Buscar

Distribuir aplicaciones Python

Una vez terminemos con el desarrollo de nuestra nueva aplicación es conveniente empaquetarla de forma que sea sencillo para los usuarios instalarla, y para nosotros distribuirla.

En Python existen dos módulos principales para este cometido: `distutils`, que es parte de la librería estándar y era el método más utilizado hasta hace poco, y `setuptools`, que extiende la funcionalidad de `distutils` y es cada vez más popular.

En este capítulo veremos el funcionamiento de ambas herramientas, y terminaremos explicando cómo crear ejecutables .exe para Windows a partir de nuestro programa en Python.

distutils

Todo programa distribuido con `distutils` contiene un script llamado por convención setup.py, que se encarga de instalar la aplicación llamando a la función `setup` de `distutils.core`. Esta función tiene montones de argumentos, que controlan, entre otras cosas, cómo instalar la

Destinados a describir la aplicación tenemos los siguientes argumentos:

- `name`: El nombre del paquete.
- `version`: El número de versión.
- `description`: Una línea describiendo el paquete.
- `long_description`: Descripción completa del paquete.
- `author`: Nombre del autor de la aplicación.
- `author_email`: Correo electrónico del autor.
- `maintainer`: Nombre de la persona encargada de mantener el paquete, si difiere del autor.
- `maintainer_email`: Correo de la persona encargada de mantener el paquete, si difiere del autor.
- `url`: Web de la aplicación.
- `download_url`: Url de la que descargar la aplicación.
- `license`: Licencia de la aplicación

También tenemos argumentos que controlan los archivos y directorios que deben instalarse, como son `packages`, `py_modules`, `scripts` y `ext_modules`.

El parámetro `scripts`, que es una lista de cadenas, indica el nombre del módulo o módulos principales, es decir, los que ejecuta el usuario. Si nuestra aplicación consistiera, por ejemplo, en un solo script ejemplo.py, el código de setup.py podría tener un aspecto similar al siguiente:

view plain copy to clipboard print ?

```
01. from distutils.core import setup
02.
03. setup(name="Aplicacion de ejemplo",
04.       version="0.1",
05.       description="Ejemplo del funcionamiento de distutils",
06.       author="Raul Gonzalez",
07.       author_email="zootropo en gmail",
08.       url="http://mundogeek.net/tutorial-python/",
09.       license="GPL",
10.       scripts=["ejemplo.py"]
11. )
```

Si hemos escrito otros módulos para ser utilizados por el script principal, estos se indican mediante el parámetro `py_modules`. Por ejemplo, supongamos que la aplicación consiste en un script principal ejemplo.py, y un módulo de apoyo apoyo.py:

view plain copy to clipboard print ?

```
01. from distutils.core import setup
02.
03. setup(name="Aplicacion de ejemplo",
04.       version="0.1",
05.       description="Ejemplo del funcionamiento de distutils",
06.       author="Raul Gonzalez",
07.       author_email="zootropo en gmail",
08.       url="http://mundogeek.net/tutorial-python/",
09.       license="GPL",
10.       scripts=["ejemplo.py"],
11.       py_modules=["apoyo"]
12. )
```

Para instalar paquetes Python (directorios que contienen varios módulos y un archivo __init__.py) usaríamos el parámetro `packages`. Si además del módulo ejemplo.py quisiéramos instalar los paquetes `gui` y `bbdd`, por ejemplo, haríamos algo así:

```
01. from distutils.core import setup
02.
03. setup(name="Aplicacion de ejemplo",
04.       version="0.1",
05.       description="Ejemplo del funcionamiento de distutils",
06.       author="Raul Gonzalez",
07.       author_email="zootropo en gmail",
08.       url="http://mundogeek.net/tutorial-python/",
09.       license="GPL",
10.       scripts=["ejemplo.py"],
11.       packages=["gui", "bbdd"]
12. )
```

`ext_modules`, por último, sirve para incluir extensiones que utilice el programa, en C, C++, Fortran, ...

Veamos ahora cómo se utilizaría el archivo setup.py una vez creado.

Al ejecutar el comando

python setup.py install

los módulos y paquetes especificados por `py_modules` y `packages` se instalan en el directorio Lib de Python. Los programas indicados en scripts, se copian al directorio Scripts de Python.

