Curso de Desarrollo en Lenguaje Python para Inteligencia Artificial (Málaga)

M.374.001.001

7 de abril 2021 09:30-13:30

Modulo 3 – Tema 1

Spiros Michalakopoulos

Gobierno de España · Ministerio de Industria, Comercio y Turismo · Escuela de organización industrial

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

polo de contenidos digitales

Ayuntamiento de Málaga

Módulo 3: Bases de Datos e Interfaces con Python

Desarrollo en Lenguaje Python para Inteligencia Artificial

Año de realización: 2021

PROFESOR
Spiros Michalakopoulos

spiros.eoi@gmail.com
https://www.linkedin.com/in/spiros-michalakopoulos

www.eoi.es

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Tema 1

Archivos y ficheros. Módulos, paquetes y distribución de software.

Módulos

Cuando queremos utilizar una función en varios scripts diferentes, no es práctico ni pythonico el tener que estar definiéndola en cada script.

- Los módulos permiten reducir el código y evitar duplicidades.
- Son ficheros que contienen definiciones y estructuras que pueden ser utilizadas en otros scripts.
- Pueden ser importados desde cualquier script para utilizar las funciones que contienen
- Facilita establecer una jerarquía coherente en los proyectos

Paquetes

- Permiten agrupar varios módulos con una jerarquía específica
- Facilitan el uso de los módulos al poderse distribuir y manejar como si fuesen librerías
- Deben tener un fichero __init__.py para ser reconocido como paquete por el intérprete
- Para que el paquete sea distribuible (utilizado desde cualquier directorio) debe ir acompañado de un fichero setup.py

Distribución. Archivo setup

Para hacer que un paquete sea distribuible y se pueda instalar en Python para poder ser utilizado por cualquier usuario, es necesario crear un archivo setup con información sobre el paquete y los subpaquetes que contiene.

Documentación setup script

Documentación setuptools

Distribución. sdist

Una vez tenemos el archivo setup, en el mismo directorio que la carpeta del paquete, abrimos terminal en ese directorio y creamos el distribuible con el comando:

```
python setup.py sdist
```

Documentación sdist

Instalar un paquete

Con el terminal abierto en el mismo directorio donde esté el archivo comprimido del paquete, ejecutamos el comando:

```
pip install nombre_archivo.zip
```

o, en el caso de unix o mac:

```
pip install nombre_archivo.tar.gz
```

Desinstalar un paquete

Desde el terminal, ahora desde el directorio que queráis, se puede desinstalar el paquete con el comando

```
pip uninstall nombre_paquete
```

Setup avanzado

Es bueno respetar un estándar en la función setup:

```
from setuptools import setup

setup(name="Paquete",  # Nombre
      version="0.1",  # Versión de desarrollo
      description="Paquete de prueba",  # Descripción del
  funcionamiento
      author="Dumbledore",  # Nombre del autor
      author_email='dumbledore@hogwarts.magic',  # Email del autor
      license="GPL",  # Licencia: MIT, GPL, GPL 2.0...
      url="http://hogwarts.magic",  # Página oficial (si la hay)
      packages=['Paquete','Paquete.subpaquete']
)
```

find_packages

Cuando tenemos muchos subpaquetes, lo mejor es importar el módulo find_packages:

```
from setuptools import setup, find_packages

setup(name="Paquete",  # Nombre
      version="0.1",  # Versión de desarrollo
      description="Paquete de prueba",  # Descripción del funcionamiento
      author="Dumbledore",  # Nombre del autor
      author_email='dumbledore@hogwarts.magic',  # Email del autor
      license="GPL",  # Licencia: MIT, GPL, GPL 2.0...
      url="http://hogwarts.magic",  # Página oficial (si la hay)
      packages=find_packages()
)
```

Dependencias

A menudo, nuestro paquete contiene código que depende de otros paquetes externos. Esto se conoce como dependencias del paquete y hace falta utilizar install_requires para que el paquete instale cualquier otro paquete externo que pueda hacer falta.

```
setup(name="Paquete",  # Nombre
      version="0.1",  # Versión de desarrollo
      description="Paquete de prueba",  # Descripción del funcionamiento
      author="Dumbledore",  # Nombre del autor
      author_email='dumbledore@hogwarts.magic',  # Email del autor
      license="GPL",  # Licencia: MIT, GPL, GPL 2.0...
      url="http://hogwarts.magic",  # Página oficial (si la hay)
      packages=find_packages(),
      install_requires=["modulo_externo","otro_modulo_externo"]
     )
```

