Tema 9:

Primeros pasos en la programación con Python (8h)

9. Primeros pasos en la programación con Python (8h)

9.1. Creación y ejecución de scripts

9.2. Organización del código en módulos y paquetes básicos

9.3. Uso de la librería estándar (math, os, sys, datetime)

9.4. Ejercicios prácticos integradores

9.5. Parte práctica (ejemplos y ejercicios)

9.6. Distribución sugerida del tiempo

Profesor: Salvador Martínez Bolinches

Centro: IES Font de Sant Lluís

Año: 2025

9.1. Creación y ejecución de scripts

Hasta ahora hemos usado el **intérprete interactivo** o IDLE. Ahora damos el paso a **crear programas completos**.

Crear un script

1. Abrir un editor (VS Code, PyCharm, etc.).
2. Escribir el código en un archivo con extensión `.py`.
3. Ejecutar el archivo:

```
python mi_programa.py
```

Ejemplo:

```
print("Bienvenidos al curso de Python")
```

9.2. Organización del código en módulos y paquetes básicos

- **Módulo**: archivo `.py` que contiene funciones, clases o variables.
- **Paquete**: carpeta que contiene varios módulos y un archivo `__init__.py`.

Ejemplo de módulo `saludos.py`:

```
def hola(nombre):
    return f"Hola, {nombre}!"
```

Usarlo en otro archivo:

```
import saludos
print(saludos.hola("Ana"))
```

👉 La modularización permite **reutilizar código** y **organizar proyectos grandes**.

9.3. Uso de la librería estándar

Python incluye una **librería estándar muy amplia** (*batteries included*). No requiere instalación extra.

Ejemplos comunes

- **math** → operaciones matemáticas.

```python
import math
print(math.sqrt(25))  # 5.0
```

- **os** → interacción con el sistema operativo.

```python
import os
print(os.getcwd())   # muestra directorio actual
```

- **sys** → interacción con el intérprete.

```python
import sys
print(sys.version)
```

- **datetime** → fechas y horas.

```python
from datetime import datetime
print(datetime.now())
```

👉 Conocer la librería estándar evita instalar dependencias innecesarias.

9.4. Ejercicios prácticos integradores

Ejercicio 1: Gestor de tareas

- Crea un programa que permita:
  - Añadir tareas con nombre, descripción y fecha límite.
  - Listar todas las tareas.
  - Marcar una tarea como completada.
  - Guardar y cargar las tareas desde un archivo (`tareas.txt` o `.json`).
- Usa:
  - `datetime` para manejar fechas.
  - `os` para verificar la existencia del archivo.
  - Modularización (por ejemplo: `tareas.py` y `main.py`).

Ejercicio 2: Conversor de unidades

- Programa un conversor con menú interactivo:
  - Conversión de temperatura (°C ↔ °F ↔ K).
  - Conversión de monedas (introducir tasa de cambio).
  - Conversión de longitudes (m ↔ cm ↔ km).
- Usa funciones separadas en un módulo (`conversor.py`).
- Implementa control de errores con `try/except`.

Ejercicio 3: Analizador de texto

- El usuario introduce una frase o texto.
- El programa debe mostrar:
  - Número de palabras y caracteres.
  - Palabra más larga.
  - Frecuencia de cada letra (usando un diccionario).
- Usa:
  - `sys.argv` para aceptar un archivo de texto como argumento.
  - `collections.Counter` (de la librería estándar).

Ejercicio 4: Agenda de contactos avanzada

- Mejora la agenda del punto 9.5:
    - Guarda los contactos en un archivo JSON.
    - Permite editar y eliminar contactos.
    - Añade validación de formato de teléfono y correo.
- Divide el código en varios módulos:
    - `agenda.py` (funciones CRUD)
    - `utilidades.py` (validaciones)
    - `main.py` (interfaz principal)

Ejercicio 5: Mini proyecto final — Calculadora científica

- Menú con operaciones básicas y avanzadas (seno, coseno, raíz cuadrada, potencia, factorial…).
- Usa `math` para las operaciones.
- Control de errores (entrada inválida, división entre cero).
- Estructura modular: `operaciones.py`, `menu.py`, `main.py`.
- Documenta con docstrings y sigue la guía PEP 8.

9.5. Parte práctica (ejemplos y ejercicios)

Ejemplo 1: Script interactivo

```
nombre = input("¿Cómo te llamas? ")
print("Encantado de conocerte,", nombre)
```

Ejemplo 2: Módulo propio

Archivo `calculadora.py`:

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

Archivo `main.py`:

```
import calculadora
print(calculadora.suma(4, 5))
```

Ejemplo 3: Librería estándar

```
from datetime import date

hoy = date.today()
print("La fecha de hoy es:", hoy)
```

Ejercicios propuestos

1. **Script simple**
    - Crear un script `bienvenida.py` que pida nombre y edad al usuario.
    - Mostrar un mensaje personalizado.
2. **Módulo y reutilización**
    - Crear un módulo `operaciones.py` con funciones de suma, resta, multiplicación y división.
    - Usarlo en otro script `main.py`.
3. **Explorando la librería estándar**
    - Usar `math` para calcular el área de un círculo.
    - Usar `datetime` para mostrar la fecha de nacimiento del alumno como objeto de fecha.
    - Usar `os` para listar los archivos del directorio actual.
4. **Diccionario como agenda**
    - Crear un script que funcione como **agenda telefónica**:
        - Permita añadir contactos.
        - Permita buscar un número por nombre.
        - Permita eliminar un contacto.
5. **Proyecto integrador**
    - Crear un programa `calculadora_avanzada.py` que:
        - Ofrezca un menú de opciones al usuario (suma, resta, multiplicación, división).
        - Pida dos números y ejecute la operación.
        - Maneje errores como entrada inválida o división entre cero.
        - Esté documentado con **docstrings** y siga la guía **PEP 8**.

9.6. Distribución sugerida del tiempo (8 h)

- **Teoría (1 h)** → scripts, módulos, paquetes, librería estándar.
- **Ejemplos guiados (0,5 h)** → creación de módulos, uso de librerías estándar, interacción con el sistema.
- **Ejercicios prácticos (6,5 h)** → trabajo individual y grupal, con proyectos pequeños.