Lección 14: Modulos y Paquetes

Índice

Anterior

Módulos

Definición

Un módulo es un archivo *.py* que contiene un conjunto de funciones y/o clases, que puede importarse desde otro módulo o que puede ejecutarse.

Creación de un módulo

Para crear un módulo se meten nuevas funciones/clases en un archivo *.py*, por ejemplo en un archivo *utils.py*:

```
# file utils.py

import string
import secrets


def password_gen(length, alphabet = string.punctuation + string.ascii_lowercase + string.asc
    '''Genera una contraseña segura de longitud length
        Parámetros por clave opcionales:
        alphabet: Cadena con caracteres a utilizar en la contraseña. Por defecto, signos de
    password = ''
    for i in range(length):
        password += secrets.choice(alphabet)
    return password
```

Importando un módulo

Desde otro fichero *.py* en el mismo directorio se puede importar el módulo utilizando *import* y llamar a las funciones clases incluídas.

```
# file main.py
import utils

print(utils.password_gen(length = 10))
```

Cuando se importa un módulo se le puede poner un álias que facilite su utilización en el resto del código.

```
# file main.py
import utils as ut

print(ut.password_gen(length = 10))
```

Tambien pueden importarse únicamente algunas funciones/clases de un módulo.

```
# file main.py
from utils import password_gen

print(password_gen(length = 20))
```

Atributos de módulo

El atributo ***name*** da acceso al nombre del módulo, pero si el módulo es ejecutado independientemente el atributo tendrá el valor *'**main**'* en vez de el nombre del módulo. Esto permite definir cómo debe ejecutarse el módulo cuando se ejecuta de forma independientes.

```
#file utils.py
# -*- coding: utf-8 -*-

"""
Some utils functions
"""

import string
import secrets


def password_gen(length,
                 alphabet = string.punctuation
                 + string.ascii_lowercase
                 + string.ascii_uppercase
                 + string.digits):
    '''Genera una contraseña segura de longitud length
        Parámetros por clave opcionales:
        alphabet: Cadena con caracteres a utilizar en la contraseña.
        Por defecto, signos de puntuación, minúsculas, mayúsculas y números
    '''

    password = ''
    for i in range(length):
        password += secrets.choice(alphabet)
    return password

# Lo que hay a partir del siguiente if únicamente se ejutará si el módulo es lanzado directa
if __name__ == '__main__':
    print(password_gen(length = 10))
```

El atributo ***file*** contiene el path al archivo del módulo.

```
import utils
```

```
print(utils.__name__)
print(utils.__file__)
```

Importando algunos módulos Built-in

Módulo OS

Con el módulo *OS* se pueden realizar muchas taréas típicas de sistema operativo.

```
# import the module
import os
# Creating a directory
os.mkdir('directory_name')
# Changing the current directory
os.chdir('path')
# Getting current working directory
os.getcwd()
# Removing directory
os.rmdir()
```

Módulo SYS

El módulo *sys* proporciona funciones y variables del entorno de ejecución de python. La función *sys.argv* devuelve una lista de argumentos de línea de comandos pasados a un script de Python. El elemento en el índice 0 de esta lista es siempre el nombre del script, en el índice 1 está el argumento pasado desde la línea de comandos.

```
# file utils.py
# -*- coding: utf-8 -*-

"""
Some utils functions
"""

import string
import secrets
import sys


def password_gen(length,
                 alphabet = string.punctuation
                 + string.ascii_lowercase
                 + string.ascii_uppercase
                 + string.digits):
    '''Genera una contraseña segura de longitud length
        Parámetros por clave opcionales:
```

```
        alphabet: Cadena con caracteres a utilizar en la contraseña.
        Por defecto, signos de puntuación, minúsculas, mayúsculas y números
    '''

    password = ''
    for i in range(length):
        password += secrets.choice(alphabet)
    return password


if __name__ == '__main__':
    print(password_gen(length = int(sys.argv[1])))
```

Desde la línea de comando se puede comprobar cómo funciona:

```
$ python utils.py 10
waK9xj5qUK
```

Módulo Math

El módulo *math* contiene constantes y funciones con operaciones matemáticas.

```
import math
print(math.pi)           # 3.141592653589793, pi constant
print(math.sqrt(2))      # 1.4142135623730951, square root
print(math.pow(2, 3))    # 8.0, exponential function
print(math.floor(9.81))  # 9, rounding to the lowest
print(math.ceil(9.81))   # 10, rounding to the highest
print(math.log10(100))   # 2, logarithm with 10 as base
```

Paquetes

Los módulos pueden (y suelen) organizarse en un tipo de carpetas especiales que se llaman paquetes. Dentro de estas carpetas deben existir necesariamente un archivo llamado ___init___.py, aunque esté vacío. No es obligatorio que todos los módulos pertenezcan a un paquete.

Si el paquete o algún módulo del paquete es importado, el código en ___init___.py se ejecutará. Se utiliza cuando hay que incializar algo para utilizar el paquete.

Los módulos dentro de un paquete se pueden importar utilizando notación con punto. Ej: *import paquete.modulo*, *from paquete import modulo*, *from paquete.modulo import clase/funcion*, *from paquete import **.

Siguiente

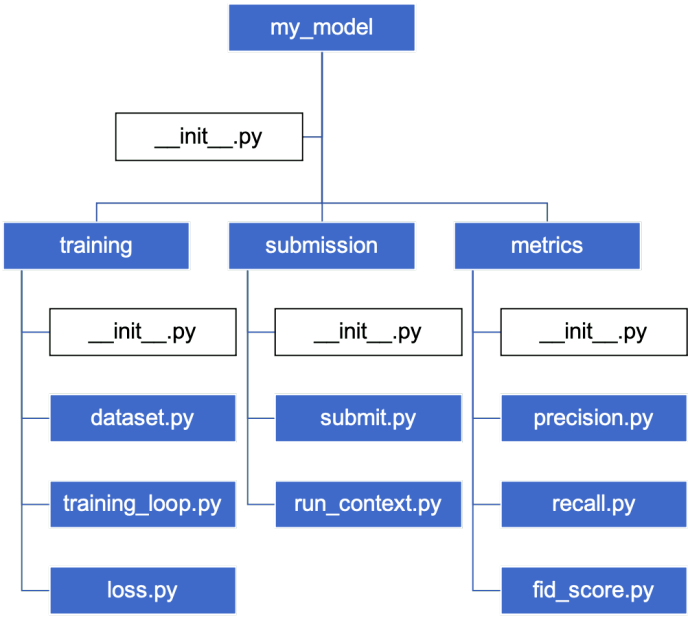

Figure 1: Estructura paquete