Python - Módulos

Módulos

Si un programa se vuelve muy largo, sería interesante separarlo en distintos archivos para un mantenimiento más sencillo. De ese modo, una función podría utilizarse en distintos programas sin copiar su definición en cada uno de ellos.

Para conseguir todo esto, Python tiene un modo de poder crear archivos con la definición de ciertas funciones y así poder utilizarlas en un cualquier programa Python sólo con importar dichos archivos. Estos archivos se denominan **módulos**.

Enlace : Introducción interesante sobre módulos.

Sumario

Tipos de módulos

Los módulos de Python pueden pertenecer a una de las siguientes categorías: módulos Python, módulos compilados, módulos integrados o extensiones C o C++ compiladas o vinculadas dinámicamente cuando se importan.

Desde el punto de vista del usuario, no hay ninguna diferencia entre estos cuatro tipos de módulos, puesto que todos ellos son importados y utilizados del mismo modo.

- **Módulos Python -** Son imples archivos que contienen instrucciones Python. Al importarlos, serán compilados y, por lo tanto, su bytecode es producido y guardado.
- **Módulos compilados -** Son archivos que contienen las instrucciones codificada en bytecode.
- **Módulos integrados -** Se obtienen de fuentes C, compiladas y vinculadas estáticamente al intérprete. El módulo *sys* de la librería estándar, es un módulo integrado.
- **Las extensiones -** Las extensiones son módulos obtenidos de fuentes C o C++ que son compilados como librerías dinámicas. tienen un sufijo .so en Linux y .dll o .pyd en Windows. El módulo math es un ejemplo de este tipo de módulos.

...

Para manejar los módulos de Python lo más cómodo es instalar **pip**. Se trata de un administrador de paquetes utilizado para instalar y configuarar paquetes de Python.

- Muchos de estos paquetes podemos encontrarlos en el Python Package Index (PyPI).
- Python 2.7.9 y posterior incluye **pip** por defecto.
- Para instalar **pip** en el sistema:

```
# Debian
$ apt-get install python-pip
# Debian y para python3
$ apt-get install python3-pip
# MAC OS X
$ sudo easy_install pip
```

- Para instalar o desinstalar un paquete cualquiera con **pip**:

```
# Instalar un paquete
$ pip install nombre_paquete
# Desinstalar un paquete
$ pip uninstall nombre_paquete
```

Las definiciones de un módulo pueden ser importadas a otros módulos o a uno principal.

Así, *un módulo* es un archivo que contiene definiciones y declaraciones de Python. El nombre del archivo es el nombre del módulo con el sufijo .py agregado.

- Dentro de un módulo, el nombre del mismo debe estar disponible en el valor de la variable global **__name__**.
- Como ejemplo, vamos a crear un módulo con dos funciones: una que escriba por pantalla los N primeros números de la serie Fibonacci y otra que devuelva una lista con los N primeros números de la serie Fibonacci.

Crea un archivo fibo.py con el siguiente contenido:

```
# -*- coding: utf-8 -*-
# módulo de números Fibonacci

def fib1(n):    # Escribe los N primeros números de la serie Fibonacci
    x = 0
    a, b = 0, 1
    while x < n:
        print '- %s' % b
        a, b = b, a+b
        x = x + 1
    print ' '

def fib2(n): # Devuelve una lista con los N primeros números de la serie Fibonacci
    resultado = []
    x = 0
    a, b = 0, 1
    while x < n:
        resultado.append(b)
        a, b = b, a+b
        x = x + 1
    return resultado
```

Luego crea el siguiente archivo **prueba.py** que hará uso del módulo **fibo.py** creado:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Importamos el módulo fibo.py que está en el mismo directorio que este archivo
import fibo
# La variable __name__ contiene el nombre del módulo
name = fibo.__name__
print 'Utilizando el módulo: %s' % name

# Pedimos al usuario que introduzca un número
n = float(raw_input('Número de elementos de la serie Fibonacci a generar: '))
# Utilizamos la función fib2() existente en el módulo fibo.py
# Esta función devuelve una lista de números
serie = fibo.fib2(n)
print 'La serie Fibonacci hasta %s es %s' %(n, serie)
```

Ejemplo Módulo Wand

Wand es un módulo que nos permite manipular imágenes. Instalación del módulo:

```
$ apt-get install libmagickwand-dev
$ pip install Wand
```

Calcular el tamaño de una imagen y cambiar el tamaño de la imagen:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from wand.image import Image
in_name = 'Mona_Lisa.jpg'
out_name = 'Mona_Lisa_small.png'

with Image(filename = in_name) as img:
    width = img.width
    height = img.height
print width
```

```
print height

print "Cambiamos tamaño"

with Image(filename = in_name) as image:
    w, h = image.size
    new_width = 800
    factor = new_width/float(w)
    new_height = int(h * factor)
    image.resize(new_width, new_height, 'lanczos', 1.0)
    image.save(filename = out_name)
    width = image.width
    height = image.height
print width
print height
```

Ejemplo Módulo psutil

Un módulo muy interesante para la administración de sistemas es psutil.
Su instalación requiere que, previamente, tengamos los paquetes de desarrollo de Python:

```
$ cd /tmp
$ wget https://bootstrap.pypa.io/get-pip.py
$ python get-pip.py
$ aptitude install python-dev           # Python headers
$ apt-get install gcc
$ reboot
$ pip install psutil
```

Ejemplo Librería matplotlib

Librería Python de dibujo en 2D.

--Volver