Dependencias

Si tenemos muchas dependencias, interesará utilizar lo que se conoce como “fichero de dependencias”, que debe estar en el mismo directorio que el archivo setup.py

Creamos un archivo requirements.txt con las dependencias, una por línea:

```
pillow==1.1.0
django>=1.10.0,<=1.10.3
pygame
```

Dependencias

Y el archivo setup.py debería ser:

```
setup(name="Paquete",  # Nombre
      version="0.1",  # Versión de desarrollo
      description="Paquete de prueba",  # Descripción del funcionamiento
      author="Dumbledore",  # Nombre del autor
      author_email='dumbledore@hogwarts.magic',  # Email del autor
      license="GPL",  # Licencia: MIT, GPL, GPL 2.0...
      url="http://hogwarts.magic",  # Página oficial (si la hay)
      packages=find_packages(),
      install_requires=[i.strip() for i in
        open("requirements.txt").readlines()]
      )
```

test_suite

Si queremos incluir en el paquete los tests que hayamos realizado, debemos tenerlos en una carpeta en el mismo directorio que setup.py y añadir el atributo test_suite en el setup con el nombre de dicha carpeta:

```
setup(name="Paquete",  # Nombre
      version="0.1",  # Versión de desarrollo
      description="Paquete de prueba",  # Descripción del funcionamiento
      author="Dumbledore",  # Nombre del autor
      author_email='dumbledore@hogwarts.magic',  # Email del autor
      license="GPL",  # Licencia: MIT, GPL, GPL 2.0...
      url="http://hogwarts.magic",  # Página oficial (si la hay)
      packages=find_packages(),
      install_requires=["modulo_externo","otro_modulo_externo"],
      test_suite="tests"
     )
```

PyPI

PyPI (Python Package Index) es un repositorio público con todos los paquetes creados por la comunidad de Python. Podemos subir nuestros paquetes también, siempre que tengan una calidad aceptable y, sobre todo, que estén bien documentados.

Para que el paquete esté bien categorizado en PyPI, se debe utilizar un parámetro llamado classifiers, que indica los valores de los clasificadores que hay disponibles en la plataforma.

Lista de clasificadores PyPI

Y el archivo setup.py debería contener:

```
setup(...,
      classifiers=[
          "Development Status :: 3 - Alpha",
          "Topic :: Utilities",
          "License :: OSI Approved :: GNU General Public License"
        ],
        )
```

Una vez que tenemos el paquete y el setup listo, para compartirlo en PyPI, lo primero es registrar una cuenta personal.

Después, desde el directorio donde esté el paquete, ejecutamos:

```
python setup.py register
```

Este comando genera una solicitud de registro del paquete en el repositorio. Irán apareciendo instrucciones y tendremos que indicar usuario y contraseña (por eso es importante haberse dado de alta como usuario antes).

Una vez registrado, hay que publicar una versión del paquete, que será el descargable como tal. para ello, debemos ejecutar:

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

Probar el paquete en modo desarrollador

Para poder probar el paquete e ir haciendo cambios sin tener que instalar y desinstalar el paquete con cada cambio, existe la instalación en modo desarrollo. Para ello, se ejecuta el comando:

```
python setup.py develop
```

De hecho, no se hace solo para probar los paquetes propios, sino también para probar los que hay en proceso, en github, por ejemplo, antes de hacer aportaciones.

Para desinstalar el paquete en modo desarrollo se ejecuta el comando:

```
python setup.py develop --uninstall
```

Pyinstaller. Generar un ejecutable

A menudo, se crean programas para que los utilicen terceros, y es necesario que lo hagan desde un fichero ejecutable.

Para ello está la librería PyInstaller:
https://www.pyinstaller.org/

Documentación de PyInstaller

Paquetes soportados por PyInstaller