

PEP 8

Estilos de Python

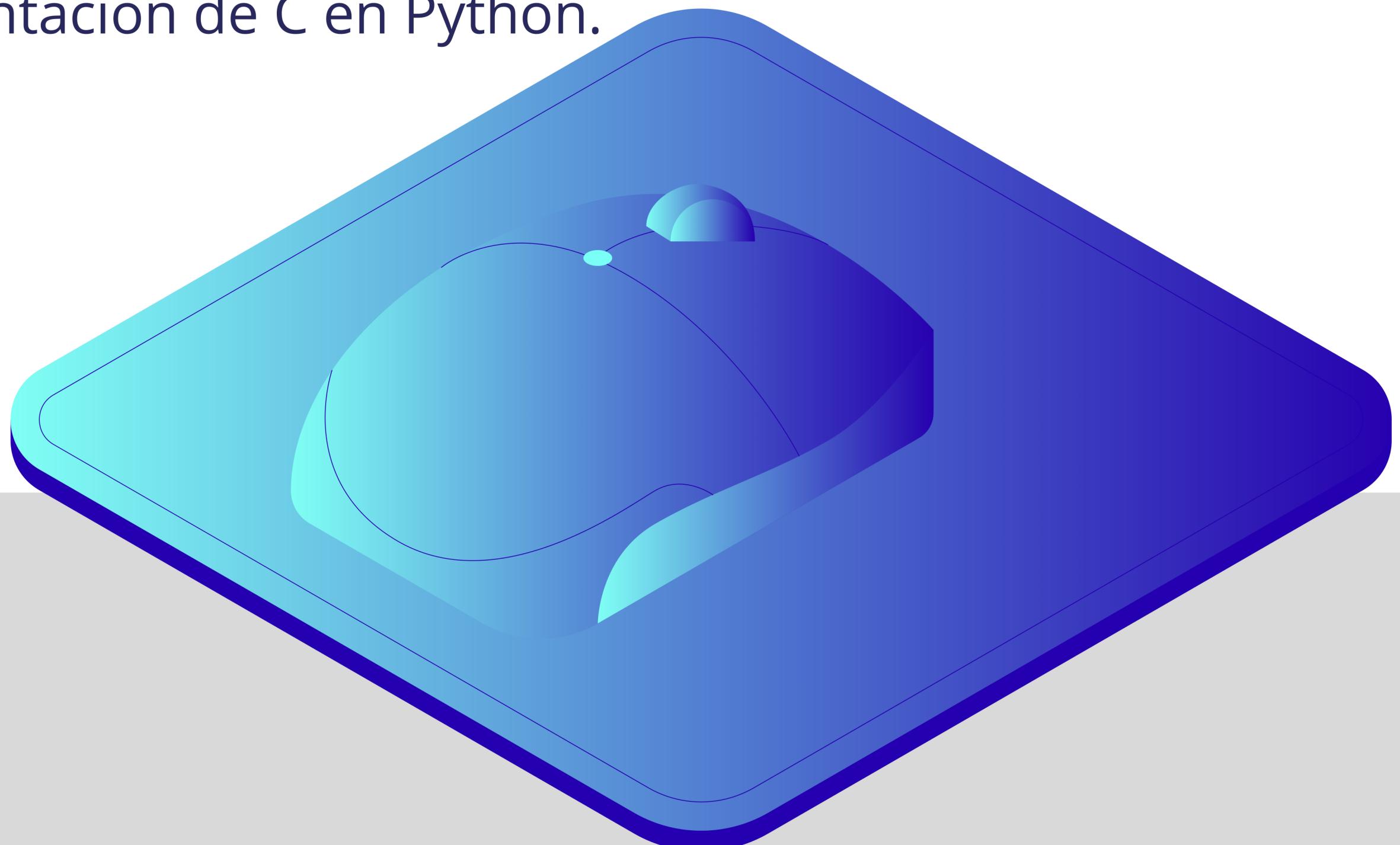
Primer Corte
Heidy Jazmin Leon Torres
Ingeniería Informatica
Cau Chiquinquirá



Estilos PEP 8 :

son las correctas formas para programar de C en la implementación de C en Python.

