PEP 8

GUIA DE ESTILOS PARA PYTHON

```
# Correct:
FILES = [
    'setup.cfg',
    'tox.ini',
    ]
initialize(FILES,
           error=True,
           )
```

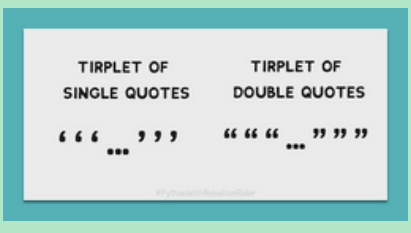
```
# Wrong:
FILES = ['setup.cfg', 'tox.ini',]
initialize(FILES, error=True,)
```

Este es el uso correcto e incorrecto de las comas para separar una lista de valores

UTF-8

El codigo debera usar UTF-8

Usa 4 espacios por cada nivel de identación



En Python, la cadena entre comillas simples y dobles es lo mismo.

A

Todas las lineas deberan tener un numero maximo de 74 caracteres

```
"""
Docstring
"""
# Comment
```

En Python, los comentarios en línea se hacen con "#" y un bloque de comentarios entre comillas simple o comillas dobles.

```
income = (gross_wages
          + taxable_interest
          + (dividends - qualified_dividends)
          - ira_deduction
          - student_loan_interest)
```

Utiliza un salto de linea antes de un operador binario

El codigo se debe hacer de manera que no ponga en desventaja ninguna implementación de Python (Cython, PyPy, Jython, etc)

\n

Utiliza 2 saltos de linea entre cada función

Estilos de nombrado:

- **b** (single lowercase letter)
- **B** (single uppercase letter)
- lowercase
- lower_case_with_underscores
- UPPERCASE
- UPPER_CASE_WITH_UNDERSCORES
- CapitalizedWords
- mixedCase

Estos deben nombrar estructuras distintas y no se deben combinar

```
def my_function(param1, param2, ...):
    pass
```

Los nombres de las **funciones** deben estar en **minúsculas**, con palabras separadas por guiones bajos según sea necesario

```
NAME = 'hemanth'
YOB = 1999
ID_NUM = 17783
print(NAME)
```

Las **constantes** generalmente se escriben en letras **mayúsculas** con guiones bajos que separan las palabras.

```
class DistanceConverter:
  kms_in_a_mile = 1.609
  def how_many_kms(self, miles):
    return miles * self.kms_in_a_mile
```

Utilice siempre "**self**" para el primer argumento de los métodos de instancia.

```
def compute_age_diff(cls,name, year):
  age= dt.now().year - year
  return cls(name, age)
```

Utilice siempre "**cls**" para el primer argumento de los métodos de clase.

```
class Car:
  def __init__(self, year, make, model):
    self.year = year
    self.make = make
    self.model = model
class DieselCar(Car):
  def __init__(self, year, make, model):
    super().__init__(year, make, model)
```

Los nombres de las clases normalmente deben usar la convención de **CapWords**.

```
# Correct:

if isinstance(obj, int):
```

```
# Wrong:

if type(obj) is type(1):
```

Las comparaciones de tipos de objetos siempre deben usar **isinstance ()**

```
# Correct:

if greeting:
```

```
# Wrong:

if greeting == True:
```

No compare los valores booleanos con Verdadero o Falso usando "=="

```
# Correct:

import os

import sys
```

Las importaciones generalmente deben estar en **líneas separadas**

```
# Correct:

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

```
# Wrong:

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

Evite extraños espacios en blanco

1 = l / I

0 = O

Nunca use los caracteres '**l**', '**O**' o '**I**' como nombres de variable de un solo carácter

Referencia

Van Rossum, G., Warsaw, B. y Coghlan, N. (32 de julio de 2013). PEP 8 -- Style Guide for Python Code. Python. Recuperado el 17 de septiembre de 2021https://www.python.org/dev/peps/pep-0008/#module-level-dunder-names