Librerías

IITA 2023

Primero...un repaso:

- Python...¿Qué es?
- SOFTWARE
- HARDWARE
- LENGUAJE DE PROGRAMACIÓN
- PROGRAMA
- TIPO DE DATO
- VARIABLE

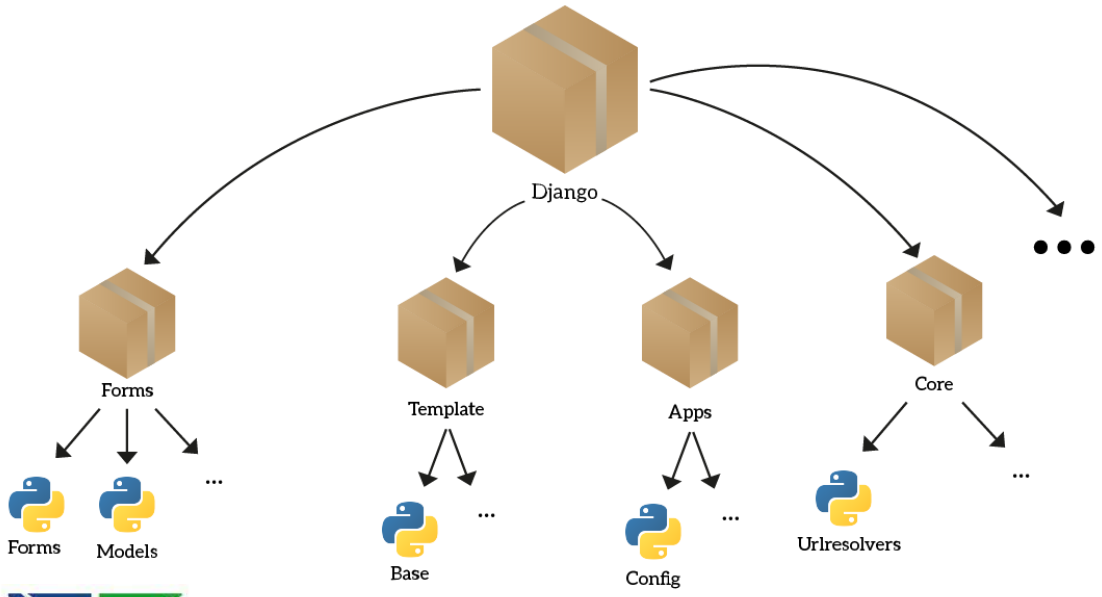
- Estructuras:
  - Selectivas
    - Simples
    - Múltiples
  - Repetitivas

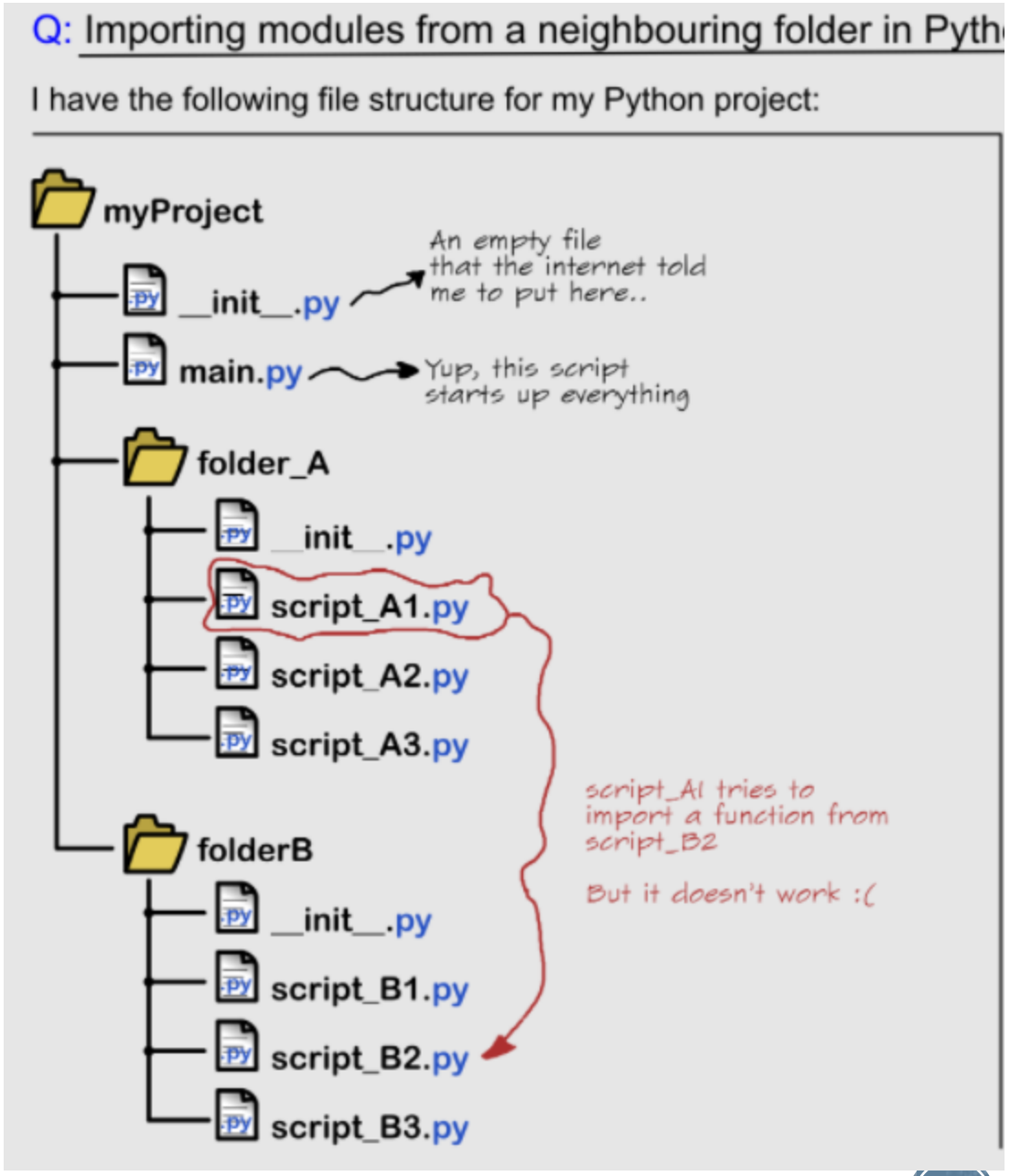
- Modularidad
  - Funciones
  - Procedimientos
  - Ámbito de las variables

