Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

miércoles, 30 de septiembre de 2015

Filtrando archivos y directorios con glob y fnmatch

El módulo Glob

El módulo **glob** incluye funciones para buscar en una ruta todos los nombres de archivos y/o directorios que coincidan con un determinado patrón que puede contener comodines como los que se utilizan en un interprete de comandos tipo Unix (*, ?, []).

Estos patrones no deben confundirse con los patrones utilizados en las expresiones regulares del módulo **re** aunque guarden cierto parecido.

A continuación, se muestran las funciones del modulo **glob** con algunos ejemplos de uso.

Obtener entradas que coincidan con un patrón: glob.glob()

La función **glob.glob()** devuelve una lista con las entradas que coincidan con el patrón especificado en **pathname**.

```
glob.glob(pathname, recursive=False)
```

La búsqueda se puede hacer también recursiva con el argumento **recursive=True** y las rutas pueden ser absolutas y relativas.

Patrón	Salidas válidas
? Un carácter:	
'/home/ant/img/201?'	'/home/ant/img/2013' '/home/ant/img/2014' '/home/ant/img/2015'
*** Uno o más caracteres:**	
'../../../img/201?/*'	'/home/ant/img/2014/Huelva' '/home/ant/img/2014/Sevilla' '/home/ant/img/2015/Granada'
'../../../img/2015/*.JPG'	'/home/ant/img/2015/uno.JPG' '/home/ant/img/2015/dos.JPG'
[] Caracteres permitidos:	
'/home/ant/img/201[45]'	'/home/ant/img/2014' '/home/ant/img/2015'
[!] Caracteres no permitidos:	

Buscar

Buscar

Python para impacientes

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

Anexos

Guía urgente de MySQL
Guía rápida de SQLite3

Entradas + populares

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

El módulo random
El módulo random de la librería estándar de Python incluye un conjunto de funciones

'/home/ant/img/201[!45]'	'/home/ant/img/2013'
[-] Rango caracteres permitidos	
'/home/ant/img/201[0-9]'	'/home/ant/img/2013' '/home/ant/img/2014' '/home/ant/img/2015'
** **Búsqueda recursiva, recursive=True** (se incluyen archivos)	
'/home/ant/img/2015/**' (con recursive=True)	'/home/ant/img/2015/Granada/1.jpg' '/home/ant/img/2015/Granada/2.jpg'
'/home/ant/img/2015/**/*.jpg' (con recursive=True)	'/home/ant/img/2015/Granada/1.jpg' '/home/ant/img/2015/Granada/2.jpg'

La búsqueda recursiva con ** está disponible a partir de Python 3.5.

En el siguiente ejemplo se utiliza **glob.glob()** con varios patrones deferentes:

```
import glob
raices = ['/home/ant/img/201?',
          '../../img/201?/*',
          '/home/ant/img/201[345]',
          '/home/ant/img/201[!345]',
          '/home/ant/img/201[0-9a-z]',
          '../../img/2015/F*/*.JPG']

for raiz in raices:
    print('\n'+raiz)
    rutas = glob.glob(raiz)
    print('rutas:', len(rutas))
    for ruta in rutas:
        print(ruta)

# glob con un patrón que utiliza recursividad
# Se obtienen los directorios y archivos coincidentes

raiz = '/home/ant/img/2015/**'
rutas = glob.glob(raiz, recursive=True)
print('\nrutas:', len(rutas))
for ruta in rutas:
    print(ruta)
```

Obtener iterador con entradas que coincidan con un patrón: glob.iglob()

```
glob.iglob(pathname, recursive=False)
```

La función **glob.iglob()** es como la función **glob.glob()** pero en vez de devolver los elementos coincidentes en una lista, retorna un iterador, evitando con ello que se tengan que almacenar la entradas en memoria simultáneamente.

```
raiz = '/home/ant/img/2015/**'
for ruta in glob.iglob(raiz, recursive=True):
    print(ruta)
```

Adaptar cadenas con caracteres especiales: escape()

La función **glob.escape()** es útil cuando los nombres de archivos o directorios a obtener pueden contener caracteres especiales ('?', '*' y '[') como los utilizados en los patrones. La función adapta la cadena del **pathname** para que los caracteres especiales tengan el mismo tratamiento que el resto de caracteres. Esta función está disponible desde Python 3.4.

```
escape(pathname)
```

Ejemplo:

que permiten obtener de distintos modos números a...

Archivo

septiembre 2015 (3)

python.org

pypi.org

Sitios

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears

escape('/Viajar?.txt') devuelve '/Viajar[?].txt'

```
raiz = glob.escape('/home/ant/img/2016/viajar?.JPG')
print(raiz)
rutas = glob.glob(raiz)
print('\nrutas', len(rutas))
for ruta in rutas:
    print(ruta)
```

El módulo fnmatch

El módulo **fnmatch** cuenta con funciones que facilitan la selección de archivos y directorios que coinciden con un determinado patrón de búsqueda. Estos patrones al igual que en las funciones del módulo **glob** utilizan los mismos comodines de un interprete de comandos tipo Unix (*, ?, [secuencia], [!secuencia]).

Las funciones de **fnmatch** son un buen complemento para **os.listdir()**, **os.scandir()** y **os.walk()** porque permiten filtrar los elementos encontrados en un directorio por una extensión, un comienzo o final de nombre determinado, por la aparición de unos caracteres específicos en unas posiciones concretas, etc.

Por otro lado, para buscar entradas que contienen en sus nombres caracteres especiales es necesario escribir en los patrones estos caracteres entre corchetes. Por ejemplo: la expresión [?] se corresponde con el carácter '?'.

A continuación, se detallan las funciones de este modulo con algunos ejemplos de utilización.

Comprobar si una entrada coincide con un patrón: fnmatch.fnmatch()

La función **fnmatch.fnmatch()** devuelve **True** si el nombre de una entrada coincide con el patrón indicado. Si el sistema operativo diferencia mayúsculas de minúsculas la función actuará de la misma forma.

```
fnmatch.fnmatch(filename, pattern)
```

Para que un sistema trate por igual las mayúsculas y minúsculas es necesario normalizar de alguna forma **filename** y **pattern**; por ejemplo, convirtiendo ambos argumentos a mayúsculas o a minúsculas.

```
import os, fnmatch

for archivo in os.listdir('.'):
    if fnmatch.fnmatch(archivo.upper(), '*.MD'):
        print(archivo)
```

Comprobar si una entrada coincide diferenciando minúsculas y mayúculas: fnmatch.fnmatchcase()

```
fnmatch.fnmatchcase(filename, pattern)
```

La función **fnmatch.fnmatchcase()** devuelve **True** si el nombre de una entrada coincide con el patrón indicado diferenciando, en cualquier caso, entre mayúsculas y minúsculas.

```
for archivo in os.listdir('.'):
    if fnmatch.fnmatchcase(archivo, '*.md'):
        print(archivo)
```

Obtener lista con entradas que coinciden con patrón: fnmatch.filter()

```
fnmatch.filter(names, pattern)
```

La función **fnmatch.filter()** devuelve una lista con los nombres que coinciden con el patrón. Es equivalente a [n for n in names if fnmatch(n, pattern)].

```
directorio = os.listdir('.')
for archivo in fnmatch.filter(directorio, 'glob*'):
    print(archivo)
```

Obtener patrón de búsqueda convertido en expresión regular: fnmatch.translate()

```
fnmatch.translate(pattern)
```

Devuelve un patrón de búsqueda convertido en expresión regular de las que se emplean con el módulo **re**.

```
import fnmatch, re

expresion_regular = fnmatch.translate('*.md')
print('Expresión regular:', expresion_regular)
objeto_re = re.compile(expresion_regular)
if objeto_re.match('Nombre_de_fichero.md'):
    print('Nombre_de_fichero.md coincide con la expresión regular')
```

Relacionado: Explorando directorios con listdir, walk y scandir

Ir al índice del tutorial de Python

Publicado por Pherkad en 11:04

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua