Módulos Librería Estándar

Antonio Espín Herranz

Contenidos

- Módulos para:
  - Sistema
  - Programador
  - Administradores
  - Necesidades funcionales
  - Internet

Módulos del sistema

- El módulo **os** nos permite acceder a funcionalidades dependientes del **Sistema Operativo.**

- El módulo os importa el módulo **ntpath** que le renombra a os.path, que aporta más funcionalidades.
  - **os.path.operacion()**

- Las operaciones se lanzan a partir de **os.operacion()**

- ***Alguna funcionalidades están más enfocadas a Linux que Windows.***

Descripción	Método
Saber si se puede acceder a un archivo o directorio	`os.access(path, modo_de_acceso)`
Conocer el directorio actual	`os.getcwd()`
Cambiar de directorio de trabajo	`os.chdir(nuevo_path)`
Cambiar al directorio de trabajo raíz	`os.chroot()`
Cambiar los permisos de un archivo o directorio	`os.chmod(path, permisos)`
Cambiar el propietario de un archivo o directorio	`os.chown(path, permisos)`
Crear un directorio	`os.mkdir(path[, modo])`
Crear directorios recursivamente	`os.mkdirs(path[, modo])`
Eliminar un archivo	`os.remove(path)`
Eliminar un directorio	`os.rmdir(path)`
Eliminar directorios recursivamente	`os.removedirs(path)`
Renombrar un archivo	`os.rename(actual, nuevo)`
Crear un enlace simbólico	`os.symlink(path, nombre_destino)`

Variables de entorno

- **os.environ**
  - Devuelve un **diccionario** (k, v) con las variables de entorno:

Ejemplo

```
for k, v, os.environ.items():
    print(k,v)
```

os.path

Descripción	Método
Ruta absoluta	`os.path.abspath(path)`
Directorio base	`os.path.basename(path)`
Saber si un directorio existe	`os.path.exists(path)`
Conocer último acceso a un directorio	`os.path.getatime(path)`
Conocer tamaño del directorio	`os.path.getsize(path)`
Saber si una ruta es absoluta	`os.path.isabs(path)`
Saber si una ruta es un archivo	`os.path.isfile(path)`
Saber si una ruta es un directorio	`os.path.isdir(path)`
Saber si una ruta es un enlace simbólico	`os.path.islink(path)`
Saber si una ruta es un punto de montaje	`os.path.ismount(path)`

Módulo sys

- Sirve para capturar los parámetros de la línea de comandos y ciertas propiedades del intérprete.

Variable	Descripción
`sys.argv`	Retorna una lista con todos los argumentos pasados por línea de comandos. Al ejecutar `python modulo.py arg1 arg2`, retornará una lista: `['modulo.py', 'arg1', 'arg2']`
`sys.executable`	Retorna el path absoluto del binario ejecutable del intérprete de Python
`sys.maxint`	Retorna el número positivo entero mayor, soportado por Python
`sys.platform`	Retorna la plataforma sobre la cuál se está ejecutando el intérprete
`sys.version`	Retorna el número de versión de Python con información adicional

Módulo sys

- Métodos:

Método	Descripción
`sys.exit()`	Forzar la salida del intérprete
`sys.getdefaultencoding()`	Retorna la codificación de caracteres por defecto
`sys.getfilesystemencoding()`	Retorna la codificación de caracteres que se utiliza para convertir los nombres de archivos unicode en nombres de archivos del sistema
`sys.getsizeof(object[, default])`	Retorna el tamaño del objeto pasado como parámetro. El segundo argumento (opcional) es retornado cuando el objeto no devuelve nada.

Módulo subprocess

- El módulo **subprocess** es aquel que nos permite trabajar de forma directa con **comandos del sistema operativo**.
- OJO funciona en linux.

```
from subprocess import call
call('clear')
comando_y_argumentos = ['ls', '-lha']
call(comando_y_argumentos)
```

Módulo subprocess

- from subprocess import **Popen**
- Popen(['ls', '-la']

- En vez de una función es un objeto, hay que pasar el comando y los parámetros.

- Podemos redirigir la salida del comando y se puede utilizar Pipes.
  - **stdout**: nomenclatura correspondiente a la salida estándar en sistemas UNIX-Like. Es la encargada de almacenar la salida de un programa.
  - **stdin**: nomenclatura correspondiente a la entrada estándar en sistemas UNIX-like. Es la encargada de enviar información a un programa.
  - **stderr**: al igual que las anteriores, se utiliza como referencia a los errores producidos en la salida de un programa.

Módulo de subprocess

- from subprocess import PIPE, Popen
- proceso = Popen(['ls', '-lha'], stdout=PIPE, stderr=PIPE)

- Se puede leer la salida como si tratara de un fichero:

```
proceso = Popen(['ls', '-lha'], stdout=PIPE, stderr=PIPE)
error_econtrado = proceso.stderr.read()
listado = proceso.stdout.read()
```

Módulos del Programador

- Utilización de **Pdb** para **depurar** programas.
- Se encuentra en el **módulo pdb**, tenemos que llamar al método: **set_trace**().
- Nos mostrará un menú:

```
n (next) ejecuta el código mostrado y salta a la siguiente línea de tu archivo
s (step) te mostrará paso a paso el camino recorrido
         hasta poder ejecutar la siguiente línea de tu archivo
c (continue) ejecuta el archivo hasta encontrar un punto de quiebre
q (quit) abandonar el debugger
```

- **Mejor trabajar con un IDE** que permita la depuración: VSC, Eclipse, PyCharm.

Ejemplo

```
import pdb from subprocess
import call, Popen, PIPE

# Limpiar la pantalla call("clear")
pdb.Pdb().set_trace() ←
proceso = Popen(['ls', '-lha'], stdout=PIPE, stderr=PIPE)
error_encontrado = proceso.stderr.read()
proceso.stderr.close()
listado = proceso.stdout.read()
proceso.stdout.close()
if not error_encontrado:
    print( listado)
else:
    print ("Se produjo el siguiente error:\n%s" % error_encontrado)
```

Módulos Administradores

- El módulo **platform**:

```
>>> import platform
>>> import platform as p
>>> p.architecture()
('32bit', 'WindowsPE')
>>> p.dist()
('', '', '')
>>> p.machine()
'AMD64'
>>> profile = [
... platform.architecture(),
... platform.dist(),
... platform.libc_ver(),
... platform.machine(),
... platform.node(),
... platform.platform(),
... platform.processor(),
... platform.python_build(),
... platform.python_compiler(),
... platform.python_version(),
... platform.system(),
... platform.uname(),
... platform.version(),
... ]
>>> profile
[('32bit', 'WindowsPE'), ('', '', ''), ('', ''), 'AMD64', 'WORKER-PC', 'Windows-
7-6.1.7601-SP1', 'Intel64 Family 6 Model 23 Stepping 10, GenuineIntel', ('v3.6.2
:5fd33b5', 'Jul  8 2017 04:14:34'), 'MSC v.1900 32 bit (Intel)', '3.6.2', 'Windo
ws', uname_result(system='Windows', node='WORKER-PC', release='7', version='6.1.
7601', machine='AMD64', processor='Intel64 Family 6 Model 23 Stepping 10, Genuin
eIntel'), '6.1.7601']
>>>
```

Módulos Necesidades Funcionales

- Obtener número aleatorios

Método	Descripción
`random.randint(a, b)`	Retorna un número aleatorio entero entre `a` y `b`
`random.choice(secuencia)`	Retorna cualquier dato aleatorio de secuencia
`random.shuffle(secuencia)`	Retorna una mezcla de los elementos de una secuencia
`random.sample(secuencia, n)`	Retorna n elementos aleatorios de secuencia

Ejemplo

```
import random

# Generar números aleatorios entre 49999 y 99999
lista = []

for n in range(0, 50):
    lista.append(random.randint(49999, 99999))

# Elegir un número al azar
numero_al_azar = random.choice(lista)

# Elegir 5 números al azar
numeros_al_azar = random.sample(lista, 5)

# reordenar los elementos de una lista
mujeres = ["Ana", "Beatriz", "Camila", "Carmen", "Delia", "Dora", "Emilse"]
random.shuffle(mujeres)
```

Módulo textwrap

- Para ajustar un texto a número de caracteres por línea, sin cortar palabras (siempre que pueda).

```
import textwrap
texto = “….”
lineas = textwrap.wrap(texto, 60)
```

Módulos e Internet

- Abrir un navegador:
  - import webbrowser
  - webbrowser.open_new_tab("http://...")

- Conectar con un servidor de **FTP**:
  - El **módulo ftplib** de la librería estándar de Python, nos provee de los métodos necesarios para crear clientes FTP de forma rápida y sencilla.

Módulo ftplib

- from ftplib import FTP
- # Conectarse con los métodos connect y login

```
ftp = FTP()
ftp.connect('66.228.52.93', 21, -999)
ftp.login('miuser', 'miclave')
```

- # Conectarse en la instancia a FTP

```
ftp = FTP('66.228.52.93', 'miuser', 'miclave')
```

Método	Descripción
`FTP.connect(host[, puerto, timeout])`	Se conecta al servidor FTP
`FTP.login(user, pass)`	Se loguea en el servidor
`FTP.close()`	Finaliza la conexión
`FTP.set_pasv(bool)`	Establece la conexión en modo pasivo si el parámetro es `True`
`FTP.getwelcome()`	Retorna el mensaje de bienvenida del servidor
`FTP.dir()`	Retorna un listado de archivos y directorios de la carpeta actual
`FTP.cwd(path)`	Cambia el directorio de trabajo actual a `path`
`FTP.mkd(path)`	Crea un nuevo directorio
`FTP.pwd()`	Retorna el directorio de trabajo actual
`FTP.rmd(path)`	Elimina el directorio `path`
`FTP.storlines('STOR destino', open(localfile, 'r'))`	Lee `localfile` y lo escribe en `destino`
`FTP.rename(actual, nuevo)`	Renombra el archivo `actual` por `nuevo1`
`FTP.delete(filename)`	Elimina un archivo
`FTP.retrlines('RETR archivo_remoto')`	Lee archivo_remoto y retorna su contenido

