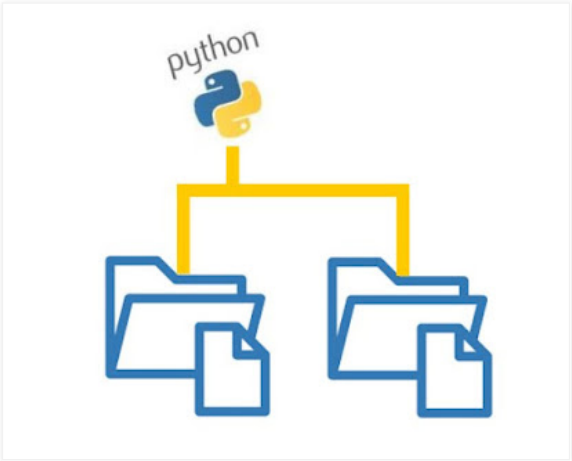
martes, 16 de octubre de 2018

Pathlib: rutas orientadas a objetos



El módulo **pathlib** disponible desde Python 3.4 consta de clases que permiten representar rutas de archivos y directorios de un sistema de ficheros con la semántica adecuada para distintos sistemas operativos.

Las clases cuentan con métodos que facilitan una amplia variedad de operaciones con directorios, archivos y enlaces simbólicos; que antes eran posibles, la mayoría, recurriendo a funciones y clases de varios módulos (como **os**, **glob**, **shutil** y **fileinput**). Lo que se ha pretendido con **pathlib** es concentrar en un módulo las operaciones más comunes teniendo en cuenta la experiencia de los desarrolladores.

Dentro de las operaciones contempladas las hay para explorar el sistema de ficheros, comprobar la existencia de rutas; recorrer los archivos y subdirectorios de un directorio filtrando los objetos mediante el uso de patrones; obtener información de los objetos (tamaño, permisos, propietarios, etc.); crear, renombrar y borrar archivos y enlaces simbólicos; crear y borrar directorios; obtener rutas del sistema en varios formatos (**Posix** y **URI**), principalmente.

Rutas puras y concretas

Las clases de **pathlib** se dividen en dos grupos en función a los tipos de rutas orientadas a objetos que manejan: puras y concretas. La diferencia entre ambos tipos de clases está en que las clases para rutas puras proporcionan métodos que en realidad no acceden al sistema de ficheros y las de rutas concretas, sí.

Las rutas son inmutables y por ser únicas en cada sistema de archivos se les pueden aplicar funciones hash. Las rutas de un mismo sistema se pueden comparar y ordenar de acuerdo a la semántica empleada en cada sistema.

Cualquier ruta se representa como una cadena adaptada a las rutas propias de cada sistema de archivos y, lógicamente, se podrán utilizar con cualquier función que opere con rutas en este mismo formato.

Hay tres formas de instanciar objetos de rutas puras:

- **class pathlib.PurePath**(**pathsegments*): clase genérica que crea una instancia de ruta en función del sistema operativo: puede ser **PurePosixPath** o un **PureWindowsPath**.
- **class pathlib.PurePosixPath**(**pathsegments*): subclase que crea una instancia de ruta para sistemas de ficheros no Windows.
- **class pathlib.PureWindowsPath**(**pathsegments*): subclase que crea una instancia de ruta para sistemas de ficheros Windows.

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

A continuación, se declaran varias rutas puras y se muestran varios ejemplos que ilustran el uso de sus métodos:

```
import os
from pathlib import PurePath, PurePosixPath, PureWindowsPath

# Declarar rutas puras

ruta1 = PurePath('home/usuario')
ruta2 = PurePath('c:\\windows')
ruta3 = PurePosixPath('home/Usuario')
ruta4 = PureWindowsPath('C:\\WINDOWS')

# Imprimir las rutas

print(ruta1)  # home/usuario
print(ruta2)  # c:\windows
print(ruta3)  # home/Usuario
print(ruta4)  # C:\WINDOWS

# Las rutas son cadenas de un tipo especial

print(type(ruta1))  #
print(type(ruta2))  #
print(type(ruta3))  #
print(type(ruta4))  #

# Las rutas se pueden comparar ...

print(ruta1 == ruta3)  # False
print(ruta2 == ruta4)  # True (en Windows)
print(ruta2 in [ruta4])  # True (en Windows)
print(PureWindowsPath('c:\\windows') > ruta4)  # False
if os.name != 'posix':
    print(ruta2 > ruta4)  # False (en Win); Error (en No Win)

# Una ruta se representa como una cadena adaptada a
# las rutas propias de cada sistema de archivos

ruta5 = PurePath('/var/www/index.html')
ruta6 = PureWindowsPath('c:/Program Files')

print(ruta5)  # /var/www/index.html
print(ruta6)  # c:\Program Files

# Obtener por separado en una tupla todas las partes de una ruta

print(ruta5.parts)  # ('/', 'var', 'www', 'index.html')
print(ruta6.parts)  # ('c:\\', 'Program Files')

# Obtener la unidad (si existe) de una ruta

print(ruta5.drive)  # ''
print(ruta6.drive)  # 'c:'

# Obtener la raíz de una ruta. Válido para rutas UNC

print(ruta5.root)  # '/'
print(ruta6.root)  # '\'

# Obtener la raíz y la ruta ¡de una vez!

print(ruta5.anchor)  # '/'
print(ruta6.anchor)  # 'c:\'

# Obtener los ancestros de una ruta

print(ruta5.parents)  #
print(len(ruta5.parents))  # Número de ancestros: 3
print(ruta5.parents[0])  # /var/www
print(ruta5.parents[1])  # /var
print(ruta5.parents[2])  # /

# Obtener el ancestro ineidato o padre de una ruta

print(ruta5.parent)  # /var/www
print(ruta6.parent)  # c:\

# Obtener nombre y extensión/es de archivos
```

que permiten obtener de distintos modos números a...

Archivo

octubre 2018 (2)

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

```
print(ruta5.name)  # index.html
print(ruta5.suffix)  # .html
print(ruta5.suffixes)  # ['.html']
print(ruta5.stem)   # index

# Obtener rutas en formato POSIX y URI

print(ruta6.as_posix())  # c:/Program Files
print(ruta6.as_uri())  # file:///c:/Program%20Files

# Obtener si una ruta es absoluta: si tiene raíz o unidad

print(ruta5.is_absolute())  # True
print(ruta6.is_absolute())  # True

# Combinar rutas

print(ruta6.joinpath('Lupa'))  # c:\Program Files\Lupa
print(ruta6)  # c:\Program Files

# Comprobar si un patrón existe en una ruta

print(ruta5.match('*.html'))  # True
print(ruta5.match('var/*.html'))  # False

# Obtener la ruta relativa (si es posible)

print(ruta5.relative_to('/var'))  # www/index.html
print(ruta5.relative_to('/var/www'))  # index.html

# A partir de una ruta obtener otra ruta cambiando
# nombre completo de archivo o extensión (si es posible)

print(ruta5.with_name('imagen.jpg'))  # /var/www/imagen.jpg
print(ruta5.with_suffix('.htm'))  # /var/www/index.htm
```

Por otra parte, los objetos de ruta concreta pertenecen a subclases de las clases de rutas puras. Estas subclases además de ofrecer las operaciones proporcionadas por las clases de las que descienden cuentan con métodos para instancias de rutas que hacen llamadas al sistema. Hay tres formas de instanciar estos objetos:

- **class pathlib.Path**(**pathsegments*): es una subclase de la clase **PurePath** que crea una instancia de ruta concreta en función del sistema operativo: puede ser **PosixPath** o **WindowsPath**.
- **class pathlib.PosixPath**(**pathsegments*): es una subclase de la clase **PurePosixPath** que crea una instacia de ruta concreta para sistemas de ficheros no Windows.
- **class pathlib.WindowsPath**(**pathsegments*): es una subclase de la clase **PureWindowsPath** que crea una instancia de ruta concreta para sistemas de ficheros Windows.

A continuación se declaran varias rutas concretas y se ofrecen varios ejemplos que muestran el uso de sus métodos:

```
from pathlib import Path, PosixPath, WindowsPath

# Declarar rutas concretas

ruta1 = Path('home/usuario')
ruta2 = Path('c:\\windows')
ruta3 = PosixPath('home/Usuario')
ruta4 = WindowsPath('C:\\WINDOWS')  # Error en Linux

# Imprimir las rutas

print(ruta1)  # home/usuario
print(ruta2)  # c:\windows
print(ruta3)  # home/Usuario
print(ruta4)  # C:\WINDOWS

# Las rutas son cadenas de un tipo especial ...

print(type(ruta1))  #
print(type(ruta2))  #
print(type(ruta3))  #
print(type(ruta4))  #

# Obtener ruta del directorio actual de trabajo
```

```
print(Path.cwd())  # /home/usuario/Directorio

# Obtener la ruta del directorio de usuario actual

print(Path.home())  # /home/usuario

# Obtener información del directio o del archivo

# Obtener tamaño en bytes

print(Path('/etc/os-release').stat().st_size)  # 386

# Obtener tiempo de modificación (expresado en segundos)

print(Path('/etc/os-release').stat().st_mtime)  # 1534722298.0

# Para enlaces simbólicos: lstat()

# Cambiar los atributos y/o permisos del directorio o del archivo

Path('/home/usu1/imagen1.png').chmod(0o666)

# Para enlaces simbólicos: lchmod()

# Comprobar si existe el archivo o directorio

print(Path('/etc/os-release').exists())  # True

# Expandir la ruta añadiendo la ruta 'home' del usuario actual

print(Path('~/bashrc').expanduser())  # /home/usuario/bashrc

# Obtener de la ruta una lista con los archivos del patrón

archivos = Path('/home/usu1/Descargas').glob('*.pdf')
for archivo in archivos:
    print(archivo)

# Otros patrones:
# '*/*.pdf' Obtener del directorio con 1 nivel de profundidad
# '**/*.pdf' Obtener del directorio actual y sus subdirectorios
# El patrón '**/*.pdf' es equivalente a rglob('*.pdf')

# Obtener nombre del grupo propietario

print(Path('/etc/os-release').group())  # root

# Conocer si una ruta es un directorio o un enlace simbólico
# que apunta a un directorio existente

print(Path('/etc/os-release').is_dir())  # False

# Conocer si una ruta es un fichero o un enlace simbólico
# que apunta a un fichero existente

print(Path('/etc/os-release').is_file())  # True

# Conocer si una ruta es un enlace simbólico

print(Path('/etc/resolv.conf').is_symlink())  # True

# Conocer si la ruta es un punto de montaje.
# No implementado para windows. Nuevo en Python 3.7

print(Path('/media/usuario').is_mount())  # True

# Obtener iterador con objetos contenidos en un directorio

for elemento in Path('/home/usu1/Descargas').iterdir():
    print(elemento)  # Lista ficheros y directorios

# Crear un directorio (si no existe)

Path('/home/u1/bd').mkdir(mode=0o777, parents=False, exist_ok=True)

# Si parents=True creará los directorios padres que falten
# en el camino
# Si exist_ok=True y el directorio a crear existe se
# omitirá mensaje de error

# Abrir un archivo para leer y/o escribir
```

```
with Path('/etc/os-release').open(mode='r') as archivo:
    for linea in archivo:
        print(linea.strip())

# Leer y escribir de/en un archivo de texto

archivo = Path('secreto.txt')
archivo.write_text('Este archivo guarda un gran secreto')
print(archivo.read_text())

# Leer y escribir de/en un archivo en formato binario

archivo = Path('secreto.dat')
archivo.write_bytes(b'Este archivo guarda un gran secreto')
print(archivo.read_bytes())

# Conocer el propietario de un archivo

print(Path('/etc/resolv.conf').owner())  # root

# Renombrar un archivo o un directorio

Path('/home/usu1/imagen1.png').rename('/home/usu1/imagen2.png')

# En Unix si se trata de un archivo existeste será reemplazado.

# Renombrar un archivo o un directorio.

Path('/home/usu1/imagen2.png').replace('/home/usu1/imagen1.png')

# Si existe archivo o directorio será reemplezado

# Convertir una ruta en absoluta

print(Path('/etc/resolv.conf').resolve())
# /run/resolvconf/resolv.conf

print(Path('..').resolve())  # /home/usuario/Local

# Borrar un directorio vacío

dir1 = Path('/home/usu1/temp-1')
if dir1.exists():
    dir1.rmdir()

# Conocer si hay coincidencia en dos rutas

documentos = '/home/usu1/.bashrc'
print(Path('/home/usu1/.bashrc').samefile(documentos))  # True

# Crear un enlace simbólico

Path('MiEnlace').symlink_to('imagen.png')

# Borrar un archivo o un enlace simbólico

Path('imagen.png').unlink()
```

Relacionado:

- Acercamiento a la biblioteca estándar
- Explorando directorios con listdir, walk y scandir
- Filtrando archivos y directorios con glob y fnmatch
- Copiar, mover y borrar archivos/directorios con shutil
- Operaciones con archivos
- Fileinput: procesar múltiples archivos fácilmente

Ir al índice del tutorial de Python

Publicado por Pherkad en 4:26

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua