

## TRABAJO FINAL

# PEP-8 Y GITHUB

## QUE ES?

PEP-8 es el estandar de programación que se usa en el lenguaje Python. Básicamente es un documento con una serie de lineamientos para escribir en Python de la mejor manera posible y seguir un estándar industrial.

## HISTORIA

PEP8 O PEP-8 fue creado alrededor del 2001 por Guido van Rossum, Barry Warsaw, and Nick Coghlan. Con el objetivo de mejorar los estándares y legibilidad del lenguaje de programación Python.

## CONVENCIÓN

Aquí la gente tiende a ser inconsistente.

Escoger buenos nombres te puede ahorrar mucho tiempo.

- `b` (single lowercase letter)
- `B` (single uppercase letter)
- `lowercase`
- `lower_case_with_underscores`

"Explicit is better than implicit."  
— The Zen of Python

## REFERENCIA

Python TM. (2021). PEP-8 Style Guide for Python Code. Consultado en: <https://www.python.org/dev/peps/pep-0008/>

## POR QUE IMPORTA?

El objetivo principal es mejorar la legibilidad y consistencia del código.

Así como evitar el uso de malas practicas o convenciones a la hora de escribir.

Imagínate si todos habláramos español de una manera distinta.  
"Code is read much more often than it is written."

## CODE LAY-OUT

Describe la manera correcta de escribir el código: indentación, teclas a usar, máximo de caracteres por línea, espacios en blanco, separación entre líneas, importaciones, entre otros, son ejemplos del Code-Lay-Out.

## COMENTARIOS

- Tomar como prioridad actualizar los comentarios conforme se cambie el código
- Los comentarios deben ser oraciones completas.
- Cada línea de un comentario de bloque comienza con un `#` y un solo espacio

Un comentario en línea es un comentario en la misma línea que una declaración

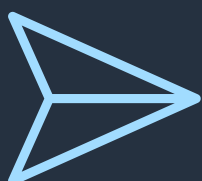
. Los comentarios en línea deben estar separados por al menos dos espacios de la declaración. Deben comenzar con un `#` y un solo espacio.

## TRABAJO FINAL

# PEP-8 Y GITHUB

### HAZ COMMITS PEQUEÑOS

Realizar commits más frecuentes y pequeños en ves de grandes pedazos de códigos



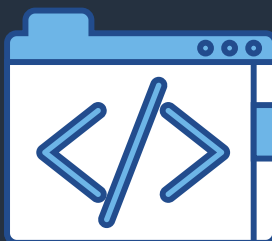
### ESCRIBE BUENOS MENSAJES DE COMMIT

Deben de ser cortos, en tiempo presente y explicación de por qué el cambio



### USA UN EDITOR DE CODIGO PARA MENSAJES DE COMMIT LARGOS

Asegurar que "subject line" y el cuerpo estén separados por una línea en blanco.



### REFERENCIA

Yossef N. (2017), Git Commit Best Practices, Consultado en:  
<https://medium.com/@nawarpianist/git-commit-best-practices-dab8d722de99>

### COMMIT COMPLETO Y BIEN PROBADO

No dejar código incompleto y asegurar antes haberlo corrido



### USA ESTILO IMPERATIVO

Evitar escribir en pasado y hacerlo en estilo imperativo



### APRENDE A USAR GIT CON LINEA DE COMANDO

Importante aprender primero desde la línea de comando y luego interfaz gráfica