Una vez hemos comprobado que la aplicación se instala correctamente, procedemos a crear archivos mediante los que distribuir la aplicación a los usuarios. Para crear archivos con el código fuente se utiliza la opción `sdist` de setup.py, que crea por defecto un archivo tar.gz en Unix y un zip en Windows.

python setup.py sdist

Sin embargo se puede utilizar `--formats` para especificar el formato o formatos que queramos generar

bztar	.tar.bz2
gztar	.tar.gz
tar	.tar
zip	.zip
ztar	.tar.Z

Para crear un archivo tar.bz2, un tar.gz y un zip, por ejemplo, se utilizaría la siguiente orden:

python setup.py sdist –formats=bztar,gztar,zip

Para generar un archivo de distribución binaria, se usa la opción `bdist`:

python setup.py bdist

Los formatos que soporta `bdist` son los siguientes:

rpm	RPM

ztar	.tar.Z
tar	.tar
wininst	Instalador Windows
zip	.zip

Para crear un archivo rpm y un instalador de Windows, por ejemplo, escribiríamos:

python setup.py bdist –formats=wininst,rpm

También es posible crear otros tipos de archivos de distribución utilizando scripts que extienden distutils, como es el caso de los paquetes deb mediante el script stdeb (http://stdeb.python-hosting.com/)

setuptools

setuptools extediende distutils añadiendo una serie de funcionalidades muy interesantes: introduce un nuevo formato de archivo para distribución de aplicaciones Python llamado *egg*, se encarga de buscar todos los paquetes que deben instalarse y añadir las posibles dependencias, permite instalar paquetes de PyPI con un solo comando, etc.

Además, como setuptools se basa en distutils, un script de instalación básico utilizando setuptools es prácticamente igual a su equivalente con distutils. Tan sólo cambiaría la sentencia de importación.

view plain copy to clipboard print ?

```
01. from setuptools import setup
02.
03. setup(name="Aplicacion de ejemplo",
04.       version="0.1",
05.       description="Ejemplo del funcionamiento de distutils",
06.       author="Raul Gonzalez",
07.       author_email="zootropo en gmail",
08.       url="http://mundogeek.net/tutorial-python/",
09.       license="GPL",
10.       scripts=["ejemplo.py"],
11. )
```

El único inconveniente que podríamos encontrar al uso de setuptools es que no está incluido por defecto en Python 2.5, aunque es probable que esto cambie en próximas versiones debido a su gran uso. Pero los desarrolladores de setuptools han pensado en todo, e incluso esto no debería suponer ningún problema, ya que con un mínimo esfuerzo por nuestra parte podemos hacer que setuptools se descargue e instale automáticamente en la máquina del usuario si este no se encuentra ya en el sistema. Basta distribuir con nuestro paquete un pequeño módulo extra ez_setup.py que viene incluido por defecto con setuptools (http://peak.telecommunity.com/dist/ez_setup.py) y llamar a la función `use_setuptools` del módulo al inicio de setup.py:

view plain copy to clipboard print ?

```
01. from ez_setup import use_setuptools
02. use_setuptools()
```

```
setup(name="Aplicacion de ejemplo",
      version="0.1",
      description="Ejemplo del funcionamiento de distutils",
      author="Raul Gonzalez",
      author_email="zootropo en gmail",
      url="http://mundogeek.net/tutorial-python/",
      license="GPL",
      scripts=["ejemplo.py"],
)
```

Veamos ahora con más detenimiento algunos de los cambios y novedades que introduce setuptools.

Integración con PyPI

Al estilo de CPAN en Perl setuptools permite instalar de forma fácil y sencilla los paquetes pertenecientes a PyPI, el Índice de Paquetes Python (http://pypi.python.org/pypi), así como subir nuestros propios paquetes.

PyPI cuenta en el momento de escribir estas líneas con 4782 paquetes, por lo que poder instalar los paquetes de este repositorio con un simple comando supone una ayuda muy a tener en cuenta.