Primer Corte



Diseño del Código

Sangria

Utiliza siempre 4 espacios y no mezclar tabuladores y espacios ejemplo:

Si: # iteración 1

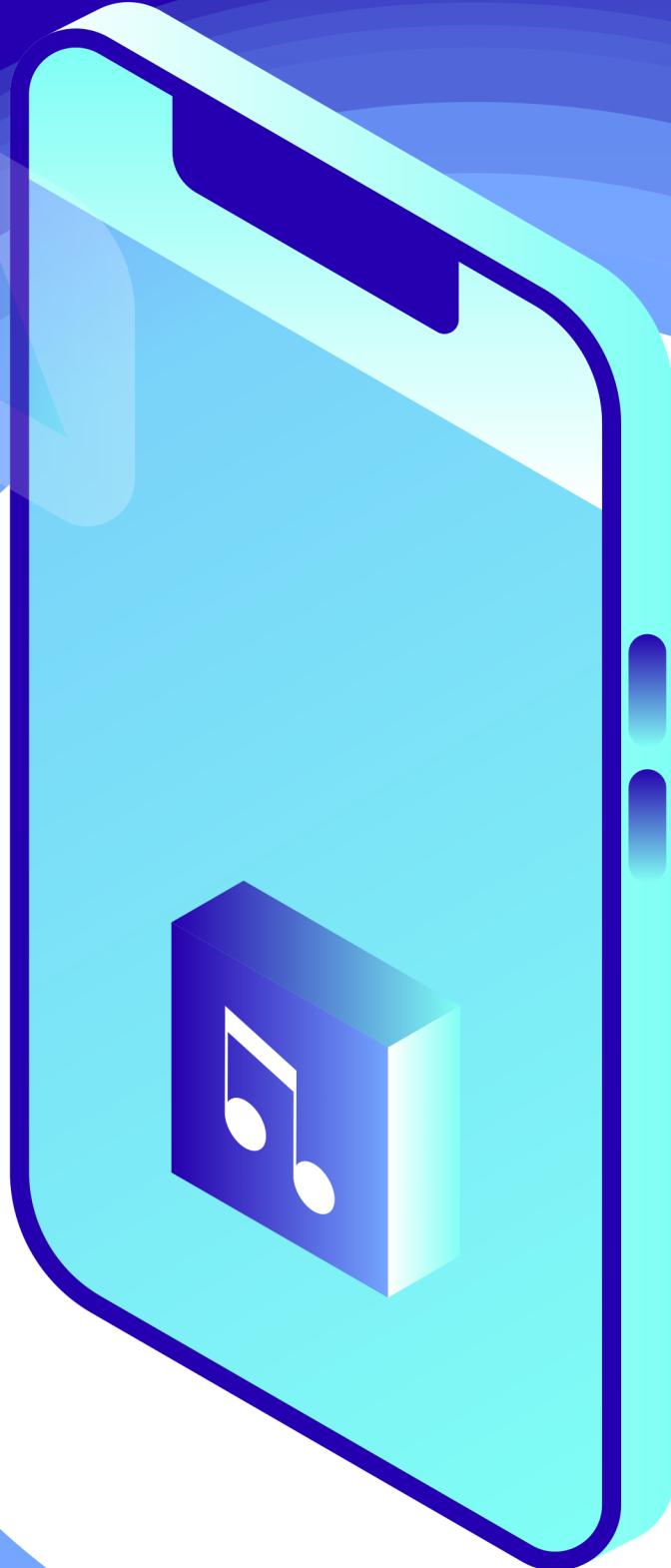
```
foo = funcion_que_crea_bar(variable_1, variable2  
                           variable_3)
```

iteración 2

```
foo = funcion_que_crea_bar(  
                           variable_1, variable2  
                           variable_3)
```

No:

```
foo = funcion_que_crea_bar(variable_1, variable2  
                           variable_3)
```





