Módulos y paquetes en Python

Introducción a Python

José Miguel Gimeno

Esta foto de Autor desconocido está bajo licencia CC BY-SA

Módulos y paquetes en Python

- En una sesión interactiva podemos crear funciones, variables, etc., pero cuando se cierre perdemos todo.
- Si usamos un editor para crear un archivo .py, lo que conseguimos es un script de forma que podemos ejecutarlo sin perderlo.

Módulos y paquetes en Python

- Cuando el programa adquiere un cierto tamaño, es conveniente partirlo en varios archivos (módulos) que trabajen conjuntamente.
- Esto facilita el mantenimiento y, además, permite reutilizar mejor el código.

Módulos y paquetes en Python

script: archivo .py preparado para ser ejecutado

módulo: archivo .py preparado para ser llamado o importado

Módulos y paquetes en Python

Un módulo es un archivo .py cuyos objetos pueden ser accedidos desde otro archivo.

Localización de módulos

Cuando se importa un módulo, el intérprete de Python lo busca en este orden:

1. En el directorio actual (donde se está ejecutando el archivo)
2. En cada directorio de la variable de shell PYTHONPATH
3. En la ruta predeterminada (en Linux normalmente es /usr/local/lib/Python)

Localización de módulos

La ruta se almacena en el módulo sys como la variable sys.path (contiene directorio actual, PYTHONPATH y ruta predeterminada).

Se puede modificar:

»import sys

»sys.path.append(‘nueva_ruta’)

Archivos compilados de módulos

Para acelerar la carda de los módulos, Python cachea las versiones compiladas de cada módulo en el directorio __pycache__ con el nombre module.version.pyc

Atributo __name__

Si el módulo, además de definiciones y declaraciones, contiene instrucciones para ser ejecutado directamente, entonces es necesario distinguir si el módulo es llamado o ejecutado

Importación de módulos

De acuerdo con PEP8, debe realizarse al comienzo del archivo, en orden alfabético

Primero los módulos propios de Python

Luego los módulos de terceros

Por último, los módulos propios

Entre cada bloque de imports dejar una línea en blanco.

Importación de módulos

La sintaxis general es

import nombre_modulo_sin_extension

Importación de módulos

Podemos importar solo los elementos que vamos a utilizar:

from modulo import funcion

Importación de módulos

Podemos asignar un alias:

from modulo import función as alias

Importación de módulos

Podemos importar todos los nombres excepto los que empiecen por __:

```
from modulo import *
```

Estructura de un módulo

1. Espacio de trabajo
2. Codificación utilizada
3. Docstring
4. Variables de documentación (con doble guion bajo __)
5. Variables y funciones (con sus docstring)
6. Control sobre si se llama como módulo o se ejecuta como programa principal

Funciones help() y dir()

Permiten acceder a la documentación y a los nombres definidos en el módulo

Ejercicio módulos

Crea dos módulos y un programa principal que los llamará:

- Módulos
  - usuario.py
  - contrasena.py
- Programa principal: validador.py

Ejercicio módulos – usuario.py

Este módulo valida de nombres de usuarios de acuerdo con estos criterios:

1. mínimo de 6 caracteres y un máximo de 12.
2. debe ser alfanumérico.
3. si tiene menos de 6 caracteres, retorna el mensaje "El nombre de usuario debe contener al menos 6 caracteres".
4. si tiene más de 12 caracteres, retorna el mensaje "El nombre de usuario no puede contener más de 12 caracteres".
5. si tiene caracteres distintos a los alfanuméricos, retorna el mensaje "El nombre de usuario puede contener solo letras y números".
6. si es válido, retorna True.

Ejercicio módulos – contrasena.py

Este módulo valida de contraseñas de acuerdo con estos criterios:

1. debe contener un mínimo de 8 caracteres.
2. debe contener letras minúsculas, mayúsculas, números y al menos 1 carácter no alfanumérico.
3. no puede contener espacios en blanco.
4. si contraseña válida, retorna "True".
5. si no válida, retorna el mensaje "La contraseña elegida no es segura"

Paquetes

El concepto de paquete hace referencia a una manera de ordenar el código.

En Python es una carpeta que contiene un archivo llamado __init__.py para indicar que en esa carpeta hay un grupo de módulos preparados para ser importados.

Paquetes

El archivo __init__.py está vacío, sólo sirve para indicar que la carpeta es un paquete.

Podemos crear subpaquetes dentro.

Opcionalmente, dentro del paquete puede haber un programa main.py para ejecutarse.

Paquetes

¿Cómo creamos un paquete?

Paquetes

¿Cómo accedemos a los módulos de un paquete?

Hay tres posibilidades…

Paquetes

1) El script está en el mismo directorio donde está el paquete: lo llamamos como hemos visto hasta ahora (from…import y demás variantes

Paquetes

2) El script está en el otro directorio: entonces debemos indicar la **ruta relativa** al paquete

Paquetes

3) Por último, podemos instalar el paquete en Python, para ello hay que convertir el paquete en distribuible.

Paquetes – distribución

Creamos un archivo setup.py

https://entrenamiento-python-basico.readthedocs.io/es/3.7/leccion8/distribucion.html#

Algunos módulos de Python

- random
- datetime y time
- os

Módulo random

- randint()
- randrange()
- random()
- uniform()
- choice()
- shuffle()

Módulo datetime y time

- datetime()
- now()
- strftime()
- timedelta()

Módulo os

- getcwd()
- listdir()
- access()
- chdir()
- chmod()
- chown()
- mkdir()
- remove()
- rmdir()
- rename()