Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

domingo, 14 de diciembre de 2014

Fundamentos para procesar imágenes con Pillow (I)

Pillow es una variante (o fork) de la popular librería **PIL** (Python Image Library) que permite procesar con facilidad imágenes con Python 2.x/3.x. El proyecto lo inició **Alex Clark** cuando PIL se quedó sin desarrollo a finales del año 2009. Actualmente, Pillow es mantenido con la ayuda de otros colaboradores.

Con Pillow podemos consultar información básica de una imagen como su tamaño, el formato que tiene (jpg, png, gif, etc.), el tipo de imagen (bits/pixel, BN/color, etc.) y, también, es posible modificar su aspecto realizando operaciones para cambiar su tamaño, recortar un área, girar, aplicar filtros y efectos, convertir el tipo de imagen y su formato, etc.

Instalación

Comprobar si tenemos alguna versión de Pillow instalada:

$ pip3 show Pillow

Name: Pillow
Version: 2.3.0
Location: /usr/lib/python3/dist-packages

Instalar Pillow en un equipo con Windows/Linux con Pip:

$ pip3 install Pillow

Instalando en una red en la que se utiliza proxy:

$ pip3 install Pillow --proxy http://usuario:passw@servidor:puerto

Abrir imágenes

Abrir y mostrar una imagen:

```
from PIL import Image
imagen = Image.open("amapolas.jpg")
imagen.show()
```

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

Abrir y mostrar una imagen capturando excepciones:

```
from PIL import Image
try:
    imagen = Image.open("amapolas.jpg")
    imagen.show()
except:
    print("No ha sido posible cargar la imagen")
```

Consultar información de las imágenes

Formato:

print(imagen.format) # JPEG

Tamaño:

print(imagen.size) # Obtiene tupla con pixels horizontal/vertical (800, 600)

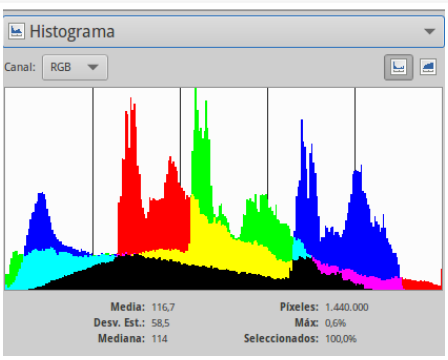
> El módulo **PIL** utiliza un sistema en la que la coordenada (0, 0) se encuentra en la esquina superior izquierda de la imagen. Si una imagen tiene un tamaño de 800x600 píxeles la coordenada (800, 600) se encontrará en la esquina inferior derecha de la imagen.

Tipo de imagen:

print(imagen.mode) # Tipo de imagen: RGB, CMYK, L, P, etc.

Histograma:

print(imagen.histogram) # Obtiene datos del histograma de una imagen



El **histograma** de una imagen representa la frecuencia relativa de los niveles de gris o de los colores básicos (rojo, azul, verde) de la imagen.

Una de las técnicas básicas de retoque de fotos consiste en modificar el histograma para aumentar el contraste de las imágenes.

Otros métodos para consultar información de una imagen: imagen.**info**, imagen.**palette**.

Edición básica (tamaños, rotar, girar...) y guardar imágenes

Cortar una región de una imagen y guardarla:

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

Archivo

diciembre 2014 (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

La región a obtener se define con una tupla con las coordenadas siguientes:(izquierda, superior, derecha, inferior)

En el ejemplo siguiente de la imagen original, que tiene un tamaño de 800x600 píxeles, vamos a "desechar" un margen de 100 píxeles alrededor de la misma; quedándonos pues con la región central que tendrá un tamaño de 600x400. Con la función ***crop()*** obtendremos la imagen resultante que será guardada en el sistema de ficheros con la función ***save()***.

```
# Define tupla con región
caja = (100, 100, 700, 500)

# Obtener de la imagen original la región de la caja
region = imagen.crop(caja)

region.show()  # Mostrar imagen de la region
region.size   # Mostrar tamaño de imagen final 600x400

# Guarda la imagen obtenida con el formato JPEG.
region.save("region.jpg")

# Guarda la imagen obtenida con el formato PNG.
region.save("region.png")
```