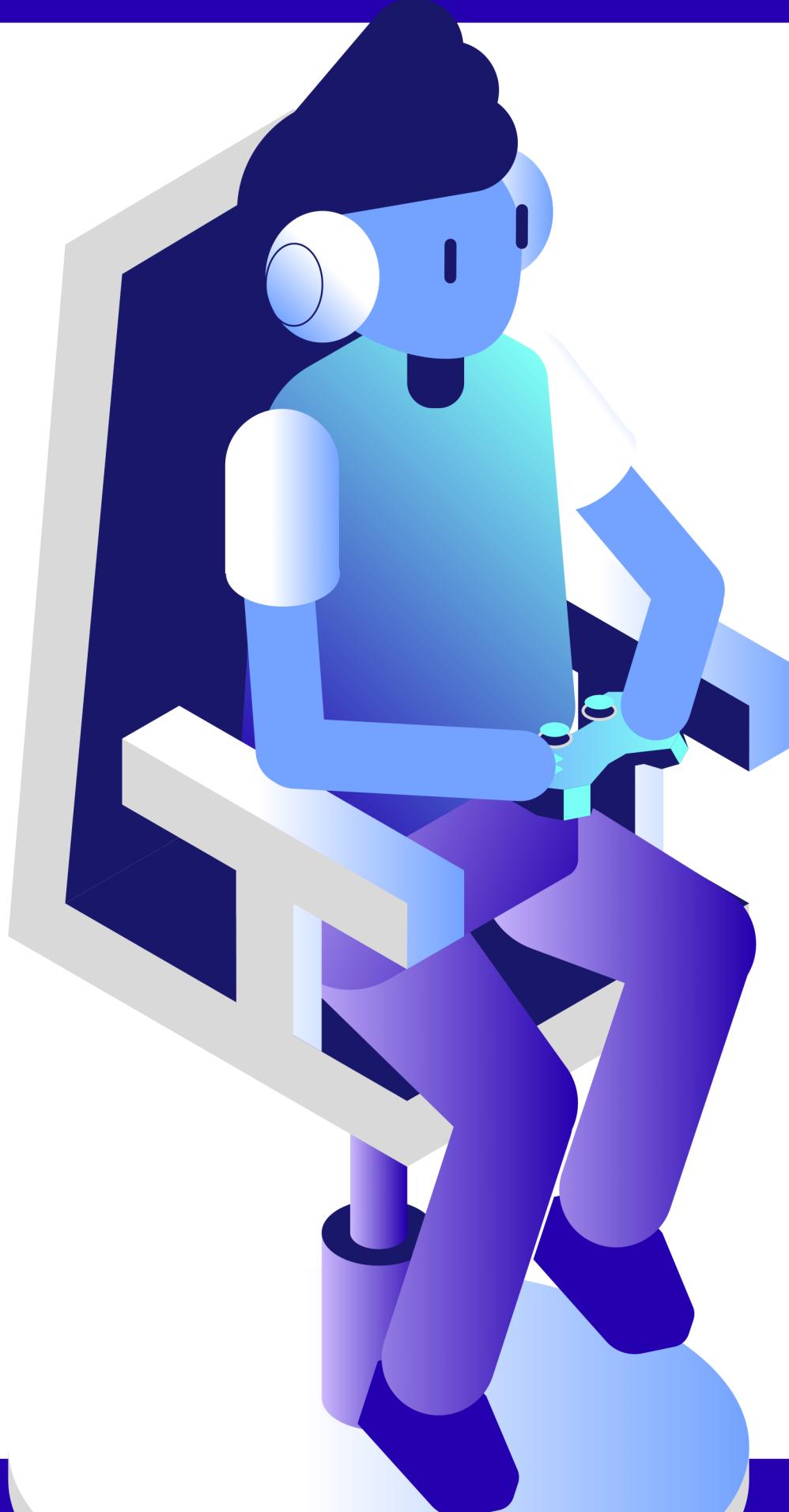
Tamaño maximo de lineas

Las líneas deben limitarse a un máximo de 79 caracteres, En el caso de largos bloques de texto (“docstrings” o comentarios), limitarlos a 72 caracteres es recomendado.

Lineas en Blanco

Separar las definiciones de las clases y funciones con dos líneas en blanco. Los métodos dentro de clases se separan con una línea en blanco.

Primer Corte



Imports

Los Imports de distintos módulos deben estar en líneas diferentes:

Si: import os
import sys

No: import os, sys

Si se pueden poner en una línea los elementos que se importan de un mismo modulo:

from subprocess import Popen, PIPE

estos deben ir al principio del fichero, después de los comentarios y de la documentación y antes de la definición de las variables globales y las constantes.

Espacios en blancos en expresiones

Evitar espacios en blancos extra en:

Si: spam(ham[1], {eggs: 2})

No: spam(ham[1], { eggs: 2 })

Si: if x == 4: print x, y; x, y = y, x No: if x == 4 : print x , y ; x , y = y , x

· Antes del paréntesis de una llamada a una función:

Si: spam(1)

No: spam (1)

· Antes del paréntesis de un índice:

Si: dict['key'] = list[index]

No: dict ['key'] = list [index]

· Siempre rodee los operadores binarios con un solo espacio en cada lado de la asignación al igual que los espacios de operadores aritméticos:

Si:

i = i + 1

submitted += 1

x = x * 2 - 1

hypot2 = x * x + y * y

c = (a + b) * (a - b)

No:

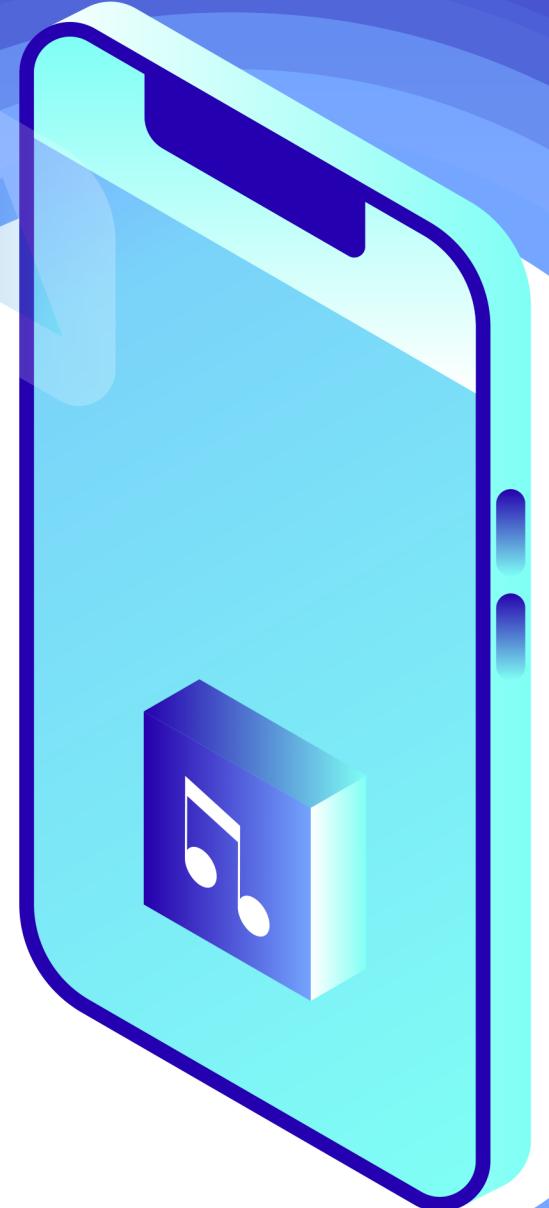
i=i+1

submitted +=1

x = x*2 - 1

hypot2 = x*x + y*y

c = (a+b) * (a-b)



Como utilizar comas finales

Las comas finales suelen ser opcionales, excepto que son obligatorias cuando se crea una tupla de un elemento

```
# Si:  
FILES = ('setup.cfg',)
```

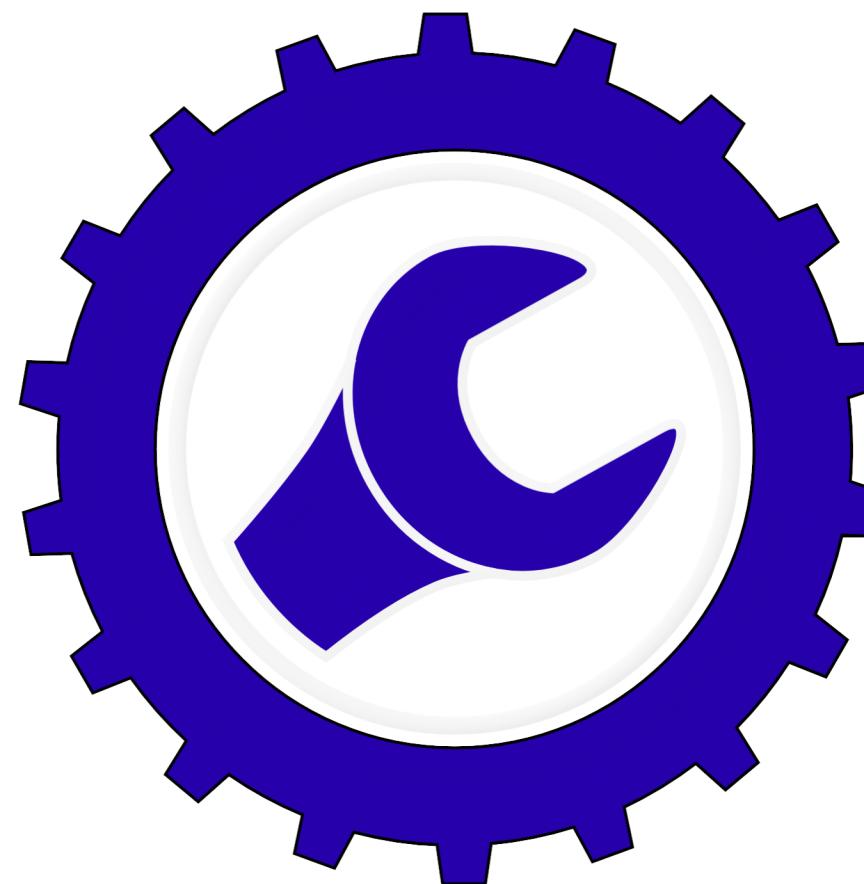
```
# No:  
FILES = 'setup.cfg',
```

Primer Corte

Descriptivo: estilos de nombre

Comúnmente se distinguen los siguientes estilos los siguientes estilos de nombre:

- b(una sola letra minúscula)
- B(una sola letra mayúscula)
- lowercase
- lower_case_with_underscores
- UPPERCASE
- UPPER_CASE_WITH_UNDERSCORES
- CapitalizedWords(o CapWords, o CamelCase, llamado así por el aspecto irregular de sus letras). Esto también se conoce a veces como StudlyCaps.
- mixedCase(¡se diferencia de CapitalizedWords por el carácter inicial en minúscula!)
- Capitalized_Words_With_Underscores(¡feo!)



Nombre de variables globales

Los nombres de las funciones deben estar en minúsculas, con las palabras separadas por guiones bajos según sea necesario para mejorar la legibilidad.

Los nombres de variables siguen la misma convención que los nombres de funciones.

Constantes

Las constantes generalmente se definen a nivel de módulo y se escriben en letras mayúsculas con guiones bajos que separan las palabras. Los ejemplos incluyen MAX_OVERFLOWy TOTAL. ●

Diseñar para la herencia

Decida siempre si los métodos y las variables de instancia de una clase (colectivamente: "atributos") deben ser públicos o no públicos. En caso de duda, elija no público; es más fácil hacerlo público más tarde que hacer que un atributo público no sea público.

Referencia:
PEP 8 – Style guide for Python code | peps.python.org. (s. f.). <https://peps.python.org/pep-0008/>