Librerias

INSTITUTO DE INNOVACIÓN Y TECNOLOGÍA APLICADA

IITA

Contenido

PYTHON

Pregunta Inicial:

¿Qué era un modulo?

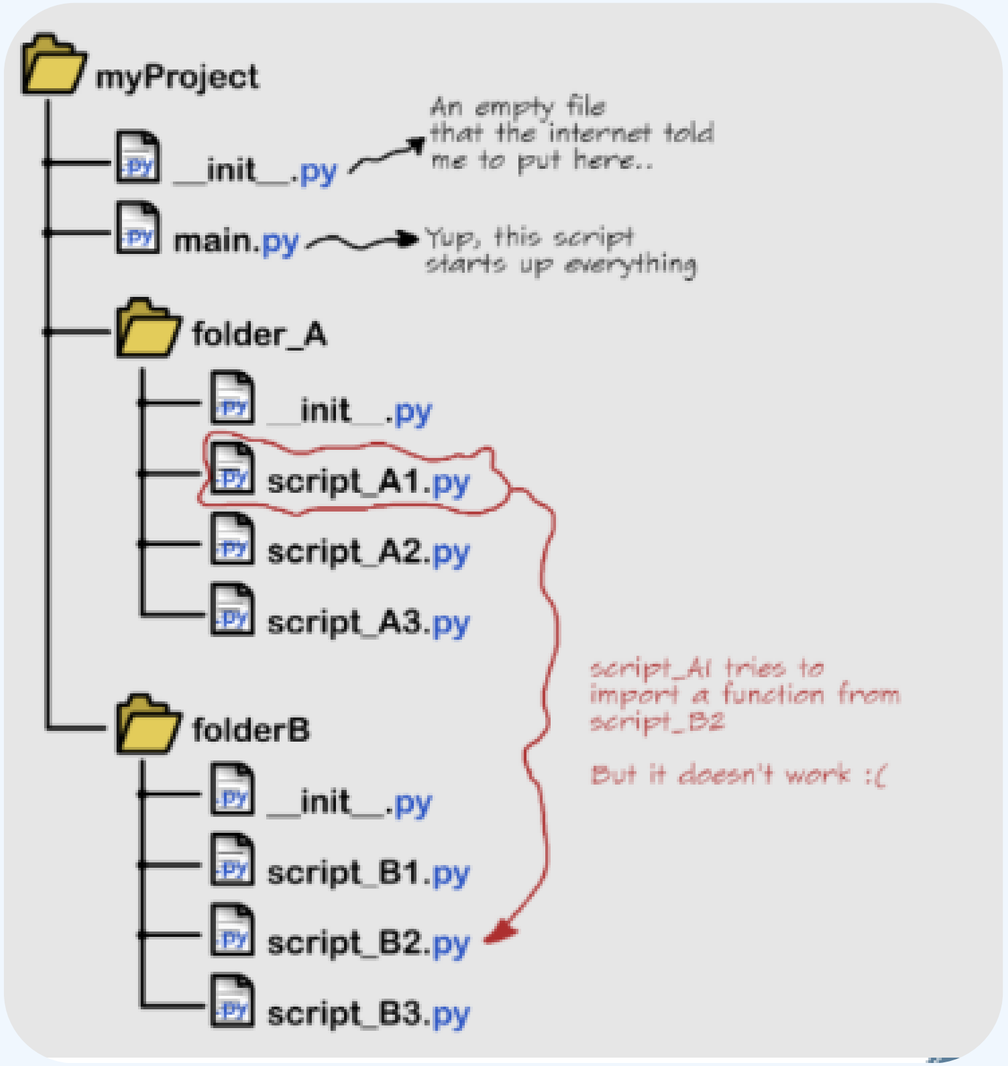
¿Qué es la programación funcional?

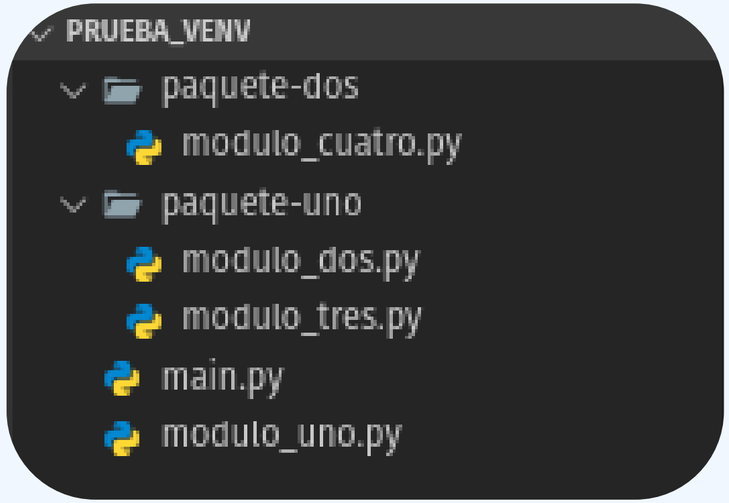
En pocas palabras, la programación funcional es un paradigma de programación distinto al **tradicional estructurado,** basado principalmente en el **uso de funciones,** siendo las mismas practicamente la única herramienta que el lenguaje nos ofrece.

MODULOS

¿Qué es un modulo?

es un archivo .py que contiene un conjunto de **funciones, variables o clases** y que puede ser usado por otros módulos.





¿Cómo se ordena?

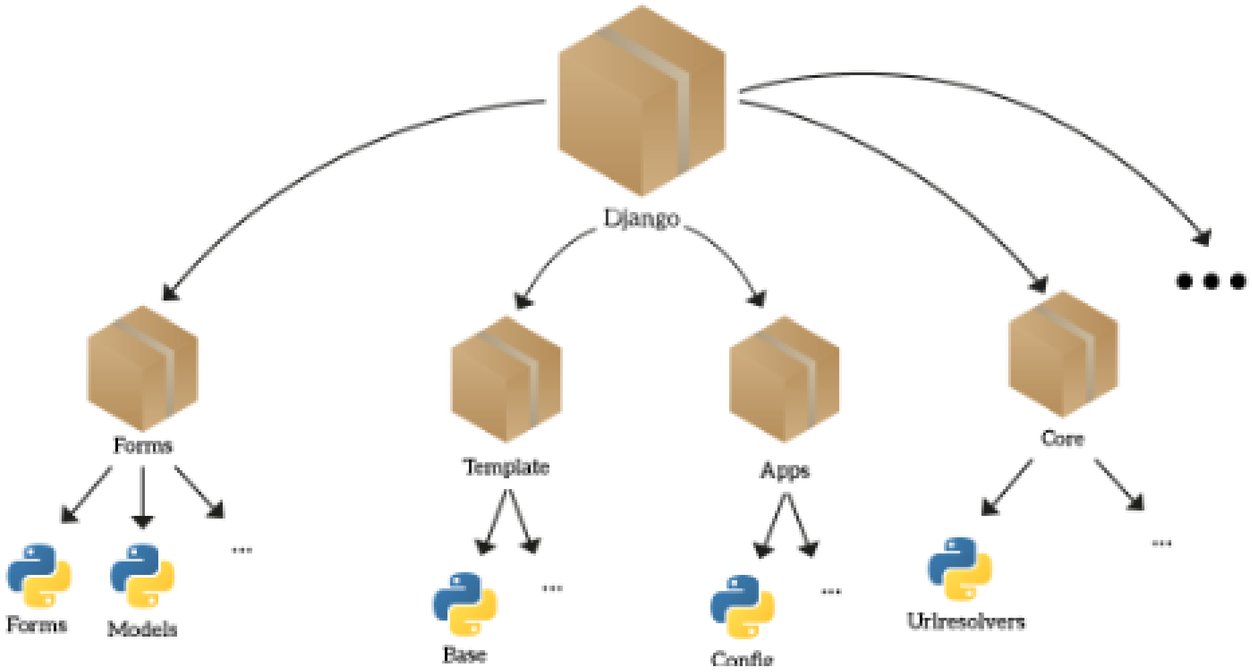
```
1   # Parte declarativa
2   import modulo_uno
3   from paquete_uno.modulo_dos import *
4   from paquete_dos.modulo_cuatro import saludar_formalmente
5
6   # Cuerpo principal de programa
7   modulo_uno.saludar()
8   saludar_formalmente()
9
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN PUERTOS TERMINAL MONITOR SERIE

```
sandokan@pop-os:/mnt/sandokan_home/workshop/prueba_venv$ /bin/py
ueba_venv/main.py
hola mundo
```

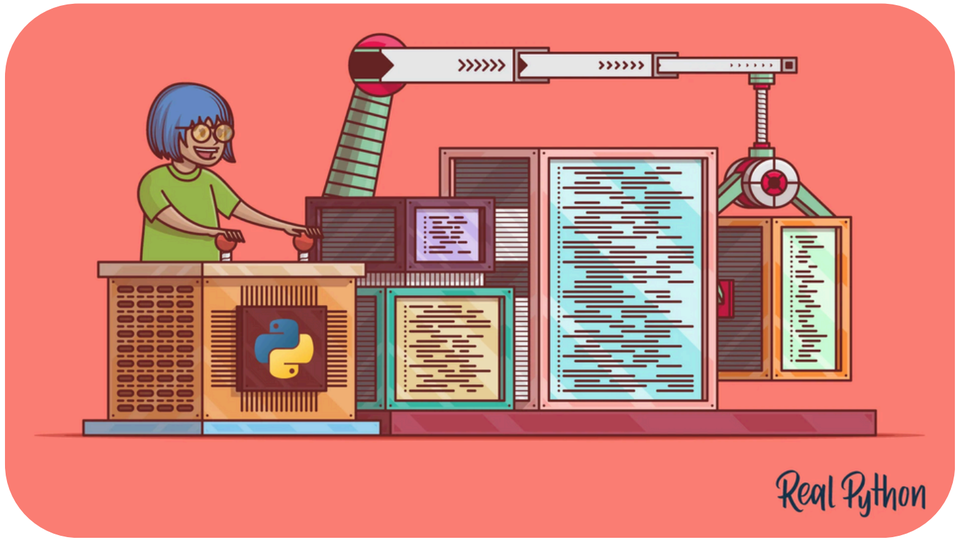
- La **parte declarativa**, donde **definimos funciones** y otros elementos esenciales.
- El **cuerpo principal** o main, donde **se llaman** las funciones definidas y se ejecutan las instrucciones del programa.

¿Cómo se ordena?



Un **paquete en Python** es una forma de organizar varios módulos (archivos .py) en una carpeta para mantener el código más ordenado.

Algunos trucos y recomendaciones...



En **Python**, importar significa traer el código de otros archivos o módulos para poder usarlo en tu programa. Para ello usamos la palabra reservada **import**

Si un módulo tiene un nombre largo o complicado, puedes darle un alias más corto al importarlo usando **as**

```
import moduloconnombrelargo as m
print(m.hola)
```

Algunos trucos y recomendaciones...

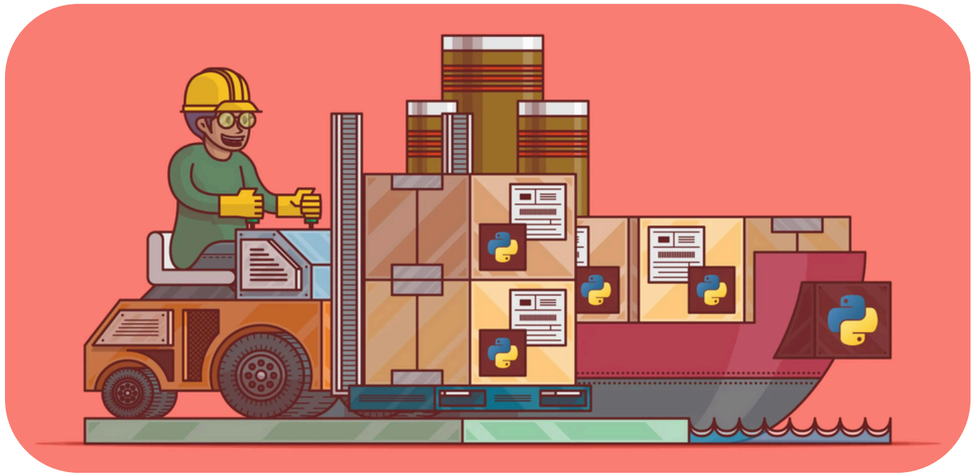
Si solo necesitas una función o clase de un módulo, **importa solo eso** en lugar de todo el módulo. Esto hace que tu código sea más **claro y ligero.**

```
from math import sqrt
result = sqrt(25)
```

Importar todo (*) puede parecer útil, pero puede **generar conflictos** si diferentes módulos tienen funciones con el mismo nombre.

```
from math import *  # NO recomendado, puede haber confusión
```

Algunos trucos y recomendaciones...



A veces los módulos estándar de Python ya cubren lo que necesitas. Evalúa bien antes de usar módulos externos.

Importar dentro de una función o método puede hacer el código más lento si la función se llama muchas veces. **EVITALO**

```
def my_function():
    import random
    return random.choice([1, 2, 3])
```

Algunos trucos y recomendaciones...

```python
def suma(a, b):
    return a + b

if (__name__ == '__main__'):
    c = suma(1, 2)
    print("La suma es:", c)
```

```python
# otromodulo.py
import modulo
```

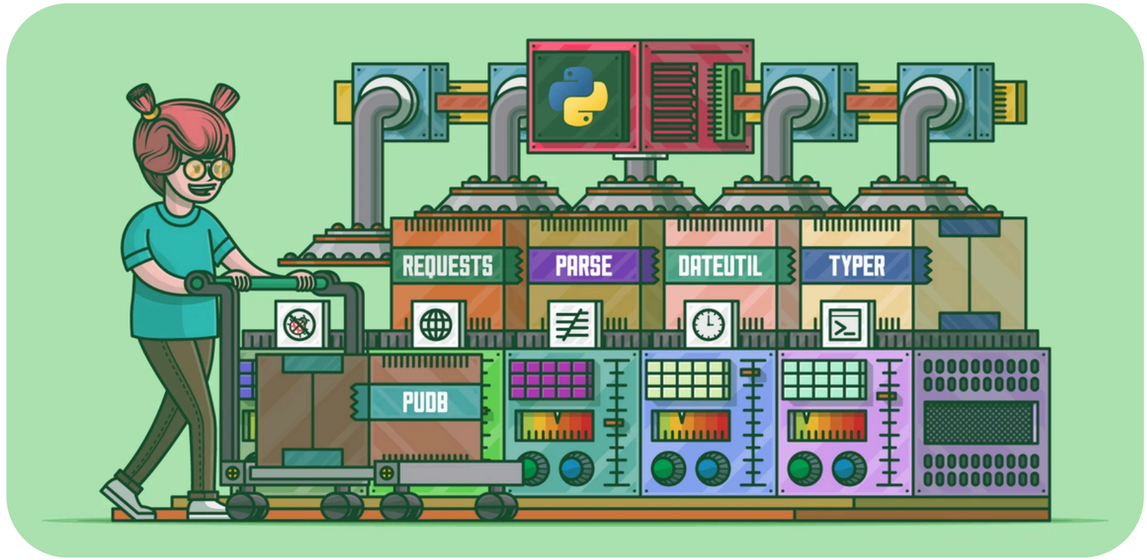
Al importar un módulo, el código fuera de funciones se ejecuta automáticamente. Para evitarlo, usamos la condición

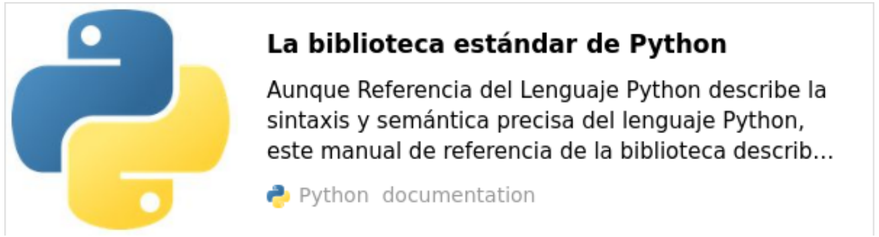
if __name__ == '__main__'

que asegura que el código solo se ejecute si el archivo es el programa principal, no cuando es importado.

L E

¿Qué es la libreria estandar en python?





Es un **conjunto de módulos y paquetes preinstalados** que vienen con Python.
Es como una **caja de herramientas lista para usar**, que te ayuda a no tener que escribir todo desde cero. SIrve para resolver problemas comunes, como manejo de archivos, procesamiento de texto, fechas, matemáticas, redes, entre otros.

PYTHON PACKAGE INDEX

¿Qué son los paquetes externos en python?

Son **módulos o librerías creados por la comunidad** o terceros que no vienen incluidos en la instalación básica de Python.

Estos paquetes ofrecen funcionalidades adicionales o especializadas que **no están** en la librería estándar, como herramientas para trabajar con datos, redes, inteligencia artificial, entre otros.

¿Qué es pip?

Es el **gestor de paquetes oficial de Python**. Permite **instalar**, actualizar y gestionar paquetes externos (módulos y librerías) de manera sencilla. Con pip, puedes acceder a una amplia variedad de paquetes disponibles en el **Python Package Index (PyPI)**

PYTHON PACKAGE INDEX

Comandos basicos de pip

- Instalar un paquete.
- Actualizar un paquete.
- Desinstalar un paquete.
- Listar paquetes instalados.
- Guardar la lista de paquetes instalados en un archivo.
- Instalar paquetes desde un archivo de requisitos.

```
pip install paquete
pip install --upgrade paquete
pip uninstall paquete
pip list
pip freeze > requirements.txt
pip install -r requirements.txt
```

¿Qué es requirements.txt?

En un proyecto, es común utilizar **varias librerías** que deben estar instaladas en todas las computadoras donde se ejecutará el proyecto.

Para facilitar la instalación de estas librerías, es una **buena práctica crear un archivo de texto** que contenga la lista de librerías junto con sus versiones correspondientes, si es necesario.

Esto permite que los usuarios puedan instalar todas las dependencias de manera rápida y sencilla.

```
annotated-types==0.7.0
anyio==4.6.2.post1
beautifulsoup4==4.12.3
certifi==2024.8.30
click==8.1.7
click-spinner==0.1.10
colorama==0.4.6
croniter==1.4.1
dnspython==2.7.0
email_validator==2.2.0
fastapi==0.115.2
fastapi-cli==0.0.5
greenlet==3.1.1
h11==0.14.0
httpcore==1.0.6
httptools==0.6.4
httpx==0.27.2
idna==3.10
```

IITA

¡Muchas Gracias!

Por su atención