Instalar un paquete de PyPI es tan sencillo como pasar al comando `easy_install` el nombre del paquete a instalar

easy_install docutils
Searching for docutils
Reading http://pypi.python.org/simple/docutils/
Reading http://docutils.sourceforge.net/
Best match: docutils 0.5
Downloading http://prdownloads.sourceforge.net/docutils/docutils-0.5.tar.gz?download
Processing docutils-0.5.tar.gz
Running docutils-0.5/setup.py -q bdist_egg –dist-dir /tmp/easy_install-wUAyUZ/docutils-0.5/egg-dist-tmp-kWkkkv
“optparse” module already present; ignoring extras/optparse.py.
“textwrap” module already present; ignoring extras/textwrap.py.
zip_safe flag not set; analyzing archive contents…
docutils.writers.newlatex2e.__init__: module references __file__
docutils.writers.pep_html.__init__: module references __file__
docutils.writers.html4css1.__init__: module references __file__
docutils.writers.s5_html.__init__: module references __file__
docutils.parsers.rst.directives.misc: module references __file__
Adding docutils 0.5 to easy-install.pth file
Installing rst2pseudoxml.py script to /usr/bin
Installing rst2html.py script to /usr/bin
Installing rst2latex.py script to /usr/bin
Installing rst2s5.py script to /usr/bin
Installing rst2newlatex.py script to /usr/bin
Installing rstpep2html.py script to /usr/bin
Installing rst2xml.py script to /usr/bin

Finished processing dependencies for docutils

Poder subir nuestros paquetes a PyPI requiere de un proceso un poco más laborioso. Primero registramos los detalles de nuestra aplicación en PyPI mediante la opción `register` del script setup.py, el cuál nos preguntará por nuestro nombre de usuario, contraseña y correo electrónico si no tenemos cuenta en PyPI, o nombre de usuario y contraseña si nos registramos anteriormente:

```
python setup.py register
running register
running egg_info
creating Aplicacion_de_ejemplo.egg-info
writing Aplicacion_de_ejemplo.egg-info/PKG-INFO
writing top-level names to Aplicacion_de_ejemplo.egg-info/top_level.txt
writing dependency_links to Aplicacion_de_ejemplo.egg-info/dependency_links.txt
writing manifest file 'Aplicacion_de_ejemplo.egg-info/SOURCES.txt'
reading manifest file 'Aplicacion_de_ejemplo.egg-info/SOURCES.txt'
writing manifest file 'Aplicacion_de_ejemplo.egg-info/SOURCES.txt'
We need to know who you are, so please choose either:
1. use your existing login,
2. register as a new user,
3. have the server generate a new password for you (and email it to you), or
4. quit
Your selection [default 1]: 1
Username: zootropo
Password:
Server response (200): OK
I can store your PyPI login so future submissions will be faster.
(the login will be stored in /home/zootropo/.pypirc)
Save your login (y/N)?y
```

Para crear y subir una distribución con el código fuente de nuestra aplicación se utiliza la opción `sdist upload`:

```
python setup.py sdist upload
```

También podríamos crear y subir un egg (un formato de archivo para distribuir aplicaciones Python que veremos en la próxima sección) utilizando la opción `bdist_egg upload`:

```
python setup.py bdist_egg upload
```

O combinar los tres pasos en un solo comando:

```
python setup.py register sdist bdist_egg upload
```

Una vez subido el paquete cualquier persona podría instalarlo en su sistema utilizando easy_install, de la misma forma que cualquier otro paquete de PyPI:

```
easy_install mi-paquete
```

Los *eggs* (huevo en inglés) son archivos de extensión .egg mediante los que distribuir aplicaciones en Python. Serían algo así como el equivalente a los archivos .jar del mundo Java. Son multiplataforma, permiten manejar dependencias, y permiten instalar distintas versiones del mismo paquete.

La forma más sencilla de instalar aplicaciones distribuidas como archivos egg es mediante el comando easy_install, el cuál comentamos brevemente en el punto anterior al hablar sobre su uso para instalar paquetes de PyPI. Para instalar un archivo egg no tenemos más que pasarle el nombre del archivo al comando easy_install:

easy_install mi-aplicacion.egg

o bien podemos pasarle la URL de la que descargar el egg:

easy_install http://mundogeek.net/mi-aplicacion.egg

Para construir nuestros propios eggs podemos utilizar el comando `bdist_egg` de setup.py, de forma similar a la manera en que construíamos paquetes RPM o instaladores para Windows con distutils:

python setup.py bdist_egg

Otros cambios destacables

Uno de los cambios más interesantes es la incorporación de un nuevo argumento para la función setup llamado `install_requires`, que consiste en una cadena o lista de cadenas que indica los paquetes de los que depende la aplicación. Si nuestra aplicación necesitara tener instalado el paquete apoyo para poder ejecutarse, por ejemplo, escribiríamos lo siguiente:

view plain copy to clipboard print ?

```
01. install_requires = ["apoyo"]
```

Y de esta forma, easy_install se encargaría de buscar e instalar el paquete si fuera necesario, bien en PyPI, o en cualquier otro repositorio indicado por el parámetro `dependency_links`.

Además podemos especificar que se necesita una versión concreta del paquete requerido, que sea mayor o menor que una cierta versión, o que no se trate de una versión determinada utilizando operadores relacionales (==, !=, <, <=, >, >=):

view plain copy to clipboard print ?

```
01. install_requires = ["apoyo >= 1.0 < 2.0"]
```

También existen argumentos similares para declarar paquetes que deben instalarse para poder ejecutar el script de instalación (`setup_requires`), para poder ejecutar las posibles pruebas incluídas con el paquete (`tests_require`) y para conseguir funcionalidades adicionales (`extras_require`, que consiste en este caso en un diccionario).

setuptools incluye también atajos útiles, como la función `find_packages()` que nos evita tener que listar todos y cada uno de los paquetes que utiliza nuestro script en el parámetro `packages`,

```
from ez_setup import use_setuptools
use_setuptools()

from setuptools import setup, find_packages

setup(name="Aplicacion de ejemplo",
      version="0.1",
      description="Ejemplo del funcionamiento de distutils",
      author="Raul Gonzalez",
      author_email="zootropo en gmail",
      url="http://mundogeek.net/tutorial-python/",
      license="GPL",
      scripts=["ejemplo.py"],
      packages = find_packages()
)
```

Crear ejecutables .exe

Tanto en Mac OS como en la mayor parte de las distribuciones Linux el intérprete de Python está instalado por defecto, por lo que los usuarios de estos sistemas no tienen mayor complicación a la hora de instalar y ejecutar aplicaciones escritas en Python.

En el caso de Windows, esto no es así, por lo que sería interesante que los usuarios de este sistema operativo no tuvieran que instalar el intérprete de Python. También sería interesante que nuestro programa consistiera en un archivo .exe en lugar de uno o varios archivos .py, para simplificar las cosas.

Todo esto lo podemos lograr gracias a py2exe, una extensión para distutils que, como su nombre indica, permite crear ejecutables para Windows a partir de código Python, y que permite ejecutar estas aplicaciones sin necesidad de tener instalado el intérprete de Python en el sistema.

Py2exe funciona examinando nuestro código fuente en busca de los módulos y paquetes que utilizamos, compilándolos y construyendo un nuevo archivo que incluye estos archivos y un pequeño intérprete de Python integrado.

Para probar el funcionamiento de py2exe creemos un pequeño programa ejemplo.py

```
view plain  copy to clipboard  print  ?
print "Soy un .exe"
```

y el archivo setup.py correspondiente. Los cambios que tenemos que realizar a setup.py son sencillos: importar py2exe, y utilizar los argumentos `console` y `windows` para indicar el nombre del script o scripts que queramos convertir en ejecutables de consola o ejecutables de interfaz gráfica, respectivamente.

```
view plain  copy to clipboard  print  ?
from distutils.core import setup
import py2exe

setup(name="Aplicacion de ejemplo",
      version="0.1",
      description="Ejemplo del funcionamiento de distutils",
      author="Raul Gonzalez"
```

```
        scripts=["ejemplo.py"],
        console=["ejemplo.py"]
)
```

Para crear el ejecutable, utilizamos una nueva opción de línea de comandos para setup.py disponible tras importar el módulo y llamada, cómo no, py2exe:

python setup.py py2exe

Con esto py2exe generará un directorio build, con las librerías compiladas, y un directorio dist, con los archivos que conforman nuestra aplicación.

Entre los archivos que podemos encontrar en dist tendremos uno o varios ejecutables con el mismo nombre que los scripts indicados en `console` y `windows`, un archivo python*.dll, que es el intérprete de Python, y un archivo library.zip, que contiene varios archivos pyc que son los módulos que utiliza la aplicación compilados.

Si queremos reducir el número de archivos a distribuir, podemos utilizar la opción `--bundle` de py2exe para añadir a library.zip las dll y los pyd (`--bundle 2`) o las dll, los pyd y el intérprete (`--bundle 1`).

python setup.py py2exe --bundle 1

o bien podemos añadir un nuevo argumento `options` a la función `setup` que indique el valor a utilizar (opción `bundle_files`), de forma que no tengamos que añadir el flag `--bundle` cada vez que usemos el comando `py2exe`:

view plain copy to clipboard print ?

```
from distutils.core import setup
import py2exe

setup(name="Aplicacion de ejemplo",
      version="0.1",
      description="Ejemplo del funcionamiento de distutils",
      author="Raul Gonzalez",
      author_email="zootropo en gmail",
      url="http://mundogeek.net/tutorial-python/",
      license="GPL",
      scripts=["ejemplo.py"],
      console=["ejemplo.py"],
      options={"py2exe": {"bundle_files": 1}}
)
```

Por último podemos incluso prescindir de library.zip e incrustarlo en el ejecutable utilizando el argumento `zipfile=None`

view plain copy to clipboard print ?

```
from distutils.core import setup
import py2exe

setup(name="Aplicacion de ejemplo",
      version="0.1",
      description="Ejemplo del funcionamiento de distutils",
      author="Raul Gonzalez",
      author_email="zootropo en gmail",
      url="http://mundogeek.net/tutorial-python/",
      license="GPL",
```

```
        zipfile=None
)
```


distutils, exe, py2exe, python, setuptools, tutorial

Comentarios

1. fidojones

   Yo recomendaria también pyinstaller que me ha dado mejores resultados con PyQT que el py2exe. Eso si recomiendo usar la version del SVN que es la que está mas actualizada.

   http://pyinstaller.python-hosting.com/

   Y para empaquetar para Mac OS, py2app. Aunque con las QT4 da problemas con las rutas.

   http://svn.pythonmac.org/py2app/py2app/trunk/doc/index.html

   El que no quiera volverse loco para empaquetar PyQT para Mac OS que mire este post, a mi me ha funcionado y he podido finalmente empaquetarlo:

   http://markmail.org/message/6evkmysjce6lgjry

   Responder

2. Matias

   muy buena info, estaba por buscar esto!!! 😀
   siempre siempre obtengo buena info de tu blog, estoy muy agredesido!!!

   Responder

3. deckerix

   La verdad es que es un crack! ;_)

4. MetalAgent

   Muy buen artículo …. como siempre….

   Responder

5. danyx

   excelente, hasta ahora se encontraba poca documentación sobre este tema.

   Responder

6. trankon

   Genial Post!!

   escribe todo lo que quieras sobre python, aqui siempre tienes un interesado y sobre todo con tan buena info

   Responder

7. Zootropo

   Gracias a todos por los comentarios 🙂

   Sobre todo a fidojones por los enlaces 😉

   Responder

8. The machine

   Oye Excelente! buscaba esto hace algún tiempo. Gracias una vez más.!

   Responder

9. CAVG

   Aqui hay un sitio donde hay una herramienta para convertir .py a exe.

   http://sites.google.com/site/cavgutiles/gui-py2exe

   Responder

10. Zootropo

¿Lo has programado tú, CAVG?

Responder

11. CAVG

    Si, por el mismo motivo, me gustaria que lo probaran, para ir mejorandolo.
    Saludos

    Responder

12. lijank

    Buenos Días,

    Interesante post pero me surge una duda: como se anexan las bases de datos?. He intentado agregar una línea con data_files e incluir el nombre de la base de datos, pero esta no se adjunta.

    He revisado la documentación y no logro encontrar el problema.
    http://docs.python.org/distutils/setupscript.html#installing-additional-files

    Tienes algún ejemplo?

