

# Sistemas de Gestión de Calidad



EPET N°12



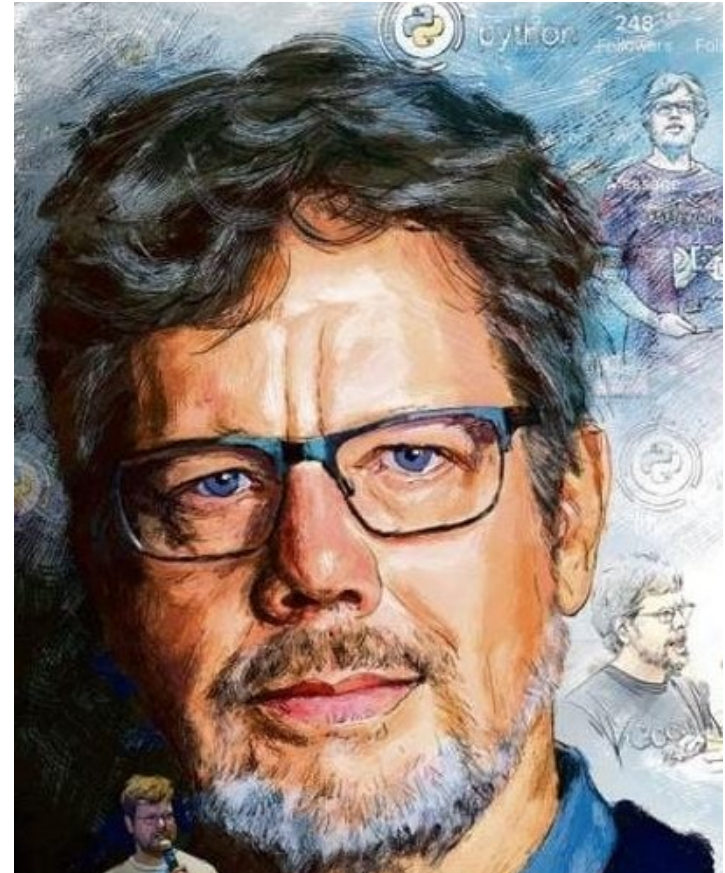
## **Buenas prácticas de codificación**

**Ing. Eduardo Kunysz**

# Orígenes y contexto

En los primeros días del desarrollo de Python, el código era escrito por una comunidad creciente de programadores que seguían diversas convenciones y estilos. Con el tiempo, se hizo evidente que una guía de estilo uniforme podría mejorar la legibilidad y la mantenibilidad del código.

Guido Van Rossum, un programador de computación de los Países Bajos, creó Python. Python comenzó en 1989 en el Centrum Wiskunde & Informatica (CWI), en principio como un proyecto de afición para mantenerse ocupado durante las vacaciones de Navidad.



# Orígenes y contexto

En 2001, **Guido van Rossum**, el creador de Python, y la comunidad Python decidieron formalizar una guía de estilo. Esta iniciativa se materializó en el PEP 8 (**Python Enhancement Proposal 8**), que fue publicado para proporcionar un conjunto de recomendaciones sobre cómo escribir código Python que sea claro y coherente.



# Ejemplos

```
def func(a,b):return a+b
def printresult(x):print("The result is",x)
def main():
    a = 5
    b=10
    result=func(a,b)
    printresult(result)
main()
```

**Código sin estilo**

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

def print_result(x):
    print("The result is", x)

def main():
    a = 5
    b = 10
    result = add_numbers(a, b)
    print_result(result)

if __name__ == "__main__":
    main()
```

**Código con estilo PEP8**

# Principios Fundamentales

PEP 8 se basa en varios principios clave:

- **Consistencia:** Mantener un estilo uniforme dentro de un proyecto es esencial.
- **Legibilidad:** El código debe ser fácil de leer y entender, incluso para quienes no lo escribieron.
- **Compatibilidad:** Evitar romper la compatibilidad con versiones anteriores de Python cuando sea posible.

# Nombres Descriptivos

Nombres de Variables y Funciones:

Usa nombres descriptivos que reflejen el propósito de la variable o función.

Ejemplo:



- Malo: `x, y, temp` 
- Bueno: `user_age, calculate_total_price, temp_in_celsius` 

# Convención de Nombres

Convención de Nombres (snake\_case):

- Variables y funciones: Usa snake\_case (minúsculas separadas por guiones bajos).

Ejemplo:

- Malo: `totalPrice, UserAge` 
- Bueno: `total_price, user_age` 

# Espaciado y Sangrías

Uso de 4 Espacios para Sangrías:

- Usa 4 espacios para sangrías en lugar de tabulaciones.

Ejemplo:

- Malo:

```
def calculate_area(radius):  
    return 3.14 * radius * radius
```



- Bueno:

```
def calculate_area(radius):  
    return 3.14 * radius * radius
```






# Espacios alrededor de Operadores

Espacios alrededor de Operadores:


- Coloca un espacio antes y después de los operadores binarios.

Ejemplo:

- Malo:

$a+b=c-d$  

- Bueno:



$a + b = c - d$  

# Líneas en Blanco

Uso de Líneas en Blanco:

- Usa líneas en blanco para separar funciones y clases.

Ejemplo:

Malo:	Bueno:
<pre>def first_function():     pass def second_function():     pass</pre> 	<pre>def first_function():     pass  def second_function():     pass</pre> 

# Longitud de Línea

Longitud de Línea:

- Limita la longitud de cada línea a 79 caracteres.

Ejemplo:

- Malo:

```
def some_function_with_a_very_long_name(parameter_one, parameter_two, parameter_three, parameter_four):  
    pass
```



- Bueno:

```
def some_function_with_a_very_long_name(  
    parameter_one, parameter_two, parameter_three, parameter_four  
):  
    pass
```



# Resumen

Resumen de PEP8:

- Usa nombres descriptivos y snake\_case.
- Mantén la sangría con 4 espacios.
- Coloca espacios alrededor de operadores.
- Separa funciones y clases con líneas en blanco.
- Limita la longitud de línea a 79 caracteres.