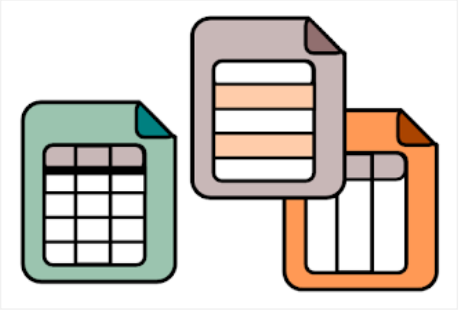
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

jueves, 12 de enero de 2017

Tablas con estilo con Tabulate



El módulo **Tabulate**, desarrollado por Sergey Astanin, permite imprimir en la salida estándar o escribir en un archivo de texto tablas con datos tabulados con varios formatos conocidos. Los datos se alinean automáticamente atendiendo a su tipo (cadena, entero y flotante) aunque es posible cambiar la configuración por defecto.

Tabulate se puede utilizar como librería (importada) en un programa y en la línea de comandos.

Instalación

Instalar Tabulate con el instalador PIP:

```
$ pip install tabulate
```

Instalar utilizando un proxy:

```
$ pip install tabulate --proxy http://user:passw@srv_proxy:puerto
```

La función tabulate()

El módulo tiene sólo una función, **tabulate()**, que imprime tablas a partir de los datos de entrada de su primer argumento, aceptando para su representación distintos tipos de estructuras de datos:

- Lista de listas u otro objeto iterable con contenido iterable.
- Lista o un iterable de diccionarios Python (cada clave del diccionario se corresponde con cada columna de la tabla).
- Diccionario Python con objetos iterables. (cada clave del diccionario se corresponde con cada columna de la tabla).
- Array NumPy bidimensional.
- Matriz Numpy.
- Y objeto pandas.Dataframe

Imprimir una tabla con datos tabulados

En el siguiente ejemplo se imprimen tabulados los datos de una lista que contiene varias listas con los nombres de algunos ríos de Andalucía y su longitud correspondiente (en kilómetros). En la salida que se obtiene los nombres de los ríos se alinean, automáticamente, a la izquierda y los números de las distancias a la derecha con respecto a su parte entera:

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...

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...

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...

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...

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute simultáneamente varias operaciones en el mismo espacio de proceso se...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos

```
from tabulate import tabulate

# Imprime tabla a partir de los datos de
# una lista de listas:

rios1 = [['Almanzora', 105],
         ['Guadiaro', 79],
         ['Guadalhorce', 154],
         ['Guadalmedina', 51.5]]

print(tabulate(rios1))

'''
------------  -----
Almanzora     105
Guadiaro       79
Guadalhorce   154
Guadalmedina   51.5
------------  -----
'''
```

A continuación, otro ejemplo que imprime los datos de un diccionario. Cada clave del diccionario se corresponde con una columna de la tabla. Los datos a imprimir en cada columna son los valores del diccionario expresados como listas.

Los diccionarios (hasta Python 3.5) son objetos que no ordenan necesariamente las claves siguiendo la secuencia en que son agregadas. Esta peculiaridad afecta al orden en que se imprimen las columnas de una tabla con **tabulate()**, que no tiene porque ser el mismo en cada ejecución. Para obtener columnas ordenadas utilizar objetos OrderedDict.

```
# Imprime tabla a partir de los datos de
# un diccionario. Las claves del diccionario son las
# etiquetas que identifica a los datos (río y longitud)
# y los valores son listas que contienen los nombres
# de los ríos y sus distancias:

rios2 = {'Río': ['Almanzora',
                 'Guadiaro',
                 'Guadalhorce',
                 'Guadalmedina'],
         'Long. (Km.)': [105,
                         79,
                         154,
                         51.5]}

print(tabulate(rios2))

'''
------------  -----
Almanzora     105
Guadiaro       79
Guadalhorce   154
Guadalmedina   51.5
------------  -----
'''
```

Imprimir una tabla con cabecera

La función **tabulate()** incluye el argumento opcional **headers** para imprimir tablas con cabecera. Las etiquetas de una cabecera se pueden obtener:

- de una lista diferente a la de los datos de las columnas,
- de las claves de un diccionario
- o de la primera lista de un conjunto de listas.

Para utilizar las claves de un diccionario asignar '***keys***' al argumento **headers**; y para usar la primera de lista de varias asignar '***firstrow***'.
A continuación, varios ejemplos que muestran las diferentes posibilidades:

```
# Imprime tabla con cabecera (headers=[lista])

print(tabulate(rios1, headers=['Río', 'Long. (Km.)']))

'''
Río            Long. (Km.)
------------  -------------
Almanzora             105
```

gestores de geometría que se utilizan para di...

Archivo

enero 2017 (1)

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

```
Guadiaro               79
Guadalhorce           154
Guadalmedina           51.5
'''

# Imprime tabla con cabecera (headers='keys')

print(tabulate(rios2, headers='keys'))

'''
Río             Long. (Km.)
------------  -------------
Almanzora             105
Guadiaro               79
Guadalhorce           154
Guadalmedina           51.5
'''

# Imprime tabla con cabecera (headers='firstrow')

rios3 = [['Río', 'Long. (Km.)'],
         ['Almanzora', 105],
         ['Guadiaro', 79],
         ['Guadalhorce', 154],
         ['Guadalmedina', 51.5]]

print(tabulate(rios3, headers='firstrow'))

'''
Río             Long. (Km.)
------------  -------------
Almanzora             105
Guadiaro               79
Guadalhorce           154
Guadalmedina           51.5
'''
```

Imprimir una tabla con índice

El argumento opcional **showindex** de **tabulate()** con los valores '***always***' o **True** se emplea para agregar una columna con un índice a una tabla. Por defecto, este índice es numérico aunque también se puede personalizar. Con los objetos pandas.Dataframe el índice se añade por defecto.

Para no incluir un índice con ningún tipo de tabla asignar a **showindex** los valores '***never***' o **False**. Para personalizar el índice, asignar un iterador con los valores a imprimir. A continuación, varios ejemplos.

```
# Imprime tabla con índice numérico:

print(tabulate(rios3, headers='firstrow', showindex=True))

'''
    Río             Long. (Km.)
--  ------------  -------------
 0  Almanzora             105
 1  Guadiaro               79
 2  Guadalhorce           154
 3  Guadalmedina           51.5
'''

# Imprime tabla con índice personalizado, basado en una
# lista que contiene caracteres alfabéticos:


indice = ['a', 'b', 'c', 'd']
print(tabulate(rios3, headers='firstrow', showindex=indice))

'''
    Río             Long. (Km.)
--  ------------  -------------
a   Almanzora             105
b   Guadiaro               79
c   Guadalhorce           154
d   Guadalmedina           51.5
'''
```

Imprimir tablas con distintos formatos

El tercer argumento opcional llamado **tablefmt** define el formato de la tabla a generar.

Los valores de formatos que se pueden asignar son los siguientes:

- '***simple***' (Se corresponde con la opción 'simple_tables' de Pandoc Markdown. Valor por defecto),
- '***plain***' (texto plano. Sin líneas de ningún tipo),
- '***grid***' (EMACS y 'grid_tables' de Pandoc Markdown),
- '***fancy_grid***' (cuadrícula),
- '***psql***' (PostgreSQL),
- '***pipe***' (PHP Markdown Extra, 'pipe_tables' de Pandoc),
- '***orgtbl***' ('org-mode' de Emacs),
- '***jira***' (lenguaje de marcado Atlassian Jira),
- '***rst***' (formato reStructuredText),
- '***mediawiki***' (formato de Wikipedia y de otras sitios basados en MediaWiki),
- '***moinmoin***' (aplicación MoinMoin de wikis),
- '***html***' (tabla en código HTML),
- '***latex***' (formato Latex),
- '***latex_booktabs***' (Latex con el paquete de estilo y espaciado 'booktabs') y
- '***textile***' (formato 'Textile')

A continuación, varios ejemplos que presentan varias tablas con distintos formatos:

```
# "plain" (texto plano. Sin líneas de ningún tipo),

print(tabulate(rios3, headers='firstrow', tablefmt='plain'))

# "simple" (Se corresponde con 'simple_tables' de Pandoc Markdown)

print(tabulate(rios3, headers='firstrow', tablefmt='simple'))

# "grid" (EMACS y 'grid_tables' de Pandoc Markdown)

print(tabulate(rios3, headers='firstrow', tablefmt='grid'))

# "fancy_grid" (cuadrícula)

print(tabulate(rios3, headers='firstrow', tablefmt='fancy_grid'))

'''
Formato: plain

Río             Long. (Km.)
Almanzora             105
Guadiaro               79
Guadalhorce           154
Guadalmedina           51.5


Formato: simple

Río             Long. (Km.)
------------  -------------
Almanzora             105
Guadiaro               79
Guadalhorce           154
Guadalmedina           51.5


Formato: grid

+--------------+---------------+
| Río          |   Long. (Km.) |
+==============+===============+
| Almanzora    |         105   |
+--------------+---------------+
| Guadiaro     |          79   |
+--------------+---------------+
| Guadalhorce  |         154   |
+--------------+---------------+
| Guadalmedina |          51.5 |
+--------------+---------------+


Formato: fancy_grid
```

```
╒═══════════════╤═══════════════╕
│ Río           │   Long. (Km.) │
╞═══════════════╪═══════════════╡
│ Almanzora     │           105 │
├───────────────┼───────────────┤
│ Guadiaro      │            79 │
├───────────────┼───────────────┤
│ Guadalhorce   │           154 │
├───────────────┼───────────────┤
│ Guadalmedina  │            51.5 │
╘═══════════════╧═══════════════╛
'''
```

Alineación y formato

La alineación de los datos es automática pero se puede cambiar con los argumentos **numalign** (alineación de números) y **stralign** (alineación de cadenas de texto) asignando los siguientes valores:

- '***right***': alinea a la derecha,
- '***center***': alinea al centro,
- '***left***': alinea a la izquierda,
- '***decimal***': alinea números con decimales (sólo para **numalign**),
- '***None***': deshabilita la alineación automática.

Los datos numéricos leídos de archivos precedidos o seguidos de saltos de línea (**\n**) u otros caracteres especiales serán tratados con números.

El argumento **floatfmt** se utiliza para cambiar el formato predeterminado de los números con decimales y se basa en el sistema de máscaras de Python.

En el ejemplo que sigue la columna de los nombres de los ríos se alinea al centro y a la de las longitudes se aplica un formato numérico que omite decimales y redondea los valores.

```
print(tabulate(rios3,
               headers='firstrow',
               tablefmt='fancy_grid',
               stralign='center',
               floatfmt='.0f'))

'''
╒═══════════════╤═══════════════╕
│     Río       │   Long. (Km.) │
╞═══════════════╪═══════════════╡
│  Almanzora    │           105 │
├───────────────┼───────────────┤
│   Guadiaro    │            79 │
├───────────────┼───────────────┤
│ Guadalhorce   │           154 │
├───────────────┼───────────────┤
│ Guadalmedina  │            52 │
╘═══════════════╧═══════════════╛
'''
```

Tabulate desde la línea de comandos

Tabulate es también una herramienta para la línea de comandos. Tiene los siguientes argumentos:

```
tabulate [opciones] [ARCHIVO ...]
```

ARCHIVO archivo con datos tabulados. Si se omite '**-**' la lectura de datos se hará desde la entrada estándar (stdin).

Opciones del comando:

- -h, --help : muestra ayuda de tabulate.
- -1, --header : establece que la primera fila sea para la cabecera de la tabla.
- -o ARCHIVO, --output ARCHIVO : la salida se almacenará en el archivo indicado (defecto: stdout).
- -s REGEXP, --sep REGEXP : establece el separador de columnas (defecto: espacio en blanco).
- -F FPFMT, --float FPFMT : define el formato para los números con decimales.

- -f FMT, --formato FMT : establece el formato de salida de la tabla: plain, simple, grid, fancy_grid, pipe, orgtbl, rst, mediawiki, html, latex, latex_booktabs, tsv: (defecto: simple)

El siguiente ejemplo imprime una tabla con el formato '***pipe***' con los datos del archivo '**fotos.txt**'. La primera línea se utiliza para la cabecera:

Datos del archivo 'fotos.txt':

Fotografia fecha hora tamaño\n
Imagen-01.JPG 10-12-2016 12:01:01 23445\n
Imagen-02.JPG 10-12-2016 12:12:12 143454\n
Imagen-03.JPG 10-12-2016 12:34:23 64974\n
Imagen-04.JPG 10-12-2016 12:54:04 212989\n
Imagen-05.JPG 10-12-2016 12:58:32 199100\n

Comando:

```
$ tabulate -1 -f pipe fotos.txt
```

Salida que se obtiene:

Fotografia	fecha	hora	tamaño
Imagen-01.JPG	10-12-2016	12:01:01	23445
Imagen-02.JPG	10-12-2016	12:12:12	143454
Imagen-03.JPG	10-12-2016	12:34:23	64974
Imagen-04.JPG	10-12-2016	12:54:04	212989
Imagen-05.JPG	10-12-2016	12:58:32	199100

Ir al índice del tutorial de Python

Publicado por Pherkad en 14:55

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua