Tema 3:

Versiones y ecosistema de Python (2h)

2. Versiones y ecosistema de Python

2.1. Evolución de Python: Python 2 vs Python 3

2.2. Versiones actuales y ciclo de soporte

2.3. PEPs (Python Enhancement Proposals) y su importancia

2.4. Entorno de paquetes y PyPI

2.5. Modos de ejecución

2.6. Parte práctica (ejemplos y ejercicios)

2.7. Distribución sugerida del tiempo

Profesor: Salvador Martínez Bolinches

Centro: IES Font de Sant Lluís

Año: 2025

Objetivos de aprendizaje

- Conocer la evolución histórica de Python (2.x → 3.x).
- Identificar diferencias clave entre versiones.
- Explorar el ciclo de vida de versiones (release, LTS, EOL).
- Familiarizarse con la estructura del ecosistema: PyPI, PEPs, comunidad, frameworks.
- Comprender la importancia de la compatibilidad y la gestión de versiones.

2.1. Evolución de Python: Python 2 vs Python 3

Cuando Python nació (1991) se diseñó con simplicidad, pero con el tiempo aparecieron limitaciones. Esto llevó a la creación de **Python 3** en 2008, que **rompió la compatibilidad con Python 2** para solucionar problemas de diseño histórico.

Diferencias principales entre Python 2 y 3

- **Impresión en pantalla**:
  - Python 2 → `print "Hola"`
  - Python 3 → `print("Hola")`
- **División de enteros**:
  - Python 2 → `5 / 2 = 2`
  - Python 3 → `5 / 2 = 2.5`
- **Cadenas de texto**:
  - Python 2 → las cadenas eran *ASCII* por defecto.
  - Python 3 → las cadenas son *Unicode* por defecto.

👉 Python 2 ya no se mantiene desde 2020. Actualmente **todo el desarrollo debe hacerse en Python 3**.

2.2. Versiones actuales y ciclo de soporte

- Python sigue un **ciclo de versiones** regular, con nuevas versiones **mayores cada ~18 meses**.
- Cada versión mayor (ej. 3.9, 3.10, 3.11…) recibe actualizaciones de **corrección de errores y seguridad durante ~5 años**.
- En la actualidad, Python se encuentra en la rama **3.x**, que seguirá evolucionando sin cambiar de "3" por un largo tiempo.

📌 Ejemplo:

- Python 3.8 → lanzado en 2019 (soporte hasta 2024).
- Python 3.11 → lanzado en 2022 (vigente).
- Python 3.12 → lanzado en 2023.
- Python 3.13 → previsto para 2024.

👉 Conclusión: siempre que sea posible, **usar la versión más reciente estable**.

2.3. PEPs (Python Enhancement Proposals)

Las **PEPs** son documentos que proponen cambios o mejoras al lenguaje.

- PEP 8 → guía de estilo (muy influyente).
- PEP 20 → *The Zen of Python*.
- PEP 484 → introducción de *type hints* (anotaciones de tipo).

👉 Entender los PEPs es clave para comprender por qué ciertas decisiones se toman en el desarrollo de Python.

2.4. Entorno de paquetes y PyPI

Una de las mayores fortalezas de Python es su ecosistema de librerías.

- **PyPI (Python Package Index)** → repositorio oficial de paquetes de Python.
- Se accede con el gestor **`pip`**.

```
pip install requests
```

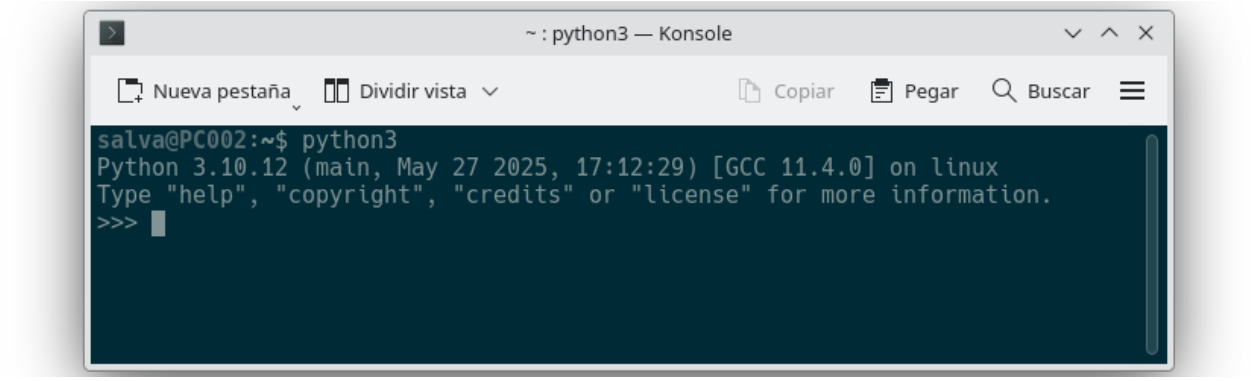
- Hoy en día existen más de **400.000 paquetes disponibles**.

Ejemplos de librerías populares:

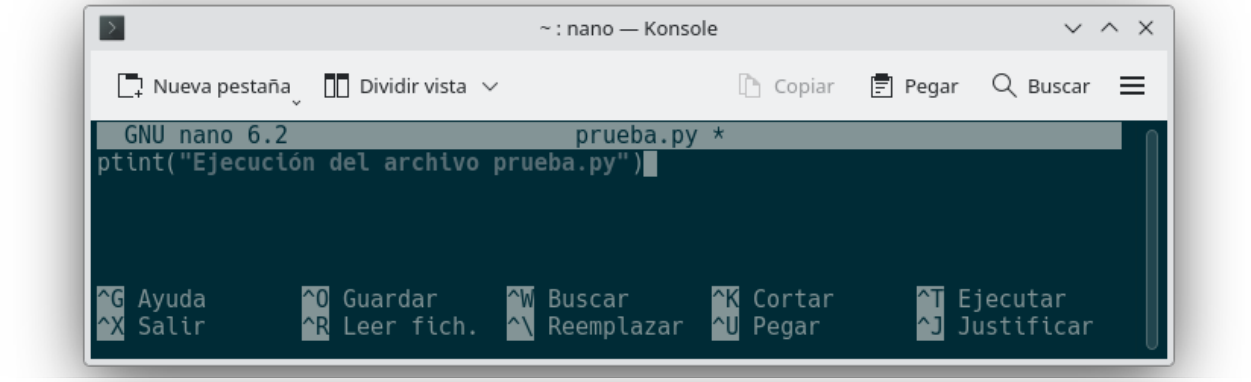
- *requests* → manejo de peticiones HTTP.
- *numpy* → cálculo numérico.
- *pandas* → análisis de datos.
- *flask* → desarrollo web.

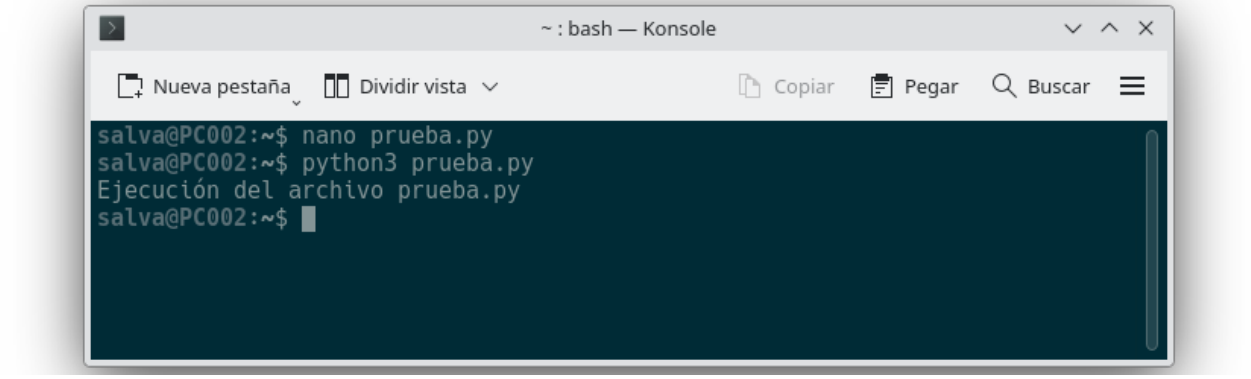
2.5. Modos de ejecución

1. Directamente desde el terminal o consola, mediante el intérpretecon python3 (Linux) o python (Windows):

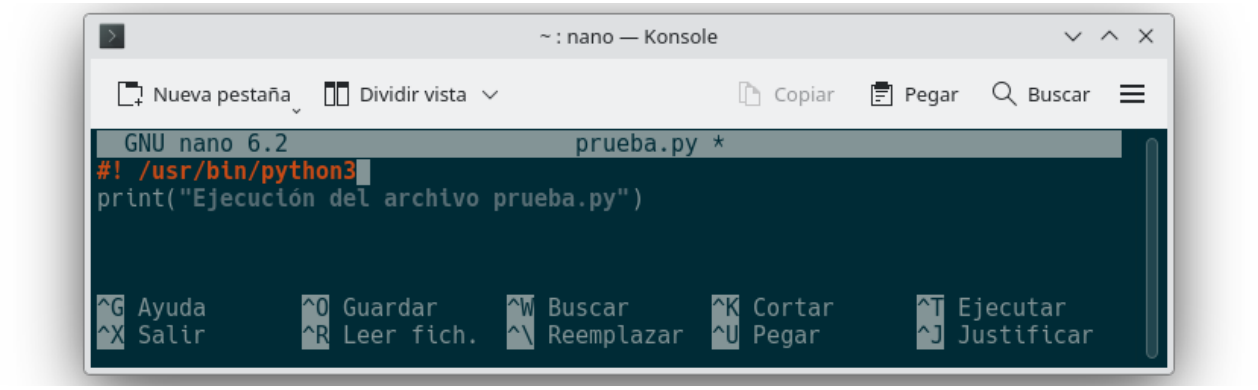


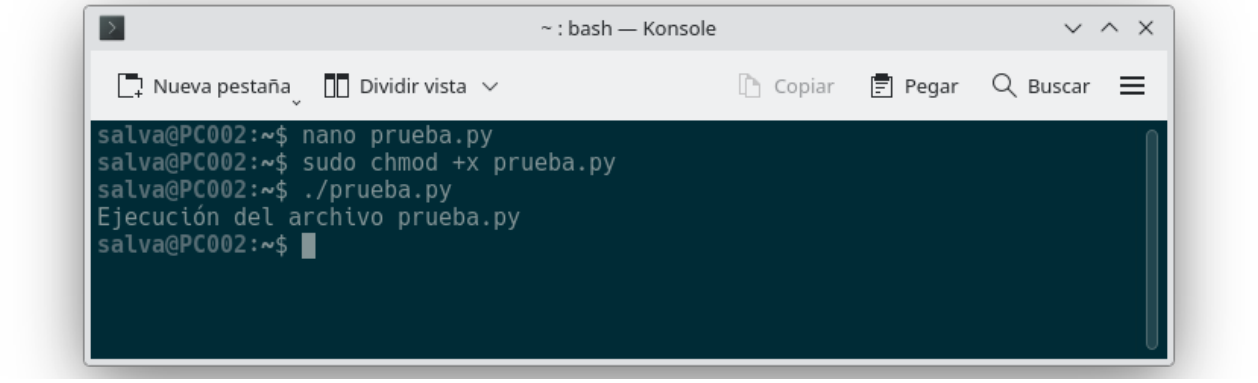
2. Escribiendo los comandos en un editor de textos y ejecutándolo desde el terminal o consola:



