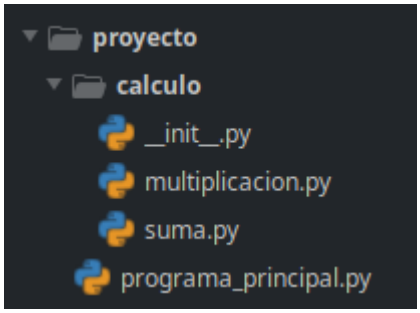
12. Paquetes y sys.path (ejercicios ampliación)

1. ¿Qué es un paquete?

- Un paquete es una carpeta que contiene módulos relacionados entre sí.
- Permite organizar y reutilizar código.
- La carpeta contiene un archivo **__init__**.py (puede estar vacío) que indica a Python que la carpeta es un paquete.

2. Módulos y paquetes

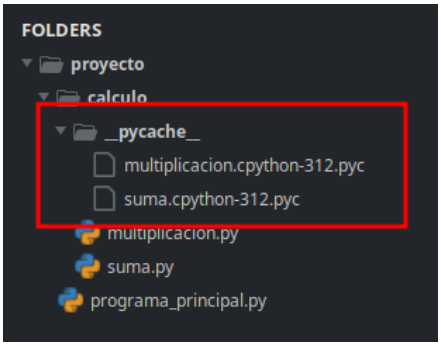
- ***Módulo***: archivo .py con funciones, variables o clases.
- ***Paquete***: carpeta con módulos y un archivo **__init__**.py.
- Ejemplo de estructura:



-
- Para importar los módulos contenidos en el paquete:

```
from calculo import suma, multiplicacion
```

Al ejecutar por primera vez el módulo principal observa que se crea una nueva carpeta llamada __pycache__ en la que el intérprete traduce y guarda el código fuente (.py) a bytecode (una versión intermedia optimizada que Python puede ejecutar más rápido).



3. Qué se puede poner en __init__.py

El archivo `__init__.py` se ejecuta automáticamente al importar el paquete, y permite controlar cómo se comporta la importación. Podemos tener:

1. ***Vacío***: marca la carpeta como paquete.
2. ***Reexportar*** las funciones de los módulos del paquete para poder hacer imports más simples desde el código principal.

Por ejemplo si dentro del módulo multiplicacion.py hay una función llamada multiplica, se podría escribir en __init__.py

```
from .multiplicacion import multiplica
```

ahora podríamos utilizar:

```
from calculo import multiplica
```

en lugar de

```
from calculo.multiplicacion import multiplica
```

3. Incluir ***Código de inicialización*** que se ejecuta al importar. Código que se ejecutará cuando realizemos la importación

5. ¿Dónde busca Python los módulos?

- Cuando hacemos un import, Python busca los módulos y paquetes en una lista de directorios definida en una variable llamada sys.path.
- Para ver su contenido y con ello los directorios que encontrará Python podemos ejecutar desde consola:

```
python3 -c "import sys; print(sys.path)"
```

Se mostrará el listado de directorios dentro de los cuales Python buscará los paquetes con los módulos que queramos utilizar.

- En la salida de sys.path, el primer elemento suele ser '', que hace referencia al directorio del programa principal (y los subdirectorios incluidos en él).
- Si queremos utilizar un paquete o módulo que está guardado en un directorio que inicialmente no está en ese directorio, Python no lo encontrará y generará un error.

6. ¿Cómo añadir una ruta temporal a un paquete?

- Podemos agregar una carpeta propia, por ejemplo donde tengamos nuestros módulos personalizados:

```
import sys
sys.path.append('/home/usuario/mis_paquetes')
```

- Al cerrar el programa sys.path volverá a su valor original.

7. ¿Cómo añadir una ruta permanente a un paquete?

- Desde consola se modifica la **variable de entorno PYTHONPATH**:, por ejemplo para añadir la carpeta mis_paquetes incluida en mi directorio personal:

```
export PYTHONPATH="/home/usuario/mis_paquetes:$PYTHONPATH"
```

8. Ejercicio

Paso 1. Crear un módulo sin paquete

1. Crea una carpeta llamada `proyecto1` y dentro un archivo `principal.py`
2. En la misma carpeta crea un archivo llamado `operaciones.py` con el siguiente contenido:

   a) Una función llamada cuadadro que reciba un parámetro y devuelva su cuadrado.

b) Una función llamada `cubo` que reciba un número y devuelva su cubo.

3. En el archivo **principal.py** importa las funciones anteriores y muestra por pantalla los resultados de calcular el cuadrado y el cubo de dos números.
4. Ejecuta el programa y comprueba que funciona correctamente.

Paso 2. Crear un módulo sin paquete

1. Crea una nueva carpeta llamada mis_funciones fuera de proyecto1.
2. Mueve dentro de ella el archivo operaciones.py.
3. Ejecuta de nuevo el archivo principal.py.
4. Observa el error que aparece.
5. Explica por qué sucede (pista: Python no encuentra el módulo en los directorios de búsqueda).

Paso 3. Añadir una ruta temporal

1. Edita el archivo principal.py para añadir las siguientes líneas al inicio:

```
import sys
sys.path.append('/ruta/donde/guardaste/mis_funciones')
```

2. (Sustituye la ruta por la que corresponda en tu equipo).
3. Ejecuta de nuevo el programa y comprueba que ahora sí funciona.
4. Muestra por pantalla el contenido de sys.path para ver la nueva ruta añadida.

Paso 4. Convertir la carpeta en paquete

1. Dentro de la carpeta mis_funciones, crea un archivo vacío llamado __init__.py.
2. Crea otro archivo nuevo llamado extra.py con una función doble(x) que devuelva el doble del número recibido.
3. En __init__.py, añade las siguientes líneas:

```
from .operaciones import cuadrado, cubo
from .extra import doble
```

4. Crea un nuevo proyecto (por ejemplo **proyecto2**) con un archivo **principal.py**.
5. Intenta importar el paquete completo con:

```
from mis_funciones import cuadrado, doble
```

6. Comprueba si funciona (recuerda que puede ser necesario volver a añadir la ruta con sys.path.append()).

Paso 5. Añadir la ruta de forma permanente

1. Abre una terminal y escribe:

```
export PYTHONPATH="/ruta/donde/guardaste:$PYTHONPATH"
```

2. Abre el intérprete de Python y muestra el contenido de sys.path para comprobar que aparece tu carpeta.
3. Ejecuta el programa del paso anterior sin añadir sys.path.append() en el código.
4. Comprueba que ahora también funciona.

Paso 6. Ejecutar código al importar

1. Abre el archivo init.py y añade al final la siguiente línea:

```
print("Paquete mis_funciones cargado correctamente")
```

2. Ejecuta el programa y observa lo que ocurre al importar el paquete.