de hacer una fuerte introducción para que ustedes sigan programando e investigando al respecto.

Si lo que queremos es saber en qué directorio estamos parados podemos hacer uso de la función `os.getcwd()`, esta función retorna un *string* con el *path* absoluto, como pueden ver en la siguiente imagen:

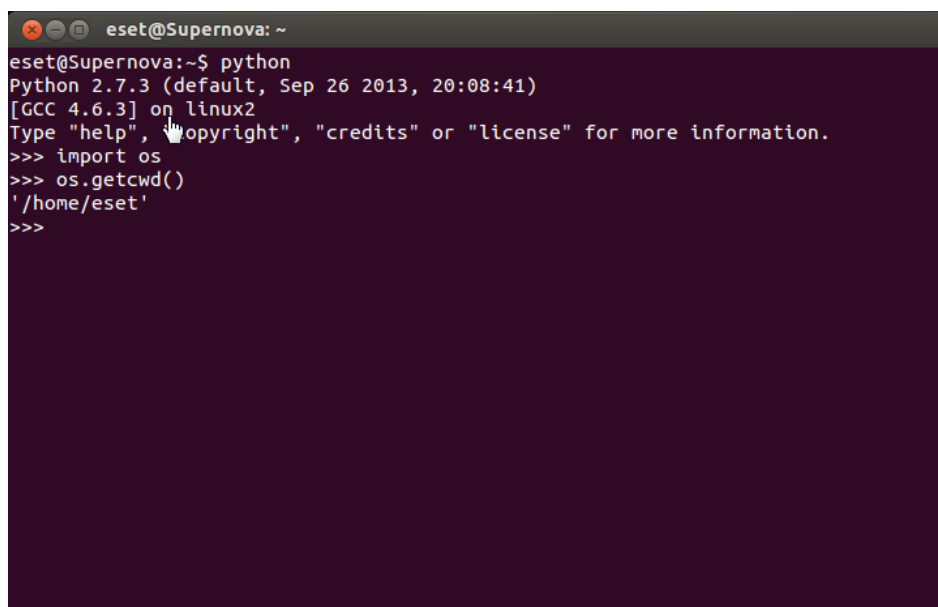
A screenshot of a terminal window with a dark background. The prompt is 'eset@Supernova: ~'. The user enters 'python', and the prompt changes to 'Python 2.7.3 (default, Sep 26 2013, 20:08:41)'. Below this, it says '[GCC 4.6.3] on linux2' and 'Type "help", "copyright", "credits" or "license" for more information.' The user then enters '>>> import os', '>>> os.getcwd()', and the output is '/home/eset'. The prompt returns to '>>>'.

Imagen 1 - `os.getcwd()`

Ahora, si por ejemplo, lo que ustedes quieren es listar el contenido de un directorio, y ver el listado de archivos y carpetas que se alojan en él, tienen que utilizar la función `os.listdir()` y pasarle como parámetro el *path* que quieren listar. Si combinan la función

¹ Brief Tour of the Standard Library: <http://docs.python.org/2/tutorial/stdlib.html>

anterior con esta, pueden listar todos los archivos y carpetas del directorio actual, ejecutando `os.listdir(os.getcwd())`. Este comando debería darles como salida un *array* con todos los nombres e archivos y carpetas que se encuentran en el directorio:

```

eset@Supernova: ~
>>> os.listdir(os.getcwd())
['curso_python', 'riak-1.4.8', 'Documents', 'riak-1.4.8.tar.gz', 'Desktop', '.compiz', '.bashrc', '.gnome2_private', '.dbus', '.config', '.gksu.lock', '.mysql', '.gtk-bookmarks', '.libreoffice', '.Xauthority', '.cache', 'Music', '.profile', 'python_edu-master.zip', '.eclipse', '.viminfo', '.gnome2', '.ssh', '.compiz-1', '.dmrc', '.gvfs', '.idlerc', '.dia', '.thumbnails', '.dropbox-dist', 'Videos', '.ICEauthority', '.xsession-errors.old', '.java', '.bash_logout', '.mysql_history', '.sudo_as_admin_successful', '.gconf', '.goutputstream-5M68EX', '.dropbox', '.irssi', '.pulse', '.mozilla', 'Templates', '.local', '.goutputstream-0ST6CX', '.esd_auth', '.pulse-cookie', 'Public', '.thunderbird', '.bash_history', 'Downloads', '.mission-control', '.goutputstream-3CXN6W', '.fontconfig', '.xsession-errors', '.swt', 'Pictures']
>>>

