caminodelaserpiente
蛇道

librerias paquetes y modulos

Librerias, Paquetes y módulos

ÍNDICE

pág.

Propósito

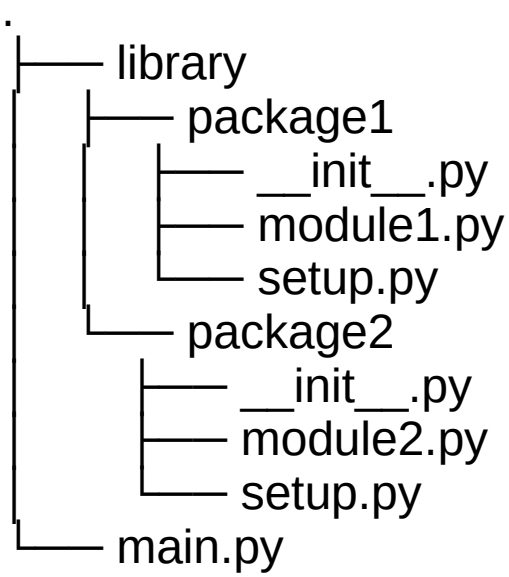
Esta guía te ayudará a llevar tu código al siguiente nivel mediante la creación de paquetes y módulos. El objetivo es maximizar la eficiencia y la reutilización del código.

Al dominar la creación de paquetes y módulos, podrás aprovechar al máximo tu código existente y desarrollar soluciones más robustas y flexibles. Podrás reutilizar fácilmente funciones y componentes en diferentes proyectos, lo que te permitirá ahorrar tiempo y esfuerzo en el desarrollo.

El enfoque práctico de esta guía te brindará los conocimientos y habilidades necesarios para avanzar en tu trayectoria como desarrollador y crear soluciones de software sólidas.

Definición de conceptos

Estructura inicial de directorios



Descripción de directorios y archivos

library: Una librería es un conjunto de paquetes y módulos que se utilizan para realizar tareas específicas. Las librerías son conjuntos de código predefinido que facilitan el desarrollo de software al proporcionar funciones y características listas para usar.

packageN.py: Un paquete es una colección de módulos relacionados. Puede contener varios archivos de código fuente (módulos). Los paquetes se utilizan para organizar y estructurar el código en jerarquías.

`main.py:` Es el archivo principal del programa. El punto de inicio y se encarga de coordinar la ejecución del programa, utilizando los módulos necesarios para realizar diferentes tareas.

`setup.py:` Se utiliza para definir metadatos y configuraciones relacionadas con el modulo.

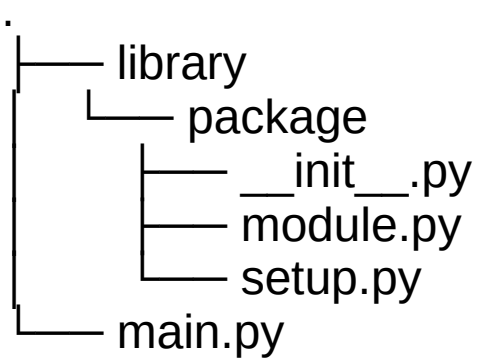
Definición de conceptos

moduleN: Un módulo es un archivo individual que contiene código fuente de programación. Puede contener variables, funciones, clases y otros elementos. Los módulos se utilizan para organizar y reutilizar el código. Puedes pensar en un módulo como un archivo de código fuente que puede ser importado y utilizado en otros programas.

__init__.py: Este archivo es necesario para que Python reconozca la carpeta como un paquete o módulo importable. No es necesario agregar contenido alguno.

Creacion de librería
Ejercicio 1

Estructura de directorios



Contenido de archivos

module.py:

```
gnu@linux:~/library/package$ cat module.py
class MiClase:
    def __init__(self, parametro1, parametro2):
        self.parametro1 = parametro1
        self.parametro2 = parametro2

    def metodo1(self):
        # Código del método 1
        print("Ejecutando método 1")

    def metodo2(self):
        # Código del método 2
        print("Ejecutando método 2")
gnu@linux:~/library/package$
```

Creacion de librería
Ejercicio 1

__init__.py:

```
gnu@linux:~/library/package$ cat __init__.py
gnu@linux:~/library/package$
```

setup.py:

```
gnu@linux:~/library/package$ cat setup.py
from setuptools import setup

setup(
    name='name_module',
    version='1.0.0',
    description='description tool',
    author='name_author',
    author_email='author@e-mail.com',
    url='https://github.com/...',
    packages=['name_module']
    install_requires=[ ]
    classifiers=[
        'Development Status :: 3 - Alpha',
            'Topic :: Software Development :: Libraries :: Python Modules',
        'License :: OSI Approved :: GNU General Public License v3 (GPLv3)',
        'Programming Language :: Python :: 3',
        'Operating System :: OS Independent',
        'Topic :: Scientific/Engineering :: Information Analysis'
    ]
gnu@linux:~/library/package$
```

Creacion de librería
Ejercicio 1

main.py:

```
gnu@linux:~$  cat main.py
from library.package.module import MiClase

def module():
    # Crear una instancia de la clase
    objeto = MiClase("valor1", "valor2")

    # Acceder a los atributos y métodos de la instancia
    print(objeto.parametro1)
    print(objeto.parametro2)
    objeto.metodo1()
    objeto.metodo2()

if __name__ == '__main__':
    module()
gnu@linux:~$
```

Ejecucion script

main.py:

```
gnu@linux:~$  python3 main.py
valor1
valor2
Ejecutando método 1
Ejecutando método 2
gnu@linux:~$
```

Creacion de librería
Ejercicio 1

Descripción del contenido en archivos

module.py:
Clase 'MiClase': Define una clase llamada MiClase.

Método __init__(self, parametro1, parametro2): Es el constructor de la clase que se ejecuta cuando se crea una instancia de MiClase. Recibe dos parámetros, parametro1 y parametro2, y asigna esos valores a los atributos parametro1 y parametro2 respectivamente.

Método metodo1(self): Define un método llamado metodo1 que no recibe parámetros adicionales. Este método contiene código que se ejecuta cuando se llama al método metodo1(). En este caso, simplemente imprime "Ejecutando método 1" en la consola.

Método metodo2(self): Define un método llamado metodo2 que no recibe parámetros adicionales. Este método contiene código que se ejecuta cuando se llama al método metodo2(). En este caso, simplemente imprime "Ejecutando método 2" en la consola.

Creacion de librería
Ejercicio 1

setup.py:
from setuptools import setup: Esta línea importa la función setup del paquete setuptools. setuptools es una biblioteca de Python que facilita la distribución y la instalación de paquetes.

name='name_module': Especifica el nombre del módulo o paquete que estás creando o distribuyendo.

version='1.0.0': Indica la versión del módulo. Es una convención utilizar un esquema de versión semántica (por ejemplo, X.Y.Z) para denotar cambios significativos en la funcionalidad.

description='description tool': Proporciona una breve para informar a los usuarios sobre el propósito y la funcionalidad del módulo.

author='name_author': Especifica el nombre del autor del módulo.

author_email='author@e-mail.com': Proporciona el correo electrónico del autor del módulo.

url='https://github.com/...': Define la URL del proyecto o del repositorio donde se encuentra el código fuente del módulo.

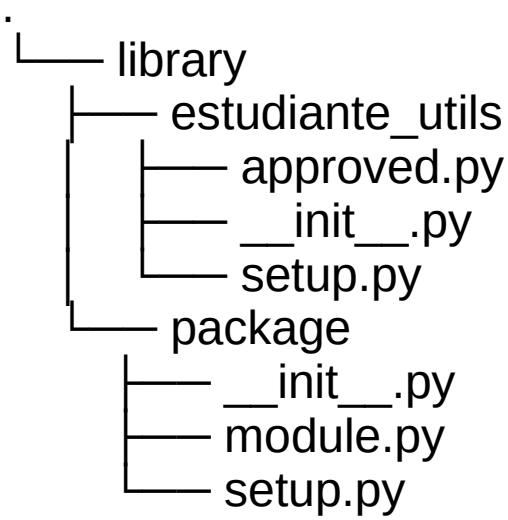
packages=['name_module']: Enumera los paquetes que deben ser incluidos en la distribución. Aquí, se especifica que el paquete name_module debe ser incluido.

install_requires=['selenium', 'pandas']: Define las dependencias requeridas para el módulo.

classifiers=[...]: Esta sección define una lista de clasificadores que describen las características del módulo. Estos clasificadores ayudan a categorizar el módulo en función de su estado de desarrollo, licencia, compatibilidad con versiones de Python, sistema operativo, etc. Los clasificadores se utilizan principalmente para ayudar a los usuarios a encontrar el módulo en los índices de paquetes y a seleccionar los paquetes adecuados para sus necesidades.

Creacion de librería
Ejercicio 2

Estructura de directorios



Contenido de archivos

approved.py:

```
gnu@linux:~/library/estudiante_utils$ cat approved.py
class Estudiante:
    def __init__(self, nombre, edad, promedio):
        self.nombre = nombre
        self.edad = edad
        self.promedio = promedio

    def imprimir_informacion(self):
        print("Nombre:", self.nombre)
        print("Edad:", self.edad)
        print("Promedio:", self.promedio)

    def es_aprobado(self):
        if self.promedio >= 7.0:
            return True
        else:
            return False
gnu@linux:~/library/estudiante_utils$
```

Creacion de librería
Ejercicio 2

__init__.py:

```
gnu@linux:~/library/estudiante_utils$  cat __init__.py
gnu@linux:~/library/estudiante_utils$
```

setup.py:

```
gnu@linux:~/library/estudiante_utils$ cat setup.py
from setuptools import setup

setup(
    name='approved',
    version='1.0.0',
    description='Verificar aprobacion de asignatura de alumno.',
    author=' 蛇道 ',
    author_email='caminodelaserpiente.py@gmail.com',
    url='https://github.com/guia_librerias_paquetes_modulos',
    packages=['approved']
    install_requires=[]
    classifiers=[
        'Topic :: Software Development :: Libraries :: Python Modules',
        'License :: OSI Approved :: GNU General Public License v3 (GPLv3)',
        'Programming Language :: Python :: 3',
        'Operating System :: OS Independent',
    ]
)
gnu@linux:~/library/estudiante_utils$
```

Creacion de librería
Ejercicio 2

main.py:

```
gnu@linux:~$  cat main.py
from library.package.module import MiClase
from library.estudiante_utils.approved import Estudiante


def module():
    # Crear una instancia de la clase
    objeto = MiClase("valor1", "valor2")

    # Acceder a los atributos y métodos de la instancia
    print(objeto.parametro1)
    print(objeto.parametro2)
    objeto.metodo1()
    objeto.metodo2()


def estudiante_utils():
    # Crear instancias de la clase Estudiante
    estudiante1 = Estudiante("Maria", 18, 8.5)
    estudiante2 = Estudiante("Gloria", 20, 6.2)

    # Imprimir información de los estudiantes
    estudiante1.imprimir_informacion()
    print("Aprobado:", estudiante1.es_aprobado(), end="\n\n")

    estudiante2.imprimir_informacion()
    print("Aprobado:", estudiante2.es_aprobado())


if __name__ == '__main__':
    #module()
    estudiante_utils()
gnu@linux:~$
```

Creacion de librería
Ejercicio 2

Ejecucion script

main.py:

```
gnu@linux:~$  python3 main.py
Nombre: Maria
Edad: 18
Promedio: 8.5
Aprobado: True

Nombre: Gloria
Edad: 20
Promedio: 6.2
Aprobado: False
gnu@linux:~$
```

https://github.com/caminodelaserpiente/guia_librerias_paquetes_modulos