Tema 8:

Estándares de calidad y buenas prácticas (3 h)

8. Estándares de calidad y buenas prácticas

8.1. Guía de estilo PEP 8

8.2. Documentación de código: docstrings

8.3. Tipado estático opcional con type hints (PEP 484)

8.4. Herramientas de formateo y análisis: black, flake8, pylint

8.5. Parte práctica (ejemplos y ejercicios)

8.6. Distribución sugerida del tiempo

Profesor: Salvador Martínez Bolinches

Centro: IES Font de Sant Lluís

Año: 2025

Objetivos de aprendizaje

- Conocer y aplicar las guías de estilo de Python (PEP 8 y PEP 257).
- Usar nombres de variables y funciones significativos.
- Comentar y documentar el código adecuadamente.
- Introducir docstrings en funciones, clases y módulos.
- Mantener un estilo de código limpio, legible y consistente.

8.1. Guía de estilo PEP 8

La **PEP 8 (Python Enhancement Proposal 8)** es la guía oficial de estilo de Python. Su objetivo es mejorar la **legibilidad** y la **consistencia** del código.

Principales recomendaciones:

- Indentación → **4 espacios** por nivel.
- Longitud de línea → máximo **79 caracteres**.
- Espacios alrededor de operadores → `a = b + c`.
- Nombres de variables y funciones en **snake_case**: `calcular_area()`.
- Nombres de clases en **CamelCase**: `class Persona`.
- Constantes en mayúsculas: `PI = 3.1416`.
- Usar **docstrings** para documentar módulos, clases y funciones.

Ejemplo correcto:

```python
def calcular_area_circulo(radio):
    """Devuelve el área de un círculo dado su radio."""
    import math
    return math.pi * radio ** 2
```

Ejemplo incorrecto:

```python
def CalcularAreaCirculo(RADIO):return 3.14*RADIO*RADIO
```

8.2. Documentación de código: *docstrings*

Los **docstrings** son cadenas de texto asociadas a módulos, clases, funciones o métodos.

Ejemplo:

```python
def fahrenheit_a_celsius(f):
    """
    Convierte grados Fahrenheit a Celsius.

    Parámetros:
    f (float): temperatura en grados Fahrenheit.

    Retorna:
    float: temperatura en grados Celsius.
    """
    return (f - 32) * 5/9
```

Se pueden consultar en tiempo de ejecución con `help()`:

```python
help(fahrenheit_a_celsius)
```

8.3. Tipado estático opcional con *type hints* (PEP 484)

Python es dinámico, pero admite **anotaciones de tipo** para mejorar la claridad y facilitar herramientas de análisis.

Ejemplo:

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

- No obligan al intérprete a comprobar los tipos.
- Sirven como ayuda para el programador y para editores/analizadores.

8.4. Herramientas de formateo y análisis

- **Black** → formateador automático.

```
pip install black
black archivo.py
```

- **Flake8** → herramienta de análisis de estilo.

```
pip install flake8
flake8 archivo.py
```

- **Pylint** → análisis estático avanzado, da puntuación al código.

```
pip install pylint
pylint archivo.py
```

👉 Estas herramientas ayudan a mantener **estándares de calidad** en proyectos colaborativos.

8.5. Parte práctica (ejemplos y ejercicios)

Ejemplo 1: Uso de docstrings

```python
def area_rectangulo(base: float, altura: float) -> float:
    """Calcula el área de un rectángulo."""
    return base * altura

help(area_rectangulo)
```

Ejemplo 2: Uso de Black

1. Escribir un script con mala indentación.
2. Ejecutar:

   ```
   black archivo.py
   ```

3. Observar cómo se reformatea automáticamente.

Ejemplo 3: Uso de Flake8

1. Escribir un script con estilo incorrecto.
2. Ejecutar:

   ```
   flake8 archivo.py
   ```

3. Analizar los avisos y corregir.

Ejercicios propuestos

1. Escribe una función que calcule el área de un triángulo. Revisa que cumpla con las reglas de PEP 8 (indentación, nombres, docstrings).
2. Documenta una función que convierta grados Celsius a Kelvin. Prueba `help(función)` para verificar la documentación.
3. Añade anotaciones de tipo a una función que reciba una lista de números y devuelva su promedio.
4. Instalar `pylint` y analizar un archivo. Mejorar la puntuación del código corrigiendo los problemas detectados.
5. Escribe un pequeño módulo `geometria.py` con funciones para:
   - Área de un círculo.
   - Área de un rectángulo.
   - Área de un triángulo.

   Añade docstrings y type hints. Pasa Black y Flake8 para verificar la calidad.
6. ¿Cuál es la longitud máxima recomendada de línea en PEP 8?
7. ¿Qué trata la PEP 257 y en qué se diferencia de PEP 8?
8. Escribe una constante llamada `PI` con valor `3.14159 y úsala en un programa.`
9. Explica por qué el siguiente comentario es innecesario:

   x = x + 1 # suma 1 a x
10. Identifica 3 errores de estilo en el siguiente código:

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

8.6. Distribución sugerida del tiempo (3 h)

- **Teoría (0,5 h)** → PEP 8, docstrings, type hints, herramientas.
- **Ejemplos guiados (0,5 h)** → docstrings, uso de Black, Flake8 y Pylint.
- **Ejercicios prácticos (2 h)** → aplicación de estándares y mejora de código existente.