

# Manual de estilo

30 de octubre de 2015

Los programas son ejecutados por máquinas, pero han de ser leídos y mantenidos por humanos. Es necesario, por tanto, escribir programas que sean fáciles de leer y en los que sea sencillo entender la estructura y la lógica del mismo, es decir, saber qué hace y cómo lo hace.

A continuación damos una serie de pautas y convenciones que ayudan a la legibilidad y comprensión de los programas. Intentaremos seguir estas normas en todos los programas que aparezcan a lo largo de la asignatura.

## 1. Nombres y comentarios

- Por claridad, para aumentar la legibilidad y evitar problemas con caracteres especiales (como acentos), los nombres de funciones y variables estarán escritos en inglés.
- Los nombres de variables serán aclaratorios. Ejemplos de nombres claros son: `maximum`, `radius`, `area`, `speed`. Ejemplos de nombres confusos son: `a1`, `c`, `xxx`. Las variables índice usadas para recorrer bucles pueden tener nombres sencillos y similares a los que clásicamente se utilizan en matemáticas: `i`, `j`, `k`.
- En muchas ocasiones, al definir con precisión un identificador para funciones o variables utilizamos varias palabras. En estos casos, las palabras que componen el identificador se unirán con un guión bajo. Por ejemplo: `min_distance`, `shortest_path`, `validate_data`...
- Todo el texto que aparezca en el programa para explicar el mismo (comentarios, `docstring`, etc.), también estará escrito en inglés.

## 2. Funciones

- Todas las funciones deben estar documentadas en inglés explicando qué hacen, qué argumentos aceptan y qué devuelven. Este comentario, que se conoce como *docstring* debe cumplir el formato que muestra la figura 1.

Esta forma de comentar es muy importante, pues las diferentes herramientas que utilicemos para programar en python generan documentación a partir de la información contenida en el `docstring`. En Spyder, por ejemplo, la documentación de la función anterior se vería como aparece en la figura 2.

- Dentro del cuerpo de las funciones se usarán los parámetros de la función o variables localmente definidas, no se usarán variables globales.

## 3. Espaciado y separación

- Solo habrá una instrucción por línea
- Se importarán los módulos en líneas separadas. Por ejemplo:

```
1 def average(a, b):
2     """
3     Given two numbers a and b, return their average value.
4
5     Parameters
6     -----
7     a : number (int or float)
8         Firsts number
9     b : number (int or float)
10        Second number
11
12    Returns
13    -----
14    float
15        The average value of a and b
16
17    Example
18    -----
19    >>> average(5, 10)
20    7.5
21    """
22    return (a + b) / 2.0
```

Figura 1: Ejemplo de docstring para una función

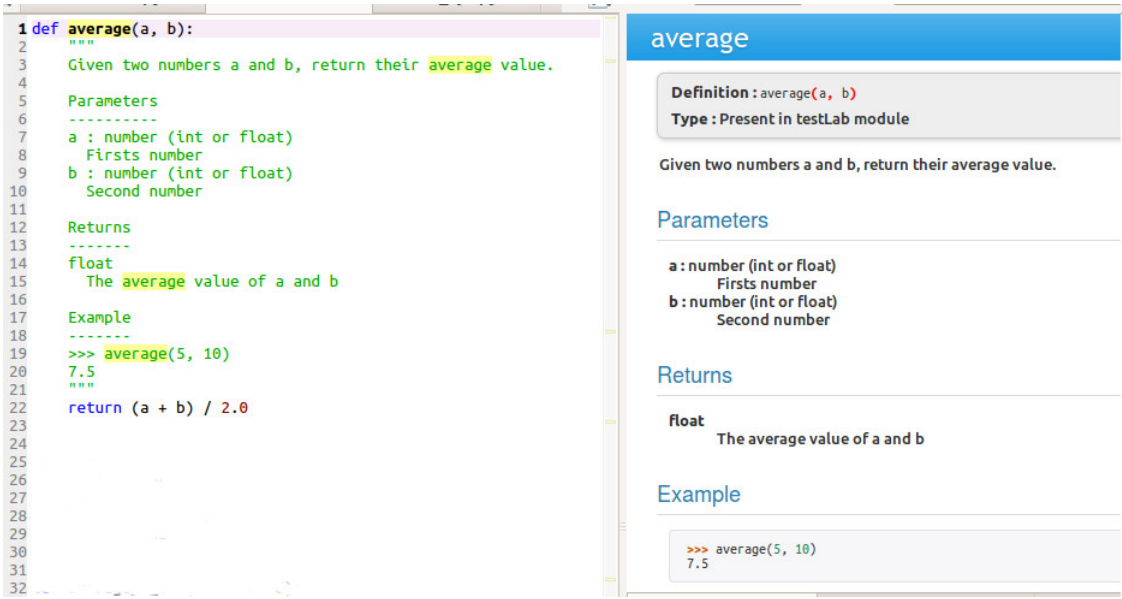


Figura 2: Ejemplo de visualización del docstring de una función en Spyder

```

1 import string
2 from PIL import Image

```

- Las líneas de texto no pueden ocupar más de 80 caracteres
- Habrá 2 líneas en blanco para separar funciones
- Como norma general, se dejará un espacio entre el operador y los operandos, incluido el operador de asignación. Ejemplos (el símbolo  representa un espacio):

```

1 pi= 3.14
2 area= side * side

```

Ejemplos de expresiones que no cumplen este estilo son:

```

1 pi=3.14
2 pi=3.14
3 area=side*side

```

En expresiones más complejas, es posible que juntar operadores y operandos, de forma uniforme y cuidadosa, mejore la legibilidad. Por ejemplo, la expresión

```

1 y=(a*f-c*e)/(-c*b+a*d)

```

puede ser más fácil de leer y entender que

```

1 y=(a*f-c*e)/(-c*b+a*d)

```

- Al invocar o definir funciones no se dejará ningún espacio al lado de los paréntesis. Los argumentos se separarán con una coma seguida de un espacio. Ejemplos:

```

1 def maximun(a,b):
2 maximum(3,45)

```

Ejemplos que no cumplen este estilo son:

```

1 def maximun(a,b):
2 def maximun(a,b):
3 def maximun(a,b):
4 maximum(3,45)
5 maximum(3,45)
6 maximum(3,45)
7 maximum(3,45)

```