    Responder

13. jorquera

    py2exe me ha funcionado sin problemas con un hola mundo, pero con un sencillo programa usando glade no me funciona. Se genera el ejecutable pero al intentar ejecutarlo se crea un archivo de texto con los errores:
    Traceback (most recent call last):
    File “ejemplo.py”, line 5, in
    File “gtk__init__.pyc”, line 48, in
    File “gtk_gtk.pyc”, line 12, in
    File “gtk_gtk.pyc”, line 10, in __load
    ImportError: No module named cairo

    El programa ejemplo.py es el siguiente:

    # -*- coding: utf-8 -*-

    import pygtk
    pygtk.require(“2.0”)

class App:

def __init__(self):
self.glade = gtk.glade.XML(“proyecto1.glade”)
self.glade.signal_autoconnect(self)
self.glade.get_widget(“window1”).show_all()

def on_window1_delete_event(self, widget, event):
gtk.main_quit()

def on_button1_clicked(self, widget):
gtk.main_quit()

if __name__ == “__main__”:
try:
a = App()
gtk.main()
except KeyboardInterrupt:
pass

¿Tengo que añadir gtk.glade en setup.py?

Gracias.

Responder

- Rolo

  Hola !!!

  Tengo tu mismo problema. Encontraste la solucion??

  Responder

14. matyvico

Hola, zootropo quiero decirte que baje tu libro, y lo imprimi, es muy bueno y he aprendido mucho, ahora mi problema llega cuando quiero generar un .exe para ejecutar en windows. He leido el libro, este artículo, y muchos mas en internet. Creo que mi pregunta es tan tonta que es hasta insultante 😛

Bien la cuestion es que yo hice un pequeño programa en python en ubuntu, pero ahora cuando quiero usar py2exe no se como, por lo que creo este programa (py2exe) se ejecuta

Responder

- DeathscytheX

  wenas ^^, si estas en Ubuntu puedes buscar en repositorios la librería que importas (python-distutils-extra, esto en fedora 10, prueba con un ‘sudo apt-get search distutils’) e instalarla, con eso es cosa que hagas los archivos setup.py y el script que quieres importar (el hola mundo para que ensayes) y con las intrucciones de Zootropo no deberías tener problemas (‘python setup.py py2exe’ por ejemplo)

Responder

- DeathscytheX

  Leyendo me e dado cuenta que está bajo Windows xD… lamento la intromisión y encima errada :P, bueno, byebye y hasta la otra ^^…

Responder

15. Universo Digital » Hacer RPM de aplicaciones Python en Fedora

    […] el resto del procedimiento pueden visitar este enlace. Categories: 64 bits, Fedora, Linux, Python Tags: Comments (0) Trackbacks (0) Leave a […]

    Responder

16. Hacer RPM de aplicaciones Python en Fedora « Linux y Mas Blog

    […] el resto del procedimiento pueden visitar este enlace. Este artículo ha sido leido 4 veces AKPC_IDS += "2063,";Popularity: unranked [?] Share and […]

    Responder

17. ariel

    Buen tutorial, pero tengo una duda:

    He logrado empaquetar mi script con py2exe, y al llevarlo a otra maquina que no tiene instalado python, NO funciona.

    ¿La otra maquina también tiene que tener instalado python o yo estoy haciendo algo mal?

    ¿Me está faltando algún paso adicional?

- Zootropo

  No, la otra máquina no necesita tener el intérprete de Python instalado. Esa es la gracia, que está integrado dentro del exe.

  Responder

  - ariel

    Gracias por la respuesta Zootropo, pero lo que me faltaba en la otra maquina era tener instalado el Paquete redistribuible de Microsoft Visual C++ 2008 (x86).

    pd:/Excelente blog!

    Responder

18. gisel lopez

    hooola,
    soy totalmente nueva en esto
    pero quiero hacer un programa simple como regalo para un amigo en donde ingrese palabras y le de un estilo de mensaje y me parecio que python es una buena idea solo que no se como comenzar
    en que partes del blog puedo ver para iniciarme o podrias ayudarme? gracias.

    Responder

19. suso

    tengo serios problemas con py2exe al crear el .exe de un script que usa librerías especializadas como reportlab, pil, numpy.

    He buscado y no existe documentación en la web del uso de py2exe con este tipo de librerías.

    Espero que alguien me pueda ayudar

    Responder

20. Carles

    Hola

empaquetar en un exe però en el momento de ejecutar-lo no puedo, empieza la ejecución, supongo que aparece algun error y se cierra inmediatamente.

Alguien me podria dar alguna idea de que ocurre, decir que la aplicación consta de multiples archivos para la geation de modelos, vistas y control.

Gracias

Responder

21. Ugescio

Hola mi dudda es ya que termone mi aplicacion en python como puedo hacer que se instale en windows mobile no se si alguien ya lo ha hecho no encuentro como hacerlo ya probe varias de las herramientas que mencionan pero siempre me aparece el mismo error de que la aplicacion no es valida para pocket PC espero me puedan ayudar estoy bastante desesperado…
Garacias.

Responder

22. Antonio Samper

Hola, soy nuevo en esto de python quisiera saber como se configura aptana para trabajar con python o Bos-constructor ya que escribo un “hola mundo”, lo ejecuto pero no hace nada.

gracias.

Responder

23. alejairo

muy buen blog

Responder

24. Miguel Angel

Muy buena toda la info. Me ha servido de gran ayuda, gracias.

Responder

25. lucas

datos de una DB creada con sqlite.
El tema es el siguiente, cuando “corro” el script con el intérprete de python funciona todo bien, ahora cuando creo el ejecutable me muestra la interface principal del programa y cuando clickeo mostrar DB, me da error si almacene alguna cadena con acento o eñe.
Resumiendo, mi duda es, como hago para crear el ejecutable y que me acepte las cadenas con acento y/o eñe.
Cualquier cosa mi duda esta posteada en el siguiente enlace:
http://www.lawebdelprogramador.com/foros/Python/1283920-%5Bayuda%5Dacentos_y_enes_en_py2exe.html
Saludos!.

Responder

26. pikassa arte

Quiero que mi trabajo se convierta en una aplicación para millones de usuarios.

Hola,

Permítame que me presente. Soy Cintia.com, y editamos
tendencias desde hace 30 años.
Somos buenos muy buenos y especialistas. Solicito Una Prueba.
Ruego por favor me indique el inicio para ser su programa.
Tenemos 161.
Muchas gracias,
Contesteme. Un cordial saludo,

Responder

27. josue

Disculpa una duda fijate que con el py2exe como puedo hacer un exe de varios py

Responder

28. Lizette

Buenisimo Tutorial, muchas gracias, me ayudo bastante, estuve observando su pagina y tiene articulos muy interesantes. Gracias

Responder

29. leonardojofre

te falto integrar innosetup para hacer el instalador de los archivos creados por py2exe

Responder

30. sabroso

    Y como lo instalo en windows

Responder

31. Ragnarok

    Hola, este post esta genial, pero tengo una duda, yo he desarrollado varias aplicaciones y utilizo linux como sistema operativo, hasta ahora todas las app que he echo han sido para este sistema, pero ahora mi nuevo cliente (un profesor bastante exigente) quiere que una aplicacion que hice la haga para window ya que esta esta orientada para personas que la mayoria usa window, he intentado buscar py2exe para linux pero parece que solo esta para window, asi que aqui va mi pregunta, como puedo hacer el setpy.py para hacer un exe en linux o donde puedo encontrar py2exe para linux mint? saludos

Responder

32. Ragnarok

    Si trabajo con pyqt5 como puedo añadir estas librerias usando setuptools o distools

Responder

33. Sergio Rustichelli

    Excelente el Post

Responder

Deja un comentario

Nombre

email (no se mostrará)

Tu web (opcional)

Enviar comentario

Recibir un email con los siguientes comentarios a esta entrada.

Recibir un email con cada nueva entrada.

Tweets by @mundo_geek

Mundo geek
@mundo_geek

Jueves 16. Madrid. Regalos por asistir. Sorteo MacBook Pro. Vino. Conferencias. Gratis hasta completar aforo tuexitoonline.es

09 Jun

Mundo geek
@mundo_geek

Pesadilla en la oficina: Te replantearás dejarlo todo y

Este sitio usa cookies para mejorar su experiencia. Si continúa en el mismo, consideramos que acepta su uso

Aceptar