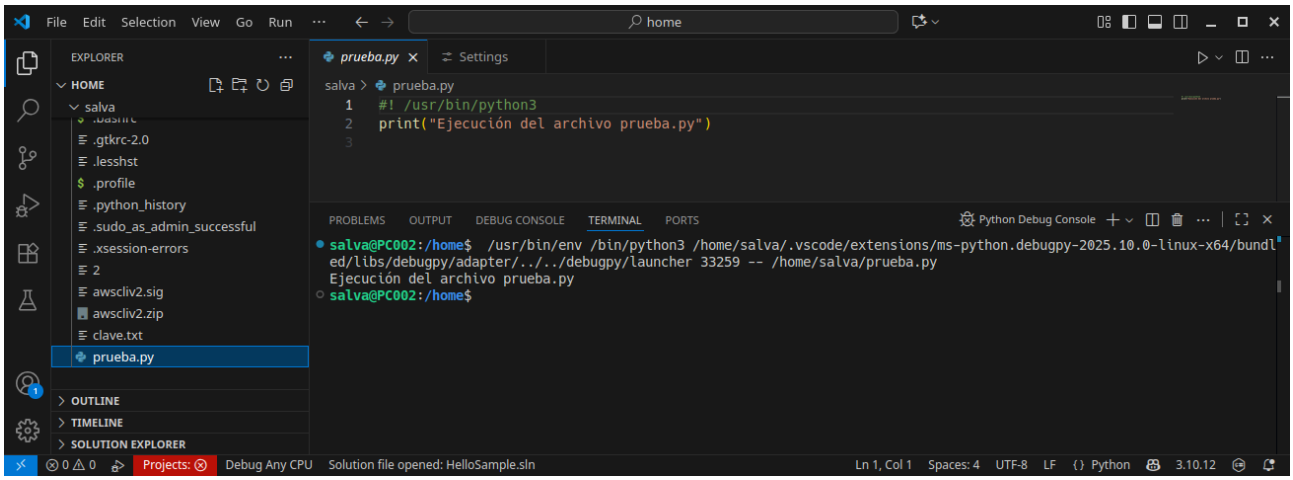


3. Escribiendo los comandos en un editor de textos, incluir el shebang (**#!**) con el ejecutable del intérprete, dándole permisos de ejecución y ejecutándolo desde el terminal o consola con **./archivo.py**:





4. Desde un IDE (VS Code, IDLE, Eclipse, …), donde puedes escribir los programas, depurarlos y ejecutarlos:



2.6. Parte práctica (ejemplos y ejercicios)

Ejemplo 1: Diferencia entre Python 2 y Python 3

```
# Python 3
print("5 dividido entre 2 =", 5 / 2)  # Resultado: 2.5
print("Hola, mundo")                  # Uso obligatorio de paréntesis
```

Ejemplo 2: Consultar versión instalada

```
python --version
```

En algunos sistemas:

```
python3 --version
```

Ejemplo 3: Instalar una librería desde PyPI

```
pip install requests
```

Y luego probar en Python:

```
import requests
resp = requests.get("https://httpbin.org/get")
print(resp.status_code)
```

Ejercicios propuestos

1. ¿Qué diferencias encontráis entre Python 2 y Python 3 respecto a Unicode?
2. Lee el resumen de la PEP 8. ¿Por qué es importante un estándar de estilo.
3. Lee el Zen of Python (PEP 20) y elige tu aforismo favorito, explicando por qué.
4. Instala la librería `requests` y haz una petición a la web del instituto/universidad.
5. Instala la librería `emoji` y muestra un texto con un emoji en consola.
6. Consulta en tu ordenadore qué versión de Python tenéis instalada.
7. Averigua en la web oficial cuál es la versión estable actual.
8. Ciclo de vida. Define: *release*, *bugfix release*, *end of life (EOL)*.
9. ¿Qué es PyPI y qué función cumple en el ecosistema Python?
10. Interpreta el número `3.11.4`: ¿qué representa cada dígito?

2.7. Distribución sugerida del tiempo (3h)

- **Teoría (1,5h)** → evolución, versiones, PEPs, PyPI.
- **Ejemplos guiados (0,5h)** → diferencias Python 2/3, instalación de paquetes.
- **Ejercicios prácticos y debate (1h)** → actividades en parejas y exposición en clase.