Cambiar el tamaño de una imagen:

Podemos cambiar el tamaño de una imagen para hacerla más grande o más pequeña. En este caso como la imagen original tiene una proporción de 4x3 vamos a reducirla manteniendo estas proporciones. Como la imagen original tiene un tamaño de 800x600 para hacerla 4 veces más pequeña indicamos el nuevo tamaño 400x300 en la función ***resize()***.

```
# Obtener imagen con el tamaño indicado
reducida = imagen.resize((400, 300))

# Mostrar imagen
reducida.show()

# Guardar imagen obtenida con el formato JPEG
reducida.save("reducida.jpg")
```

Rotar una imagen:

Para rotar una imagen utilizaremos el método ***rotate()*** y tendremos que indicar el número de grados teniendo en cuenta que el giro seguirá el sentido contrario a las agujas del reloj. En el ejemplo siguiente la imagen obtenida (a partir de "reducida.jpg") estará girada 45 grados.

```
# Obtener imagen girada 45º
girada1 = reducida.rotate(45)

# Mostrar imagen
girada1.show()
```

```
# Guardar la imagen con el formato JPEG
girada1.save("girada.jpg")
```

Trasponer una imagen:

Otra posibilidad para rotar una imagen es con el método ***transpose()***. El módulo **Image** provee funciones para girar una imagen con rotaciones de grados múltiples de 90 (**ROTATE_90**, **ROTATE_180**, **ROTATE_270**) y otras que permiten darles la vuelta (**FLIP_LEFT_RIGHT**, **FLIP_TOP_BOTTOM**).

```
# Obtener imagen girada 90º
girada2 = reducida.transpose(Image.ROTATE_90)
girada2.show()
girada2.save("girada2.jpg")
```

```
# Voltear imagen de arriba a abajo
volteada = reducida.transpose(Image.FLIP_TOP_BOTTOM)
volteada.show()
volteada.save("volteada.jpg")
```

Girar una región dentro de una imagen:

En el siguiente ejemplo después de obtener una imagen girada 90º la pegamos en la imagen original con la función ***paste()***.

```
# Abrir la imagen "pantera.jpg"
imagen = Image.open("pantera.jpg")

# Mostrar tanaño de la imagen:
imagen.size  # 800x600

# Definir tupla con las coordenadas de la región
caja = (200, 100, 600, 500)

# Obtener región
region = imagen.crop(caja)

# Girar región 90º
region = region.transpose(Image.ROTATE_90)

# Pegar región girada en la imagen original
imagen.paste(region, caja)

# Mostrar imagen final y guardar
imagen.show()
imagen.save("regiongirada90.jpg")
```

Unir imágenes:

La función ***paste()*** la podemos utilizar para unir varias imágenes en una. En el siguiente ejemplo se utiliza el método new() para crear una imagen vacía "RGB" con un tamaño de 800x600 píxeles y con el fondo negro. Después se abren dos imágenes con el mismo tamaño 400x600 y se pegan horizontalmente, una a continuación de la otra, sobre la primera.

```
final = Image.new("RGB",(800,600),"black")
imagen1 = Image.open("nevando2-izq.jpg")
imagen2 = Image.open("nevando2-der.jpg")
final.paste(imagen1, (0,0))
final.paste(imagen2, (399,0))
final.show()
final.save("nevando2-unidas.jpg")
```

Si utilizamos dos imágenes transparentes del mismo tamaño y pegamos una en la coordenada (0, 0) de la otra obtentrámos una imagen mezclada.

Crear miniaturas:

La función **thumbnail()** se utiliza para crear miniaturas de las imágenes:

```
imagen = Image.open("amapolas.jpg")
miniatura = (160, 120)
imagen.thumbnail(miniatura)
imagen.save("miniatura.jpg")
```

Continuar en: Fundamentos para procesar imágenes con Pillow (II)

Ir al índice del tutorial de Python

Publicado por Pherkad en 4:59

Etiquetas: Pillow

Entrada más reciente Inicio Entrada antigua