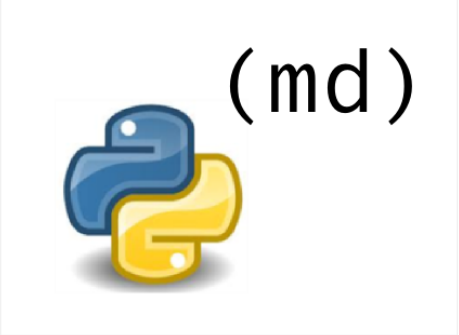
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

jueves, 18 de septiembre de 2014

Markdown para Python (y II)



El modulo Markdown

Este módulo es una implementación en Python del lenguaje **Markdown**, del que tratamos sus fundamentos en el capítulo anterior. Es compatible, casi completamente, con la referencia oficial del propio lenguaje.

Los objetivos que persiguen los responsables del proyecto son:

- Mantener al día el módulo para Python 2.x y Python 3.x (con una capa CLI opcional) apto para ser utilizado en entornos de servidores web; como analizador/convertidor de Markdown que cumple las reglas de sintaxis y el comportamiento del desarrollo original en Perl (*markdown.pl*), con muy pocas diferencias.
- Facilitar una **API** para desarrollar extensiones que permitan cambiar y/o extender el comportamiento del analizador/convertidor.

Además, de soportar la sintaxis básica de **Markdown**, cuenta con otras características:

- Soporte Internacional: acepta la entrada en cualquier idioma soportado por **Unicode** incluyendo texto bidireccional.
- Extensiones: existen varias extensiones que permiten cambiar y/o extender la sintaxis de base y una API pública para desarrollar extensiones propias.
- Formatos de salida: básicamente, el módulo puede convertir código Markdown a HTML y XHTML. De momento los formatos de salida soportados (*output_format*) son los siguientes: ("xhtml": salida con la última versión compatible XHTML (actualmente XHTML 1.1); "html4": salida en HTML 4; "html5": salida en HTML con etiquetas estilo HTML 5; "html": Salida con la última versión compatible HTML.
- Línea de comandos: el módulo puede utilizarse en los programas como cualquier otro módulo Python y, además, como veremos más adelante con varios ejemplos, es posible ejecutar scripts desde la línea de comandos.

Instalación

Aunque existen otras formas de instalar el módulo **Markdown** tomaremos el camino más fácil instalando con **pip**:

- En GNU/Linux:

```
$ sudo pip3 install markdown
```

- En Windows:

```
> pip3 install markdown
```

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...

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute

Uso básico del módulo

El módulo proporciona las funciones "*markdown.markdown*" y "*markdown.markdownFromFile*" de la clase **markdown.Markdown**.

Para ver cómo se utiliza dicho módulo mostraremos algunos ejemplos, a continuación.

(1) En el primer ejemplo se convierte una cadena de texto Markdown a HTML:

```
import markdown
texto = """ \
##Lenguaje de marcas ligero \
--------------------------- \

Un **lenguaje de marcas ligero** es un tipo de formato de \
texto más o menos estandarizado, que ocupa poco espacio y \
es fácil de editar con un editor de texto.                 \

Fuente: \
[Wikipedia](https://es.wikipedia.org/wiki/Lenguaje_de_marcas_ligero) \
"""
html = markdown.markdown(texto)
print(html)
```

Salida obtenida (en HTML):

```
<h2>Lenguaje de marcas ligero</h2>
<hr />
<p>Un <strong>lenguaje de marcas ligero</strong> es un tipo de formato
de texto más o menos estandarizado, que ocupa poco espacio y
es fácil de editar con un editor de texto.

</p>
<p>Fuente:
<a href="https://es.wikipedia.org/wiki/Lenguaje_de_marcas_ligero">
Wikipedia</a> </p>
```

(2) En el siguiente ejemplo se lee un archivo de texto Markdown codificado en "utf-8" y se muestra su correspondiente salida en formato HTML.

```
import markdown, codecs
archivo_entrada = codecs.open("archivo.txt",
                              mode="r",
                              encoding="utf-8")
texto = archivo_entrada.read()
html = markdown.markdown(texto)
print(html)
```

Para obtener la salida equivalente en formato HTML 5:

```
html5 = markdown.markdown(texto,
                          output_format="html5")
print(html5)
```

(3) En este caso la salida codificada en "utf-8" se escribe en un archivo HTML:

```
archivo_salida = codecs.open("archivo.html",
                             "w",
                             encoding="utf-8",
                             errors="xmlcharrefreplace")

archivo_salida.write(html)
```

Con la función "***markdown.markdownFromFile***" se puede leer código Markdown de un archivo de entrada, convertirlo al formato deseado y escribirlo en un archivo de salida.

simultáneamente varias operaciones en el mismo espacio de proceso se...

Archivo

septiembre 2014 (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

Línea de comandos

A continuación, varios ejemplos de ejecución desde la línea de comandos:

(1) Convertir texto Markdown a HTML:

```
$ echo "**Markdown** ***Ligero***" | python -m markdown
```

Salida:

```
<p><strong>Markdown</strong> <strong><em>Ligero</em></strong></p>
```

(2) Convertir el texto Markdown de un fichero de entrada y guardar en un archivo HTML la salida obtenida:

```
$ python -m markdown entrada.txt > salida.html
```

(3) Ver otras posibilidades de ejecución consultando la ayuda del propio módulo:

```
$ python -m markdown --help
```

(4) Cargar una o varias extensiones Markdown (accesible/s desde PYTHONPATH):

```
$ python -m markdown -x ruta.del.mod1 -x ruta.del.mod2 ent.txt
```

(5) Utilizar directamente desde la línea de comandos los ejecutables "***markdown_py***" en Linux y "***markdown_py.bat***" en Windows.

- En Linux:

Localizar el archivo "***markdown_py***" con el comandos **which**, comprobar que tiene los atributos de ejecución (asignarlos si fuera necesario) y, finalmente, verificar que la ruta del archivo se encuentra en el **path** del sistema. Si todo es correcto podremos ejecutar:

```
$ echo "**Hola**" | markdown_py
```

que convierte el texto Markdown de **echo** a HTML:

```
<p><strong>Hola</strong></p>
```

- En Windows:

Añadir la ruta del archivo "***markdown_py.bat***" al **path** del sistema y si todo es correcto se podrán ejecutar líneas como la que sigue que convierte texto Markdown a HTML.

```
> markdown_py entrada.txt >salida.html
```

Publicado por Pherkad en 14:50

Etiquetas: Markdown

Entrada más reciente Inicio Entrada antigua