- Listas
  - Operaciones
- Tuplas
- Diccionarios
- Conjuntos

Módulos

- En Python, un módulo es un archivo ".py" que contiene definiciones de funciones, variables y otras herramientas, que pueden ser llamadas por otro archivo ".py"





IITA 2023

Módulos

- Cuando programamos en Python, nuestro código tiene dos partes:
  - La “parte declarativa” donde definimos funciones y otros elementos, y
  - El “cuerpo principal” (main) donde llamamos funciones y ejecutamos instrucciones.

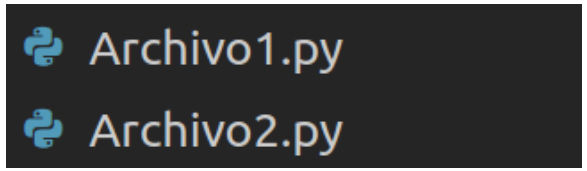
Clase 5 > mi_modulo.py > ...

```python
#mi_modulo.py
'''
    PARTE DECLARATIVA
'''
def fib(n): # los primeros n numeros de lo
    a, b = 0, 1
    for numero in range(n):
        print(b, end=' ')
        a, b = b, a+b
    print()
'''
    CUERPO PRINCIPAL
'''

print("Bienvenido al modulo! :) ")
```

Módulos

- Cualquier archivo .py que escribamos puede funcionar como módulo. Para llamar un módulo desde otro utilizamos la palabra reservada import. Una vez importado el módulo, podemos acceder a sus elementos usando un punto (.) y el nombre de la función/variable que queramos llamar. Por ejemplo:




```python
#Archivo 1

def MiFuncion():
    print("Holaa")
```

```python
#Archivo 2

import Archivo1
Archivo1.MiFuncion()
```

```
/Python/Archivo2.py
Holaa
```

Módulos: consideraciones

- **Al importar un módulo, el código que esté en su cuerpo principal (si existe) será ejecutado**

  →Por eso, a menos que sea necesario ejecutar código al momento de importar -por ejemplo, para crear archivos necesarios o inicializar variables- una buena práctica es que nuestros módulos solo contengan definiciones y solamente ejecutemos códigos desde un archivo principal

- El import del ejemplo funcionará siempre y cuando mi_modulo.py esté en la MISMA carpeta que el archivo principal.py. Este comportamiento se puede cambiar de dos formas:
  → Modificando la variable nativa de Python sys.path
  → Estructurando nuestro código mediante paquetes. Un paquete es una colección de módulos que se organizan de una determinada manera dentro de una carpeta (pueden estar todos los módulos dentro de la carpeta principal o dentro de subcarpetas, creando una jerarquía de módulos)

Módulos: consideraciones

- A modo de ejemplo, un paquete puede tener la siguiente estructura:

```
paquete/
        paquetito1/
                modulo1.py
                modulo2.py
        paquetito2/
                modulo1.py
                modulo2.py
```

→ Para llamar módulos contenidos en una estructura como esta, sepamos los nombres de los paquetes con un PUNTO:

```
import paquete.paquetito1.modulo2

paquete.paquetito1.modulo2.funcion1()
```

→ Para hacerlo menos tedioso, podemos hacer:

```
from paquete.paquetito1 import modulo2

modulo2.funcion1()
```

Librería estándar en Python (LE)

- La librería estándar en Python, incluye otros paquetes como ser:
  - Math → funciones matemáticas

```
1   import math
2   print(math.sin(25))
```

  - Random → manejo de números (pseudo) aleatorios
  - Datetime → manejo de fechas y horas
  - Smtplib → envío y manejo de correo electrónico
  - Y muchas más: https://docs.python.org/es/3/library/index.html

Paquetes fuera de la LE

- Si estamos buscando una funcionalidad más específica que la que encontramos en la librería estándar, tenemos que descargar los paquetes que necesitemos y colocarlos en un lugar desde el cual podamos llamarlos sin problemas.
- Para este fin, existe en internet un repositorio central de todos los paquetes creados por terceros y publicados para el resto de la comunidad conocido como el Python Package Index. Podemos buscar este repositorio en la web y consultar los paquetes por categoría para encontrar el deseado, o podemos usar herramientas que faciliten este proceso tales como pip.
    - pip install <modulo_name>

Instalacion de librerias

- En un proyecto por lo general se usan muchas librerias, las cules tiene que ser instaldas en todas las computadoras en las que se quieran correr estos proyectos.
- Una buena practica para facilitar la instalacion de estas librerias de LE, es en un archivo de texto nombrar las librerias con su respectiva version si es necesario.

Librerias.txt

```
1   numpy == 1.7.2
2   pyqt5
3
4
```

```
seba@DLNB163:~/Documents/Python$ pip3 install -r /home/seba/Documents/Python/Librerias.txt
Defaulting to user installation because normal site-packages is not writeable
Collecting numpy==1.7.2
  Downloading numpy-1.7.2.zip (3.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 1.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... /
```