Generando códigos QR con Python.

Publicado el sábado 13 de junio de 2015.

Actualizado el sábado 16 de noviembre de 2019.

Introducción.

En el día de hoy aprendimos que se pueden generar "QR codes" (la cual es la abreviatura, en inglés, de "Quick Response Code") con un paquete para el lenguaje Python. Fueron creados en Japón para el manejo de piezas y manufactura de vehículos de la marca Toyota, a fin de agilizar grandemente los inventarios. Explicar cómo funcionan merece su entrada completa y aparte, aquí vamos a explicar brevemente cómo instalarlo y usarlo con Python.

Agradecimiento.

Agradecemos de antemano a los programadores de software libre de "Industrias Diana" cuya exposición fue brillante en el "Décimo Primer Congreso de Software Libre". Dicha exposición abarcó varios temas acerca del proceso y manejo de la información, entre otros la comunicación con impresoras fiscales y la generación de códigos QR, caso que nos ocupa hoy. **Muchas gracias por compartir el conocimiento, cumplimos con seguir difundiendolo.**

https://twitter.com/gerardomaduro/status/609788744724578304

https://twitter.com/CNSLCarabobo/status/609788377223819265

Instalación del módulo QR en Python.

Lo primero que debemos hacer es abrir una ventana terminal (¿no sabe cómo? visite nuestro pequeño tutorial sobre *shell*) y verificar si tenemos Python instalado en nuestro ordenador:

```
python -V
```

Podrán visualizar algo como esto:

Y a continuación escribimos la orden para instalar el módulo necesario para los **códigos QR:**

```
sudo apt-get install python-qrcode
```

La mayoría de las distribuciones GNU/Linux traen Python instalado por defecto, acá usamos Ubuntu en este ejemplo:

```
jimmy@KEVIN:~$ sudo apt-get install python-qrcode
[sudo] password for jimmy:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  kde-l10n-engb kde-l10n-es linux-headers-3.13.0-48
  linux-headers-3.13.0-48-generic linux-headers-3.13.0-49
  linux-headers-3.13.0-49-generic linux-headers-3.13.0-51
  linux-headers-3.13.0-51-generic linux-headers-3.13.0-52
  linux-headers-3.13.0-52-generic linux-image-3.13.0-48-generic
  linux-image-3.13.0-49-generic linux-image-3.13.0-51-generic
  linux-image-3.13.0-52-generic linux-image-extra-3.13.0-48-generic
  linux-image-extra-3.13.0-49-generic linux-image-extra-3.13.0-51-generic
  linux-image-extra-3.13.0-52-generic linux-image-generic
  linux-signed-image-3.13.0-48-generic linux-signed-image-3.13.0-49-generic
  linux-signed-image-3.13.0-51-generic linux-signed-image-3.13.0-52-generic
Use 'apt-get autoremove' to remove them.
Se instalarán los siguientes paquetes NUEVOS:
  python-qrcode
0 actualizados, 1 se instalarán, 0 para eliminar y 20 no actualizados.
Necesito descargar 19,9 kB de archivos.
Se utilizarán 116 kB de espacio de disco adicional después de esta operación.
Des:1 http://mirror.cogentco.com/pub/linux/ubuntu/ trusty/universe python-qrcode
 all 4.0.1-2 [19,9 kB]
Descargados 19,9 kB en 0seg. (22,0 kB/s)
Seleccionando el paquete python-qrcode previamente no seleccionado.
(Leyendo la base de datos ... 419284 ficheros o directorios instalados actualmen
te.)
Preparing to unpack .../python-qrcode_4.0.1-2_all.deb ...
Unpacking python-qrcode (4.0.1-2) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Configurando python-qrcode (4.0.1-2) ...
Processing triggers for python-support (1.0.15) ...
jimmy@KEVIN:~$ qr "Some text" > test.png
jimmy@KEVIN:~$ ls
```

Una vez instalado lo probamos, tal cual dice el ejemplo publicado en "Python Software Fundation" qrcode 5.1, es decir, crear una etiqueta QR con una información en un archivo de imagen en formato .png:

```
qr "Some test" > test.png
```

¡Y listo! (más tardamos explicando la introducción al tema que lo que se tardó instalando).

Uso de la librería QR.

Claro está, estos códigos les debemos dar usos útiles, en nuestro caso la imagen que encabeza esta entrada es la dirección URL de nuestro blog https://www.ks7000.net.ve

En realidad el comando **qr** es un guión "script" ubicado en /usr/bin/qr que nos facilita la realización de códigos QR de una manera rápida y sencilla. Si queremos avanzar un poco más podemos usar el siguiente código en lenguaje Python, el cual explicaremos luego línea por línea:

```
import qr imagen = qrcode.make('https://www.ks7000.net.ve') imagen.save('
ks7000_url_qr.png', 'png')
```

- La primera línea carga en memoria la librería necesaria y capaz de generar los códigos QR.
- La segunda línea llama al método "hacer" de la librería **qr** para "dibujar" la línea que está como argumento (entre paréntesis y entrecomillado).
- La tercera línea guarda en la carpeta desde donde se llamó a la aplicación y el primer argumento es el nombre del archivo y el segundo argumento el tipo de imagen.

Dichos códigos QR tienen una característica particular: utiliza un algoritmo de correción de errores basado en código Reed-Solomon de hasta un 30% (Y NO DE 50% COMO PENSABAMOS -¿de dónde sacamos eso?) y lo podemos especificar de la siguiente manera:

```
import qrcode qr = qrcode.QRCode(   error_correction=qrcode.constants.ERR
OR_CORRECT_H,   version=1,   box_size=10,   border=4, ) qr.add_data('http
s://www.ks7000.net.ve') qr.make(fit=True) imagen = qr.make_image() imagen
.save('ks7000_url_qr.png', 'png')
```

- **ERROR_CORRECT_H** corresponde a un 30% de error de lectura (le coloca información redundante previendo el daño de la etiqueta donde va a ir el código QR). Es el error máximo soportado, los otros valores son -por letra-: "Q", 25%; "M", 15%; "L", 7%. Todo depende en dónde vamos a colocar el código QR generado, recordemos que se inventaron para manejo de inventario de productos que debido a la manipulación y traslado puede dañar las etiquetas autoadhesivas que hagamos
- **version**: un valor que va de 1 a 40 y controla el tamaño de la matriz de cuadrados. Aquí le

colocamos 1, valor mínimo, pero como más adelante especificamos un valor en box_size deberemos usar el comando fit para que el software nos ajuste a la matriz de tamaño correcta.
- **box_size**: el tamaño en píxeles que tendrá la caja.
- **border**: el número de cuadros que tendrá el borde, cosa importante ya que son cuatro lados en blanco que permite ubicar la lectura. La cantidad mínima de cuadros es 4.

También observamos que el comando **qr.make()** lleva un solo argumento que indica que la imagen se ajuste automáticamente a los valores establecidos (ver anterior explicación de parámetros) y el comando **qr.make_image()** que lleva a cabo la conversión en sí de la imagen. Todos estos parámetros nos permiten generar códigos según necesitemos: tamaño, seguridad de lectura, etc.

Elementos de un "QR Code":

Esta figura, en formato de "Scalable Format Graphic" (SVG) tomada de la Wikipedia bajo licencia "Creative Commons" (modificamos el archivo y le hicimos traducción al idioma castellano) nos muestra dónde están ubicados, por ejemplo los niveles de correción para errores de lectura en el cuadro guía superior izquierdo (esos tres cuadrados grandes a su vez con cuadrados insertados se utilizan para ubicar y rotar la imagen en memoria antes de proceder a descodificarlas, permite la lectura en cualquier dirección y es muy útil para productos grandes y pesados, tomáis la foto como se encuentre e igual se lee):

[SVG: Información de formato "QR code"]

Fuentes consultadas:

En idioma francés:

- Herramienta en línea para realizar etiquetas, afiches e imágenes de QR para los dominios web.

En idioma inglés:

- «The word "QR Code" is registered trademark of DENSO WAVE INCORPORATED»
- "History of QR Code" at Denso Wave Incorporated.
- "qrcode 5.3" by Lincoln Loop at Python.org
- "qrcode" by Lincoln Loop at GitHub.com
- «ZXing ("zebra crossing") is an open-source, multi-format 1D/2D barcode image processing

library implemented in Java, with ports to other languages.» at GitHub.com
- "QRCode for JavaScript" by Kazuhiko Arase at D-Project.com
- "pyqrnative" at Code.Google.com
- "QR code" at Wikipedia.com

En idioma castellano:

- «GQRCode y usos de los códigos QR en Ubuntu, Linux Mint, etc» por Atareao.
- «Cómo generar códigos QR en KDE Plasma» por JavierInSitu.