

PEP8: Guía de Estilo para Python

PEP8 es un conjunto de directrices de estilo para el código Python. Fue creado para promover la coherencia y legibilidad en el código Python. PEP8 no es una norma obligatoria, pero se recomienda ampliamente seguirlo para mejorar la calidad del código.

```
# Heading 1

## Heading 2

This is `inline code`.

```c
// fenced code block
void foo() {
 printf("bar");
}
...

Lists

- bold
- italic

Checklists

- [] TBD
- [x] Done

{
 "year": 2020,
 "sucks": true
}
```

# Importancia de PEP8

1

## Legibilidad

El código legible es más fácil de entender y mantener, lo que facilita la colaboración entre los desarrolladores.

2

## Consistencia

PEP8 garantiza que el código Python se vea y funcione de manera similar en diferentes proyectos.

3

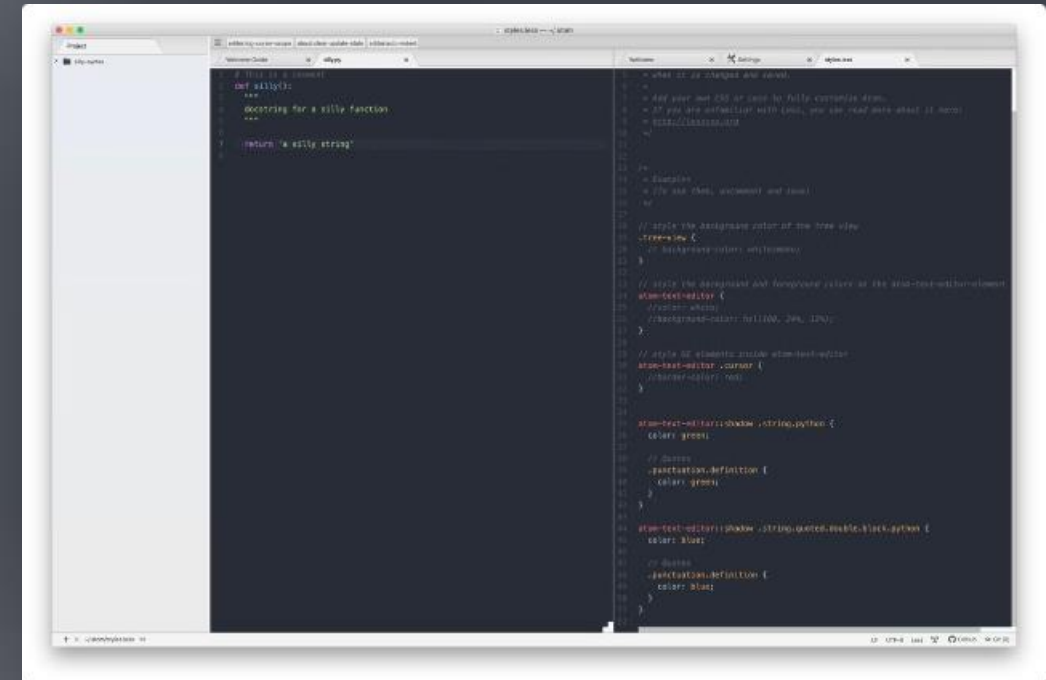
## Mantenimiento

El código consistente es más fácil de modificar y depurar, lo que reduce el tiempo y los costes de desarrollo.

4

## Eficiencia

Seguir las directrices de PEP8 puede ayudar a evitar errores comunes y a mejorar la eficiencia del código.



# Convenciones de Estilo

## Sangrías

Utiliza 4 espacios para cada nivel de sangría.

No utilices tabulaciones.

## Espaciado

Utiliza un espacio en blanco alrededor de operadores como ``=`` y ``+``.

Utiliza dos líneas en blanco para separar funciones y clases.

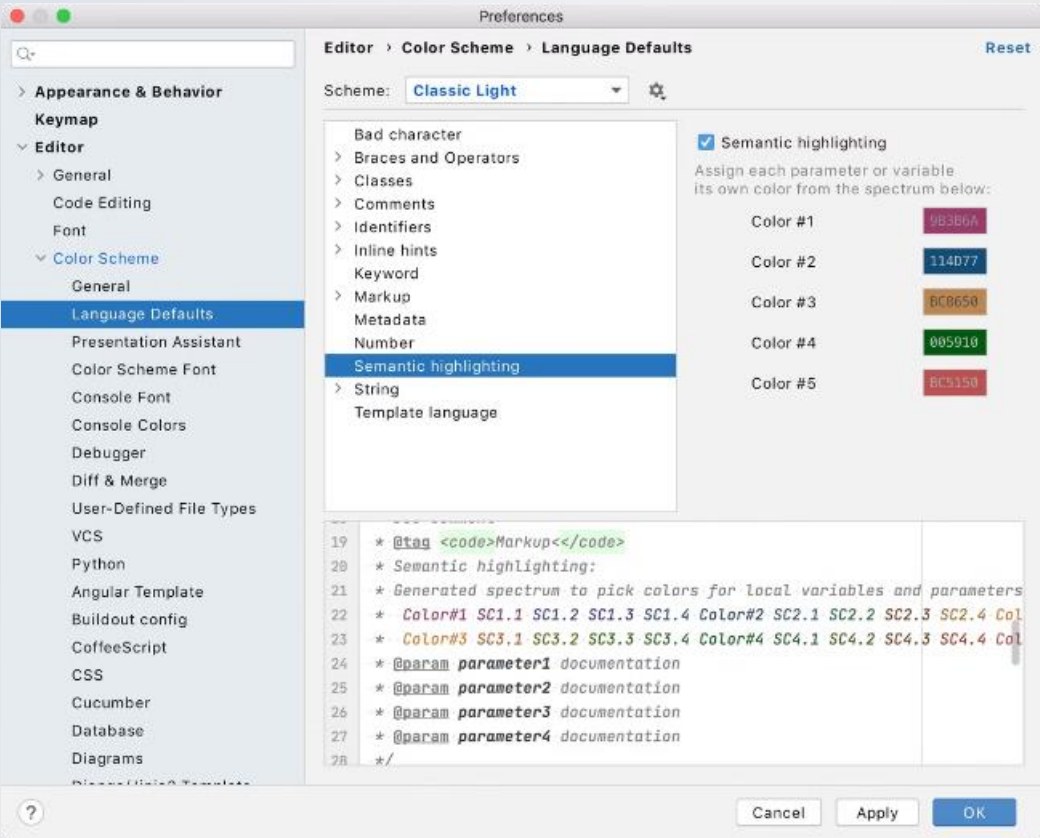
## Longitud de Líneas

Limita las líneas de código a 79 caracteres.

Utiliza barras invertidas (``\``) para continuar líneas largas.

# Nomenclatura de Variables y Funciones

Tipo	Ejemplo	Descripción
Variable	my_variable	En minúsculas, separadas por guiones bajos.
Función	my_function	En minúsculas, separadas por guiones bajos.
Constante	MY_CONSTANT	En mayúsculas, separadas por guiones bajos.
Clase	MyClass	CamelCase, comenzando con una letra mayúscula.



# Longitud de Líneas

1

## Límites

Mantén las líneas de código a 79 caracteres.

2

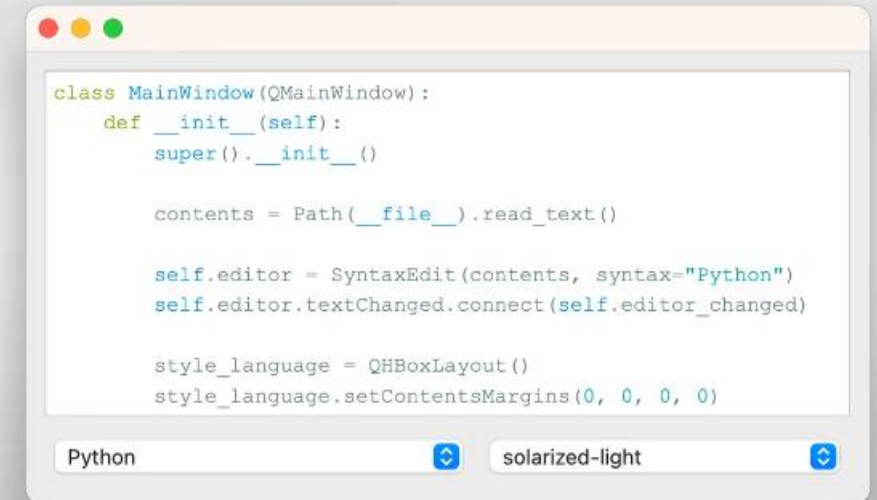
## Continuación

Usa barras invertidas (`\`) para continuar líneas largas.

3

## Paréntesis

Utiliza paréntesis para dividir líneas largas.



```
1 from dataclasses import dataclass
2
3
4 @dataclass
5 class Point:
6 x: int
7 y: int
8
9
10 def where_is(point):
11 match point:
12 case Point(x=0, y=0):
13 print("Origin")
14 case Point(x=0, y=y):
15 print(f"Y={y}")
16 case Point(x=x, y=0):
17 print(f"X={x}")
18 case Point():
19 print("Somewhere else")
20 case _:
21 print("Not a point")
22
```

# Espaciado y Sangrías

## Sangrías

Utiliza 4 espacios para cada nivel de sangría.

## Espaciado

Utiliza un espacio en blanco alrededor de operadores como `=` y `+`.

## Líneas en Blanco

Utiliza dos líneas en blanco para separar funciones y clases.

## Comentarios

Utiliza espacios en blanco para separar el código de los comentarios.

# Documentación y Comentarios



## Docstrings

Las docstrings documentan funciones, clases y módulos. Se utilizan para generar documentación.



## Comentarios en Línea

Los comentarios en línea se utilizan para explicar el código en una sola línea.



## Comentarios de Bloque

Los comentarios de bloque se utilizan para explicar secciones más grandes de código.

```
04-DynamicProgramming1 / SpidermansWorkout.py / 19 lines
22 for case in range(num_of_cases):
23 try:
24 number_of_moves = int(sys.stdin.readline().strip())
25 moves_str = sys.stdin.readline().strip().split(" ")
26 if number_of_moves > 0:
27 moves = [int(i) for i in moves_str]
28 sum_moves = sum(moves)
29 height = sum_moves//2+1
30
31 dp = [[None for _ in range(number_of_moves+1)] for _ in range(height)]
32
33 for h in range(height):
34 dp[h][0] = 0
35
36 start_points = {0}
```

PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL

SpidermansWorkout.py 04-DynamicProgramming1 19

- String statement has no effect pylint(pointless-string-statement) [Ln 5, Col 1]
- Unused variable 'case' pylint(unused-variable) [Ln 22, Col 6]
- Module name "SpidermansWorkout" doesn't conform to snake\_case naming style pylint(invalid-name) [Ln 1, Col 1]
- Missing module docstring pylint(missing-module-docstring) [Ln 1, Col 1]
- Line too long (108/100) pylint(line-too-long) [Ln 11, Col 1]
- Line too long (120/100) pylint(line-too-long) [Ln 12, Col 1]
- Missing function or method docstring pylint(missing-function-docstring) [Ln 19, Col 1]
- Variable name "dp" doesn't conform to snake\_case naming style pylint(invalid-name) [Ln 31, Col 5]
- Trailing whitespace pylint(trailing-whitespace) [Ln 32, Col 1]
- Variable name "h" doesn't conform to snake\_case naming style pylint(invalid-name) [Ln 33, Col 9]
- Variable name "el" doesn't conform to snake\_case naming style pylint(invalid-name) [Ln 39, Col 10]
- Comparison to None should be 'expr is None' pylint(singleton-comparison) [Ln 44, Col 11]
- Comparison to None should be 'expr is None' pylint(singleton-comparison) [Ln 52, Col 11]
- Trailing whitespace pylint(trailing-whitespace) [Ln 58, Col 1]
- Comparison to None should be 'expr is None' pylint(singleton-comparison) [Ln 59, Col 8]
- Variable name "up" doesn't conform to snake\_case naming style pylint(invalid-name) [Ln 66, Col 7]
- Comparison to None should be 'expr is not None' pylint(singleton-comparison) [Ln 70, Col 11]
- Comparison to None should be 'expr is not None' pylint(singleton-comparison) [Ln 72, Col 13]
- Final newline missing pylint(missing-final-newline) [Ln 104, Col 1]



# Beneficios de Seguir PEP8

## Legibilidad

El código legible es más fácil de entender y mantener.

1

2

## Colaboración

PEP8 facilita la colaboración entre desarrolladores al garantizar la consistencia del código.

## Mantenimiento

El código consistente es más fácil de modificar y depurar.

3

4

## Eficiencia

Seguir PEP8 puede ayudar a evitar errores comunes y mejorar la eficiencia del código.