```

Imagen 2 - `os.listdir(os.getcwd())`

Cuando tenemos un listado de elementos y deseamos conocer si se trata de un archivo o un directorio, ¿cómo creen que pueden hacer? Bueno, Python lo tiene resuelto en dos simples funciones, `os.path.isfile()` y `os.path.isdir()`, ambas funciones retornarán un valor **booleano** que será True en caso de que la condición sea cierta. De esta manera, es sencillo conocer de qué estamos hablando si un archivo o un directorio, sin importar en qué sistema operativo estemos ejecutando nuestro código.

Otras funciones interesantes para investigar y probar en su código son `os.path.basename()` y `os.path.abspath()` pero se los vamos dejar a ustedes para que se entretengan un rato programando en Python.

A continuación pueden ver un pequeño script que al ejecutarse en una carpeta va a listar todo su contenido y comentar si se trata de un directorio, un archivo o algo que no ha podido identificar:

```

modulo_os.py
1  # Codigos de ejemplo acerca del uso del modulo os
2
3  import os
4
5  # Obtener el path en donde se esta ubicado
6  print "En este momento estas en %s" % os.getcwd()
7
8  # Listar todos los archivos
9
10 print r"En el directorio en donde estas ubicado hay %d archivos o carpetas" % len(os.listdir(os.getcwd()))
11
12 # Armarmamos un array con todos los archivos
13
14 listado = os.listdir(os.getcwd())
15 for elemento in listado:
16     if os.path.isdir(elemento):
17         print "--> %s es una carpeta" % elemento
18     elif os.path.isfile(elemento):
19         print "--> %s es un archivo" % elemento
20     else:
21         print "--> No se que es %s" % elemento
22

```

Imagen 3 - Recorriendo directorios

Variables de entorno

El módulo OS también nos permite acceder a las variables de entorno del sistema a través de su atributo ***os.environ*** o conocer el ***PID*** (*Process Identifier*) con el que podemos conocer el número del proceso con el cuál se está ejecutando nuestro programa, si es que en algún momento lo deseamos conocer.

Módulo sys

Esta librería nos permite obtener información acerca del entorno en el que se está ejecutando nuestro programa. Por ejemplo, a través de la llamada a ***sys.modules*** podemos conocer todos los módulos que se han cargado. Esto es muy sencillo de probar desde una consola de Python.

Además, cuenta con otros atributos que nos permiten conocer la cantidad de parámetros con los que se llamó al programa (***sys.argv***), en qué plataforma se está ejecutando (***sys.platform***), que directorios están incluidos (***sys.path***) hasta incluso agregar nuevos directorios a él.

Nuestros propios módulos

Cuando nosotros creamos nuestros propios archivos .py, y definimos en ellos las funciones o clases que creemos necesarias, podemos importarlas desde otro archivo de la misma manera que lo hacemos con los módulos ***os*** y ***sys***. De esta manera es posible organizar y dividir la lógica de nuestro programa de manera que sea lo más óptima posible.

Por si no lo recuerdan, para importar nuestros propios módulos desde otro script en Python debemos usar la palabra reservada ***import***, y luego el nombre del módulo que queremos importar:

```
>> import mi_modulo
```

Además, también se pueden importar clases o funciones específicas utilizando el siguiente formato:

```
>> from mi_modulo import mi_funcion, mi_clase
```

La principal diferencia entre importar un módulo entero o solo algunas de sus funciones o clases impacta en el consumo de memoria o en el rendimiento. Es algo sobre lo que pueden buscar un poco más de información en la web.

En algunas de las actividades que hicimos a lo largo del curso ustedes tuvieron que escribir sus propios módulos e importarlos desde otros scripts.