Digital **NAO**

Documentación de Código

Instructivo

DOCUMENTACIÓN DE CÓDIGO
INSTRUCTIVO

Presentación

En este instructivo se ha compilado una serie de recomendaciones generales para que, al momento de programar en Python, Java o JavaScript, la documentación de código resulte funcional. Ya sea en el caso de que deba ser retomado para el desarrollo de una fase posterior del producto o solución para el cual fue creado, o en el escenario de que deba presentarse ante algún experto. Así se facilita el proceso de mejoras al ofrecer información clave sobre lo que se ha estado haciendo y su finalidad.

Objetivo

Como aprendedor, identificarás la importancia de la documentación de código y atenderás las recomendaciones específicas que se manejan para los lenguajes Python, Java y JavaScript, con el propósito de garantizar la claridad en la programación y de considerar un margen para realizar mejoras, cambios o adaptaciones.

Contenido

1. Importancia de la documentación de código

La importancia de la documentación de código radica en su función para facilitar la localización y comprensión de determinados códigos. Dado que en el uso de los lenguajes de programación intervienen diversos factores, tales como los distintos niveles de experiencia y los estilos de trabajo diferentes; los programadores suelen utilizar diversos lenguajes, lo mismo para problemas que para soluciones. Es ahí donde se aprecia la funcionalidad de la documentación de código, puesto que ayuda a terceros a ubicar, distinguir y procesar determinados códigos.

En casos en los que, por ejemplo, las soluciones son tan extensas y complejas que se dificulta llevar un registro de los detalles o se requiere del trabajo de un equipo de programadores, la documentación de código simplifica ubicar los errores, realizar las mejoras o implementar adaptaciones necesarias.

La documentación de código es también un elemento clave en aquellos escenarios en los que una persona se encarga del desarrollo original y alguien más retoma el código para otra etapa del proceso. Documentar desde el inicio aporta a quien deba realizar ajustes la claridad necesaria para comprender a qué se refiere cada código y definir los elementos a los que debe prestar mayor atención.

En este sentido, la documentación de código implica, para quien lo retoma o lo revisa, añadir comentarios que tienen como propósito solicitar explicaciones respecto a ciertos elementos para aclarar su funcionalidad a lo largo del desarrollo del código. Los comentarios que se añaden suelen responder a preguntas como las que se enlistan a continuación:

- ¿De qué se encarga una clase (módulo)?
- ¿Para qué sirve un método (función)?
- ¿Cuál es el uso esperado de un método (función)?, ¿cómo gestiona los errores?
- ¿Para qué se usa una variable?, ¿cuál es su uso esperado?
- ¿Qué algoritmo se está usando?, ¿de dónde se ha obtenido?
- ¿Qué limitaciones tiene tanto el algoritmo como su implementación?
- ¿Qué se debería mejorar si hubiera tiempo?

2. Elementos a considerar al documentar el código

a. Cabeceras de clase

Bloque de comentarios (o comentario) de varias líneas que dan a conocer el objetivo general de una clase. ¿Qué representa esta clase?, ¿cuál es el propósito común de los métodos en ella?

b. Cabeceras de método o función

Bloque de comentarios (o comentario) de varias líneas que describen la funcionalidad de un método o función y que, por lo general, especifican cuáles son los parámetros que se reciben y el valor que se regresa.

c. Líneas dentro de los métodos

Dentro de cada método o función es muy útil agregar comentarios de una línea que indican lo que se está haciendo. Esto servirá para no perder el hilo de lo que se espera en cierto punto del código y, con ello, averiguar dónde es necesario hacer algún cambio.

Entre las ventajas de emplear esta metodología se distinguen:

- Ahorrar tiempo de desarrollo a mediano y largo plazo.
- Agilizar el desarrollo de código
- Evitar duplicar el código
- Tener una modularidad en la programación
- Identificar con facilidad los errores
- Ahorrar costos

3. Recomendaciones para documentar

A continuación, se presentan las recomendaciones para realizar la documentación en los lenguajes de programación Python, Java y JavaScript.

a. Python

De acuerdo con la guía de estilo, **PEP 8** (Python Enhancement Proposal) promueve una codificación fácil de leer y visualmente agradable. Sus puntos más importantes son:

- Preferir espacios en vez de etiquetas
- Usar sangrías de cuatro espacios, no tabuladores
- Recortar las líneas para que no superen los 79 caracteres
- Usar líneas en blanco para separar funciones y clases, así como bloques grandes de código dentro de las funciones
- Poner comentarios en una sola línea

- Usar *docstrings* o cadena de documentos, es decir, una cadena de caracteres que se coloca como primer enunciado de un módulo, clase, método o función, cuyo propósito es explicar su intención
- Usar espacios alrededor de operadores y luego de las comas, pero no directamente dentro del paréntesis. Ejemplo: a = f(1, 2) + g(3, 4)
- Nombrar las clases y funciones consistentemente. La convención para clases es el uso del estilo de escritura Camello (*CamelCase*); para funciones y métodos, minúsculas con guiones bajos
- Declarar las funciones en minúscula y las palabras separadas por guiones bajos. Ejemplo: def_funcion_cool()
- Comenzar los métodos privados de una clase con doble guion bajo. Ejemplo: def __private_method()
- Comenzar los métodos protegidos de una clase con guion bajo. Ejemplo: def _protected_method()
- Usar en mayúsculas separadas por guiones bajos las constantes del módulo. Ejemplo: NUMERO_MAXIMO = 10
- Usar el *self* como primer parámetro con los métodos de instancia de una clase
- Usar *cls* como primer parámetro en los métodos de clase, para referirse a la misma clase
- Usar negación en línea (*if a is not b*) en vez de negar una expresión positiva (*if not a is b*)
- No validar valores vacíos usando *len if* (len(lista) == 0), sino usar *if not lista*
- Colocar siempre los *imports* al inicio del archivo
- Importar siempre funciones y clases usando *from my_module import MyClass* en vez de importar el módulo completo *import my_module*

b. Java

Javadoc es el estándar para documentar clases de *Java* y debe incluir:

- Nombre de la clase, descripción general, número de versión y nombre de autores.
- Documentación de cada constructor o método: nombre del constructor o método, tipo de retorno, nombres y tipos de parámetros (si los hay), descripción general, descripción de parámetros (si los hay) y descripción del valor que devuelve.

- Archivo *Léeme* o *Readme*, donde se incluyen por lo menos el título del proyecto, cómo arrancará, una descripción, versión, autores y las instrucciones para los usuarios.

Para que *Javadoc* sea capaz de generar documentación automáticamente, deben seguirse estas reglas:

- La documentación para *Javadoc* ha de incluirse entre símbolos de comentario que inician con una barra y doble asterisco, y terminan con un asterisco y barra simple.
- La ubicación le define a *Javadoc* qué representa el comentario: si está incluido justo antes de la declaración de clase, se considerará un comentario de clase; y si está incluido justo antes de la signatura de un constructor o método, se considerará un comentario de ese constructor o método.

Para alimentar *Javadoc* se usan ciertas palabras reservadas como *etiquetas* (o *tags*) precedidas por el carácter "@". Las principales etiquetas son:

Etiqueta	Descripción
@author	Nombre del desarrollador.
@deprecated	Indica que el método o clase es obsoleto (propio de versiones anteriores) y no se recomienda su uso.
@param	Definición de un parámetro de un método. Es requerido para todos los parámetros del método.
@return	Informa lo que devuelve el método; no se aplica en constructores o métodos *void*.
@see	Asocia con otro método o clase.
@version	Versión del método o clase.

Tabla 1. Etiquetas Javadoc.

c. **JavaScript**

De acuerdo al estándar JSDoc, las reglas para documentar en JavaScript son las siguientes:

