

# Módulos y paquetes



# Módulos

A medida que los programas crecen, es necesario separarlo en varios archivos para que el mantenimiento sea más sencillo. Quizás también sea necesario usar una función ya escrita en otro programa sin la necesidad de copiar su definición en cada programa que la quiera utilizar. Para ello, los módulos son la solución.



# Generalidades

- Un módulo en Python es un archivo que contiene definiciones de funciones, clases y variables, junto con cualquier otro código ejecutable.
- Tienen la extensión .py.
- Pueden ser importados y utilizados en otros programas de Python.
- Ayudan a organizar y reutilizar el código al dividirlo en componentes más manejables y específicos.

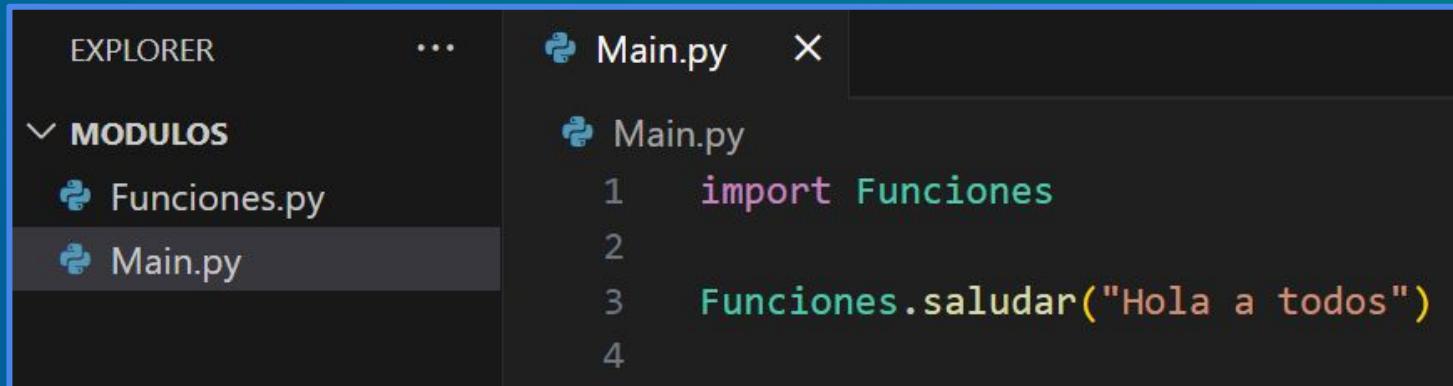
# Creación

Supongamos que definimos las siguientes funciones y variables para saludar en un archivo .py (en este caso nuestro módulo)

```
EXPLORER ...  
MODUL... D+ E+ ⌂ ⌂  
Funciones.py X  
Funciones.py > ...  
Funciones.py  
Main.py  
1 MENSAJE_INICIAL = "Hola, bienvenidos a programación I"  
2 MENSAJE_FINAL = "Gracias por asistir"  
3  
4  
5 def saludar(mensaje:str)->None:  
6     print(mensaje)  
7  
8 def saludar_alumno(nombre:str, mensaje: str)->None:  
9     print(f"Hola, {nombre}. {mensaje}")  
10  
11 def saludar_profesor(nombre:str, materia:str)->None:  
12     print("Hola, soy el profe {nombre} de la materia {materia}")  
13
```

# Importación

Para poder utilizar las funciones definidas en el módulo debemos importar dicho módulo en el archivo en donde queremos utilizar las funciones:



The screenshot shows a code editor interface with two tabs: 'Main.py' and 'Funciones.py'. The 'Main.py' tab is active, displaying the following code:

```
1 import Funciones
2
3 Funciones.saludar("Hola a todos")
```

The 'Funciones.py' tab is also visible in the background.

En este caso, para utilizar cualquiera de las funciones definidas en el módulo, tenemos que anteponer el nombre del mismo antes de llamar a la función, seguido del operador punto.

# Importación

Observemos las siguientes alternativas para importar módulos y sus componentes:

```
Main.py X
Main.py
1 from Funciones import saludar
2
3 saludar("Hola a todos")
4
```

*Caso 1: desde el módulo Funciones,  
únicamente importamos la función  
saludar.*

```
Main.py X
Main.py
1 from Funciones import saludar, saludar_alumno
2
3 saludar("Estamos probando modulos")
4
5 saludar_alumno("Pepe", "Tenes que estudiar mas!!!")
6
```

*Caso 2: desde el módulo Funciones,  
importamos las funciones saludar y  
saludar\_alumno.*

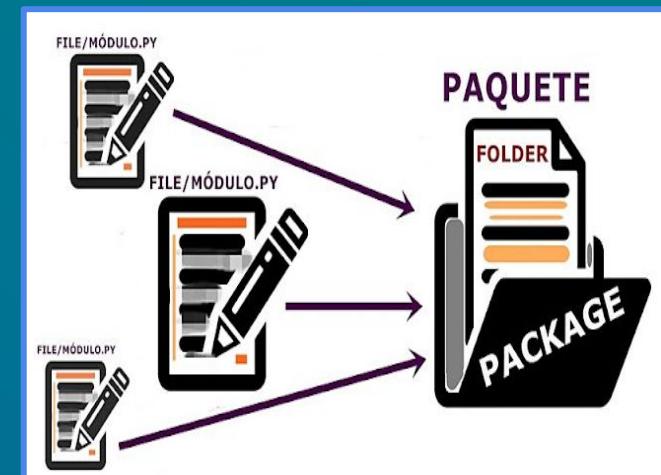
```
Main.py X
Main.py
1 from Funciones import *
2
3 saludar(MENSAJE_INICIAL)
4
5 saludar_profesor("Juan", "Química")
6
7 saludar_alumno("Pepe", "Tenes que estudiar mas!!!")
8
9 saludar(MENSAJE_FINAL)
```

*Caso 3: desde el módulo Funciones,  
importamos todas las funciones y  
variables globales con el operador \*.*

# Paquetes

Un paquete en Python es una forma de estructurar y organizar módulos relacionados.

Permiten reutilizar código de manera más efectiva, al agrupar funcionalidades relacionadas en un solo lugar. Esto facilita la gestión de proyectos grandes y complejos al proporcionar una estructura jerárquica y modular para el código.



# Creación

Para crear un paquete debemos crear una carpeta junto con un archivo `__init__.py` (de esta manera el intérprete de python entenderá que la carpeta se trata de un paquete). Luego agregamos en el mismo los módulos que necesitemos agrupar.

The screenshot shows a code editor interface with a dark theme. On the left, the Explorer sidebar displays a file tree under the 'MODULOS' section. It includes a package named 'Paquete\_curso' containing files '\_init\_.py', 'Funciones\_Paquete.py', and 'Funciones.py'. Below these are 'Main.py' and another 'Funciones.py' file. The main workspace shows a Python script with the following code:

```
1 def iniciar_clase_particular(profesor, alumno, materia):
2     print("Comienza la clase de {materia}, entre el profesor" +
3           "{profesor} y el alumno {alumno}")
```

# Importación

De la misma forma que importamos módulos independientes, podremos importar el/los módulos necesarios que se encuentran en un paquete.

```
Main.py  X
Main.py
1 #UTILIZAMOS UN ALIAS
2 import Paquete_curso.Funciones_Paquete as PC
3
4 PC.iniciar_clase_particular(profesor = "Pedro", alumno = "Juan", materia = "Matemática")
5
```

```
Main.py  X
Main.py
1 from Funciones import * # Importando todas las funciones del módulo Funciones
2
3 # Importando la función iniciar_clase_particular del módulo Funciones_Paquete
4 # del paquete Paquete_curso
5 from Paquete_curso.Funciones_Paquete import iniciar_clase_particular
6
7 iniciar_clase_particular(profesor = "Pedro", alumno = "Juan", materia = "Matemática")
```

# Comunicar módulos

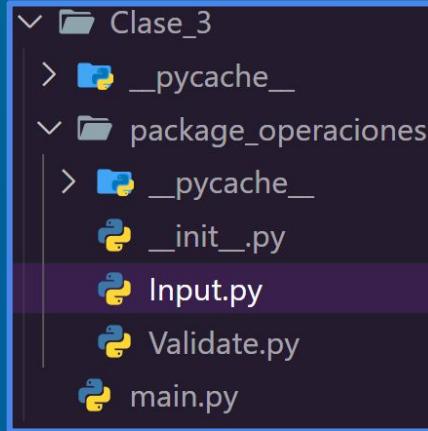
Si necesitamos comunicar 2 módulos de un mismo paquete como por ejemplo Input y Validate:

Desde Input deberemos importar Validate de la siguiente manera.

```
from .Validate import *
```

Esto se debe a que Python utiliza la convención de puntos para organizar los módulos en paquetes.

# Comunicar módulos



```
● ● ●  
1 from .Validate import * # Manera de importar entre modulos  
2  
3 def get_int():  
4     validate_int() # Función de el modulo Validate
```

**package**

**module**

**function**

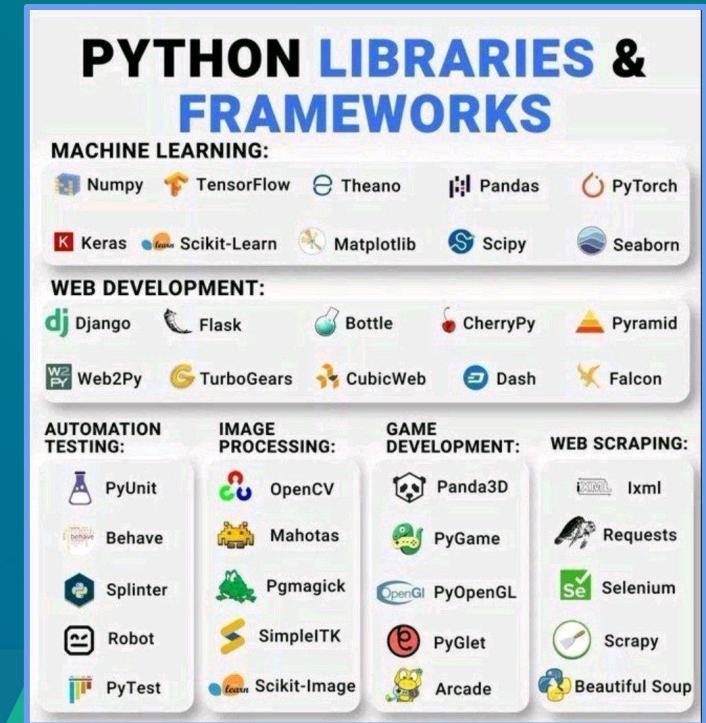
# Bibliotecas estándar

Son un conjunto de módulos y paquetes que se distribuyen junto con Python. Muchas de las operaciones más comunes ya se implementan en ellas.



# Módulos de terceros

Es inmensurable la cantidad de módulos y paquetes desarrollados por la comunidad de python, cada uno de ellos con propósitos específicos.



# Instalando módulos de terceros

Cada uno de estos paquetes está identificado por su nombre. Para instalar alguno de ellos utilizaremos una herramienta llamada pip que se incluye con la instalación de Python.

```
pip install paquete
```

En caso de que tu SO no reconozca pip:

```
python -m pip install paquete
```



```
py -m pip install paquete
```



[Si no pudiste solucionarlo, consulta esta página](#)

# pygame



Probemos instalando el módulo pygame  
Para asegurarnos que funciona correctamente,  
corramos el siguiente comando:

```
[python | py] -m pygame.examples.aliens
```

