100 Ejercicios

Tema 5: Módulos y Paquetes

Temas Cubiertos:

1. Importación de Módulos
2. Creación de Módulos
3. Estructura de Paquetes
4. Uso de pip y Gestión de Dependencias

1ra Parte: 50 Ejercicios para Practicar y Afianzar Conocimientos del Tema 5

Importación de Módulos

1. **Importar Módulo Estándar:** Escribe un programa que importe el módulo math y calcule la raíz cuadrada de un número.
2. **Uso de Funciones de un Módulo:** Importa el módulo random y genera un número aleatorio entre 1 y 100.
3. **Importar Específicamente:** Importa solo la función sqrt del módulo math.
4. **Importar con Alias:** Importa el módulo datetime con el alias dt.
5. **Uso de Módulos Internos:** Importa el módulo os y muestra el directorio actual.
6. **Listar Módulos Disponibles:** Escribe un programa que imprima todos los módulos disponibles en el entorno.
7. **Importar Módulos desde un Archivo:** Importa un módulo desde un archivo .py ubicado en el mismo directorio.
8. **Uso de importlib:** Utiliza importlib para importar un módulo dinámicamente.
9. **Importar Módulos desde un Paquete:** Importa un módulo específico desde un paquete.
10. **Explorar Módulos con dir():** Utiliza dir() para explorar los atributos de un módulo importado.

Creación de Módulos

11. **Crear un Módulo Simple**: Crea un archivo mimodulo.py con una función que devuelva un saludo.
12. **Importar y Utilizar tu Módulo**: Importa mimodulo.py y llama a la función definida.
13. **Función con Argumentos**: Modifica mimodulo.py para que contenga una función que acepte argumentos.
14. **Módulo con Variables**: Crea un módulo con variables y accede a ellas desde otro archivo.
15. **Funciones de Módulo**: Escribe varias funciones en un módulo y llámalas desde otro archivo.
16. **Módulo con Clases**: Crea un módulo con una clase simple y utiliza esa clase en otro archivo.
17. **Documentar Módulos**: Documenta tu módulo utilizando docstrings.
18. **Módulo con Funciones de Clase**: Crea una clase con métodos y atributos en un módulo.
19. **Importar Módulo de Otro Proyecto**: Importa un módulo de otro proyecto utilizando sys.path.
20. **Módulo con Constantes**: Define constantes en un módulo y accede a ellas desde otro archivo.

Estructura de Paquetes

21. **Crear un Paquete**: Crea una estructura de directorio para un paquete con submódulos.
22. **Estructura de Directorios**: Crea un paquete con varios módulos y un archivo __init__.py.
23. **Importar Módulos del Paquete**: Importa módulos de un paquete desde un script principal.
24. **Uso de Submódulos**: Crea submódulos en un paquete y accede a ellos desde otro módulo.
25. **Organizar Módulos en Carpetas**: Organiza módulos en diferentes carpetas dentro de un paquete.

26. **Importar Submódulos Específicos**: Importa submódulos específicos de un paquete.
27. **Uso de __init__.py**: Configura __init__.py para inicializar tu paquete.
28. **Uso de __all__ en Paquetes**: Define __all__ en __init__.py para controlar qué se exporta.
29. **Crear un Paquete con Módulos Relacionados**: Organiza módulos relacionados en un paquete.
30. **Acceder a Módulos del Paquete**: Usa import relativo para acceder a módulos dentro de un paquete.

Uso de pip y Gestión de Dependencias

31. **Instalar un Paquete con pip**: Instala el paquete requests utilizando pip.
32. **Verificar Paquetes Instalados**: Escribe un script que liste todos los paquetes instalados con pip.
33. **Actualizar un Paquete con pip**: Actualiza un paquete específico con pip.
34. **Desinstalar un Paquete con pip**: Desinstala un paquete utilizando pip.
35. **Instalar Paquetes desde un Archivo**: Instala paquetes desde un archivo requirements.txt.
36. **Uso de pip freeze**: Guarda los paquetes instalados en un archivo requirements.txt con pip freeze.
37. **Crear un Entorno Virtual**: Crea y activa un entorno virtual utilizando venv.
38. **Instalar Paquetes en un Entorno Virtual**: Instala paquetes en un entorno virtual y verifica su instalación.
39. **Desactivar un Entorno Virtual**: Desactiva un entorno virtual y verifica que no esté activo.
40. **Instalar una Versión Específica de un Paquete**: Instala una versión específica de un paquete con pip install.

Ejercicios de Repaso

41. **Crear un Paquete de Módulos**: Crea un paquete con varios módulos y escribe un script para probarlos.
42. **Actualizar y Gestionar Paquetes**: Escribe un script que actualice todos los paquetes listados en requirements.txt.
43. **Crear un Paquete con Dependencias**: Define las dependencias de tu paquete en setup.py.
44. **Construir un Paquete con setuptools**: Utiliza setuptools para construir un paquete Python.
45. **Publicar un Paquete en PyPI**: Publica tu paquete en PyPI utilizando twine.
46. **Instalar Paquetes desde GitHub**: Instala un paquete desde un repositorio GitHub utilizando pip.
47. **Verificar Dependencias de Paquetes**: Escribe un script que verifique las dependencias de un paquete.
48. **Instalar Paquetes de Python 2 y 3**: Instala un paquete compatible con Python 2 y 3 utilizando pip.
49. **Automatizar la Instalación de Paquetes**: Escribe un script que automatice la instalación de paquetes desde requirements.txt.
50. **Configurar pip con Opciones Avanzadas**: Utiliza opciones avanzadas de pip para instalar paquetes, como --user y --upgrade.

2da Parte: 50 Ejercicios Avanzados sobre el Tema 5

Importación de Módulos

1. **Cargar Módulos Dinámicamente**: Utiliza importlib para cargar módulos de forma dinámica desde una ruta específica.
2. **Modularización de Código Extenso**: Divide un programa extenso en módulos y crea una estructura de proyecto modular.
3. **Manipular sys.modules**: Interactúa con sys.modules para importar y modificar módulos en tiempo de ejecución.

4. **Escribir Módulos en Tiempo Real:** Escribe y ejecuta módulos en tiempo real utilizando exec().
5. **Usar Módulos Ocultos:** Importa y utiliza módulos ocultos (no documentados) de Python.
6. **Crear un Módulo de Interfaz de Usuario:** Escribe un módulo que interactúe con la interfaz de usuario utilizando tkinter.
7. **Cargar Módulos desde URLs:** Utiliza importlib.util para cargar módulos desde URLs.
8. **Inyectar Dependencias en Módulos:** Utiliza técnicas de inyección de dependencias en tus módulos.
9. **Crear un Módulo con Códigos de Error:** Implementa un módulo con manejo avanzado de errores y códigos de estado.
10. **Modificar Módulos en Tiempo Real:** Utiliza técnicas avanzadas para modificar módulos en tiempo real durante la ejecución.

Creación de Módulos

11. **Implementar un Módulo con Decoradores:** Crea un módulo que utilice decoradores personalizados.
12. **Crear Módulos con Generadores:** Implementa funciones en un módulo utilizando generadores.
13. **Crear Módulos con Context Managers:** Implementa context managers avanzados en un módulo.
14. **Módulos con Decoradores de Clase:** Escribe módulos que utilicen decoradores de clase.
15. **Incorporar Métodos Mágicos en Módulos:** Implementa métodos mágicos en un módulo y explícalos.
16. **Crear Módulos con Métodos de Clase:** Escribe módulos que utilicen métodos de clase y estáticos.
17. **Escribir un Módulo con Meta-Clases:** Implementa un módulo utilizando meta-clases.
18. **Desarrollar un Módulo con Tipos Genéricos:** Utiliza typing para definir tipos genéricos en un módulo.

19. **Módulo con Funcionalidades Avanzadas:** Crea un módulo que utilice técnicas avanzadas como memoización y caching.
20. **Módulos con Decoradores Anidados:** Implementa decoradores anidados en un módulo y muestra ejemplos prácticos.

Estructura de Paquetes

21. **Crear Paquetes con Subpaquetes:** Crea una estructura de paquetes con varios niveles de subpaquetes.
22. **Modularización Avanzada:** Divide una aplicación grande en múltiples paquetes y subpaquetes, asegurando una estructura modular y escalable.
23. **Documentar Paquetes:** Utiliza herramientas de documentación como Sphinx para documentar un paquete completo.
24. **Testeo de Paquetes:** Escribe un conjunto de tests unitarios para un paquete utilizando unittest o pytest.
25. **Automatización de Construcción:** Utiliza herramientas como setuptools y twine para automatizar la construcción y distribución de un paquete.
26. **Paquetes con Dependencias Externas:** Define y maneja dependencias externas en un paquete utilizando requirements.txt o setup.py.
27. **Paquetes con Datos Estáticos:** Incluye datos estáticos (archivos de texto, imágenes) en un paquete y accede a ellos desde el código.
28. **Compatibilidad con Múltiples Versiones de Python:** Asegura que un paquete sea compatible con múltiples versiones de Python.
29. **Uso de __main__.py en Paquetes:** Crea un archivo __main__.py para permitir la ejecución de un paquete como un script.
30. **Distribución de Paquetes Privados:** Configura un índice privado de paquetes (como devpi) para distribuir paquetes privados.

Uso de pip y Gestión de Dependencias

31. **Creación de Entornos Virtuales Personalizados:** Configura entornos virtuales personalizados para diferentes proyectos utilizando virtualenv.
32. **Configuración de pip Avanzada:** Utiliza opciones avanzadas de configuración de pip como pip.conf y variables de entorno.
33. **Uso de pipenv para Gestión de Dependencias:** Gestiona dependencias de un proyecto utilizando pipenv.
34. **Instalación de Paquetes desde Repositorios:** Instala paquetes directamente desde repositorios de código como GitHub.
35. **Automatización de Tareas con invoke:** Utiliza invoke para automatizar tareas comunes de desarrollo en un proyecto.
36. **Gestión de Dependencias con poetry:** Utiliza poetry para gestionar dependencias y la publicación de paquetes.
37. **Integración Continua:** Configura un pipeline de integración continua para un proyecto Python utilizando herramientas como GitHub Actions o Travis CI.
38. **Uso de tox para Testing en Múltiples Entornos:** Configura tox para ejecutar tests en múltiples entornos virtuales y versiones de Python.
39. **Instalación de Paquetes con Ruedas:** Compila y distribuye ruedas (wheels) para una instalación de paquetes más rápida.
40. **Creación de Archivos de Distribución Source:** Crea archivos de distribución fuente (sdist) para un paquete Python y publícalos en PyPI.

Ejercicios de Repaso

41. **Crear un Paquete Completo:** Desarrolla un paquete Python completo que incluya módulos, submódulos, tests, y documentación.

42. **Automatización de Publicación de Paquetes:** Escribe un script que automatice la publicación de un paquete en PyPI.

43. **Gestión de Dependencias Avanzada:** Configura y gestiona dependencias complejas para un proyecto utilizando pip, pipenv, o poetry.

44. **Configuración de CI/CD para un Paquete:** Configura un pipeline de CI/CD completo para un paquete Python, incluyendo tests, linting y despliegue.

45. **Creación de Extensiones en Cython:** Desarrolla una extensión de Cython para un paquete Python y compílala.

46. **Integración de Herramientas de Calidad de Código:** Configura herramientas de calidad de código como flake8, black, y mypy en un proyecto.

47. **Desarrollo de Plugins:** Desarrolla un sistema de plugins para un paquete Python utilizando el patrón de diseño de plugins.

48. **Manejo de Versionado Semántico:** Implementa versionado semántico en un proyecto y automatiza la actualización de versiones.

49. **Despliegue de Paquetes en Entornos Cloud:** Configura el despliegue automático de un paquete Python en entornos de nube como AWS Lambda o Google Cloud Functions.

50. **Desarrollo de Paquetes para Aplicaciones Web:** Desarrolla un paquete Python específico para una aplicación web utilizando frameworks como Flask o Django.