- **Regla 1. El mejor comentario es el implícito o auto-documentador.**
  Es importante que los nombres de las variables, funciones, parámetros, etc. sean cortos, descriptivos y, principalmente, que sean términos en inglés por tratarse de un lenguaje universal en programación.
- **Regla 2. Usa un estilo de comentarios y no lo cambies, pero respeta el existente.**
  Si se trata de un proyecto nuevo, se debe usar un mismo estilo de comentarios a lo largo de su desarrollo; si se trata de un proyecto ya desarrollado, será necesario ajustarse al estilo de comentarios que se esté usando.
- **Regla 3. Usa comentarios multilínea en cabeceras de funciones y en línea en otros casos.**
  Si una línea es demasiado larga, se deberá dividir en varias líneas. La primera estará fija y el resto se desplazará hacia dentro respecto de la primera. Los comentarios puntuales en línea se harán con **//** (doble *slash*).
- **Regla 4. Usa un comentario al principio de un archivo *js* para describir su contenido.**
  Al principio de cada archivo se debe incluir un comentario descriptivo sobre qué es lo que contiene. Deberá comenzar con la etiqueta ***@fileoverview***.
- **Regla 5. Documenta las funciones, incluye los parámetros que usan y los que devuelven.**
  Se debe incluir un comentario típico para describir una función que integrará una descripción de lo que esta hace, de sus parámetros y de lo que devuelve la función.
  La descripción de un parámetro comienza con la etiqueta ***@param***. La descripción de lo que devuelve la función se hace comenzando con la etiqueta ***@return***. Si la función no devuelve nada se omitirá esta etiqueta.
- **Regla 6. Usa comentarios para constructores y para documentar las propiedades.**
  Cuando se defina un tipo de objeto habrá que usar la etiqueta ***@constructor*** como una declaración que indica que se está definiendo el constructor para un tipo de objeto. La etiqueta ***@type*** se usará para describir qué tipo de dato corresponde a la propiedad.
- **Regla 7. Usa etiquetas normalizadas.**

Al incluir la descripción de un elemento se usarán etiquetas normalizadas. Las más recomendadas son:

Etiqueta	Descripción
@author	Indica el nombre del autor del código.
@const {type}	Señala que una propiedad o variable serán constantes y su tipo.
@constructor	Precisa que se define un tipo de objeto.
@deprecated	Muestra que una función, método o propiedad no deberían usarse por ser obsoletas. Debe indicarse, entonces, cuál es la función, método o propiedad que debe emplearse en su lugar.
@enum	Indica que se trata de una enumeración.
@extends	Usado con *@constructor*, precisa que un tipo de objeto hereda algo de otro tipo de objeto.
@fileoverview	Comentario para describir los contenidos de un fichero.
@interface	Señala que una función define una interfaz.
@implements	Usado con *@constructor*, indica que un tipo de objeto implementa una interfaz.
@license	Muestra los términos de licencia.
@override	Indica que un método o propiedad de una subclase sobreescribe el método de la superclase. Si no se indica otra cosa, los comentarios y la documentación serán los mismos que los de la superclase.
@param {type}	Indica un parámetro para un método, función o constructor y el tipo de dato que es.
@return {type}	Indica qué devuelve un método o función y su tipo.
@see	Comunica una referencia a documentación más amplia.
@supported	Indica navegadores para los que se conoce que aceptan el código.
@type {Type}	Identifica el tipo de una variable, propiedad o expresión.
Otros	Existen más etiquetas, sin embargo, en esta experiencia de aprendizaje se centrará en las que aquí se refieren.

Tabla 2. Etiquetas normalizadas.

4. Recomendaciones generales

Finalmente, considera las siguientes indicaciones que tienen como propósito contribuir a un mejor desempeño en la documentación de código:

- Opta por códigos claros y descriptivos; de esa manera, se generará una menor cantidad de comentarios.
- Añade comentarios al principio de cada clase, método, variable y de códigos no evidentes.
- Genera solo la documentación necesaria, realmente valiosa e imprescindible.
- Revisa y actualiza comentarios cuando debas mejorar un código.
- Procura que tus comentarios sean claros, sencillos y descriptivos.
- Documenta con consistencia lo que implementes al desarrollar un código.

Referencias

APR. (2019). *Guía de estilo JavaScript: comentarios proyectos*. https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=881:guia-de-estilo-JavaScript-comentarios-proyectos-jsdoc-param-return-extends-ejemplos-cu01192e&catid=78&Itemid=206

Python Software Foundation. (2022). *4. Más herramientas para control de flujo*. https://docs.python.org/es/3.8/tutorial/controlflow.html#documentation-strings

Rodríguez, A. (2019). *Documentar proyectos Java con Javadoc. Comentarios, símbolos, tags*. https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=646:documentar-proyectos-java-con-javadoc-comentarios-simbolos-tags-deprecated-param-etc-cu00680b&catid=68&Itemid=188