Ejemplo

```
from ftplib import FTP
ftp = FTP()
ftp.connect('66.228.52.93', 21, -999)
ftp.login('user', 'pass')
print( ftp.getwelcome() )
ftp.mkd('nuevo-dir')
ftp.cwd('nuevo-dir')
print( ftp.pwd() )
ftp.storlines('STOR example.txt', open('ftp_examples.py', 'r'))
ftp.rename('example.txt', 'example.py')
ftp.dir()
archivo = ftp.retrlines('RETR example.py')
print(archivo)
ftp.close()
```

glob

- **glob.glob**(pathname, *, root_dir=None, dir_fd=None, recursive=False, include_hidden=False)

- Retorna una lista posiblemente vacía de nombres de ruta que coincidan con pathname, que debe ser una cadena de caracteres que contenga una especificación de ruta. pathname puede ser absoluto (como /usr/src/Python-1.5/Makefile) o relativo (como ../../Tools/*/*.gif) y puede contener wildcards de estilo shell. Los enlaces simbólicos rotos se incluyen en los resultados (como en el shell). La clasificación de los resultados depende del sistema de archivos. Si un archivo que cumple las condiciones se elimina o se agrega durante la llamada de esta función, no se especifica si se incluirá un nombre de ruta para ese archivo.

- Si **root_dir** no es None, debería ser un path-like object que especifique el directorio raíz para la búsqueda. Tiene el mismo efecto en glob() que cambiar el directorio actual antes de llamarlo. Si pathname es relativo, el resultado contendrá rutas relativas a root_dir.

glob

- Esta función puede admitir rutas relativas a descriptores de directorio con el parámetro dir_fd.

- Si recursive es verdadero, el patrón «**» coincidirá con cualquier fichero y cero o más directorios, subdirectorios y enlaces simbólicos a directorios. Si el patrón va seguido de un os.sep o os.altsep los ficheros no coincidirán.

- Si include_hidden es verdadero, el patrón «**» coincidirá con los directorios ocultos.

Ejemplo

```
import glob


print('Named explicitly:')
for name in glob.glob('/home/geeks/Desktop/gfg/data.txt'):
    print(name)

# Using '*' pattern
print('\nNamed with wildcard *:')
for name in glob.glob('/home/geeks/Desktop/gfg/*'):
    print(name)
```

•

Ejemplo

```
# Using '?' pattern
print('\nNamed with wildcard ?:')
for name in
glob.glob('/home/geeks/Desktop/gfg/data?.txt'):
    print(name)

# Using [0-9] pattern
print('\nNamed with wildcard ranges:')
for name in glob.glob('/home/geeks/Desktop/gfg/*[0-9].*'):
    print(name)
```

datetime

El módulo `datetime` proporciona clases para manipular fechas y horas.
Si bien la implementación permite operaciones aritméticas con fechas y horas, su principal objetivo es poder extraer campos de forma eficiente para su posterior manipulación o formateo.

```
class datetime.datetime(year, month, day,
hour=0, minute=0, second=0, microsecond=0,
tzinfo=None, *, fold=0)
```

https://docs.python.org/es/3/library/datetime.html#datetime-objects

math

- Este módulo proporciona acceso a las funciones matemáticas definidas en el estándar de C.

- Estas funciones no pueden ser usadas con números complejos; usa las funciones con el mismo nombre del módulo cmath si requieres soporte para números complejos. La distinción entre las funciones que admiten números complejos y las que no se hace debido a que la mayoría de los usuarios no quieren aprender tantas matemáticas como se requiere para comprender los números complejos. Recibir una excepción en lugar de un resultado complejo permite la detección temprana del número complejo inesperado utilizado como parámetro, de modo que el programador pueda determinar cómo y porqué se generó en primer lugar.

- Este módulo proporciona las funciones descritas a continuación. Excepto cuando se indique lo contrario explícitamente, todos los valores retornados son flotantes.

math

- Funciones de redondeo:
  - floor, ceil, round
- Trigonométricas: sin, cos, tan
- Propiedades: pi, e

timeit

```
python3 -m timeit '"-".join(str(n) for n in range(100))'
10000 loops, best of 5: 30.2 usec per loop

python3 -m timeit '"-".join([str(n) for n in range(100)])'
10000 loops, best of 5: 27.5 usec per loop

python3 -m timeit '"-".join(map(str, range(100)))'
10000 loops, best of 5: 23.2 usec per loop
```

timeit

```
import timeit
timeit.timeit('"-".join(str(n) for n in range(100))', number=10000)
0.3018611848820001

timeit.timeit('"-".join([str(n) for n in range(100)])', number=10000)
0.2727368790656328

timeit.timeit('"-".join(map(str, range(100)))', number=10000)
0.2370267980732023 7
```