Comentarios en Python

(Data Engineer)

Introducción

Bienvenidos al módulo 6 - Comentarios en Python. En el mismo, los objetivos de aprendizaje estarán orientados a incorporar conocimientos teóricos y buenas prácticas para comentar nuestro código.

Una vez finalizado este módulo estarás capacitado para las siguientes acciones:

- Conocer la importancia de comentar nuestro código.
- Utilizar diferentes alternativas para realizar comentarios.
- Utilizar atajos del teclado para realizar comentarios.
- Conocer buenas y malas prácticas.

Tema 1. Comentarios

Objetivos

Una vez finalizado este tema estarás capacitado para las siguientes acciones:

- Conocer acerca de la importancia de comentar el código de un programa y tanto para quien lo escribe como para quien lo lee.
- Comentar tu código Python utilizando diferentes opciones.
- Utilizar atajos para hacer tus comentarios más rápidos.

Introducción

Al escribir nuestros programas en Python, es importante asegurarse de que otros puedan entender fácilmente el código . Dar nombres obvios a las variables, definir funciones explícitas y organizar el código son excelentes maneras de hacer esto.

¡ Otra forma asombrosa y fácil de aumentar la legibilidad de nuestro código es mediante el uso de **comentarios** !

Por qué es importante comentar el código

Los comentarios son una parte integral de cualquier programa. Pueden venir en forma de cadenas de documentación a nivel de módulo, o incluso explicaciones en línea que ayudan a arrojar luz sobre una función compleja.

En el siguiente ejemplo podemos ver cómo comentar una función

```
def reload(self):
    """
    Reloads the GeoServer catalog and configuration from disk.

    This operation is used in cases where an external tool has modified the on-disk configuration.
    This operation will also force GeoServer to drop any internal caches and reconnect to all data stores.
    curl -X POST http://localhost:8080/geoserver/rest/reload -H  "accept: application/json" -H  "content-type: application/json"
    """
    try:
        url = "{}/rest/reload".format(self.service_url)
        r = requests.post(url, auth=(self.username, self.password))
        return "Status code: {}".format(r.status_code)

    except Exception as e:
        return "reload error: {}".format(e)
```

Antes de profundizar en los diferentes tipos de comentarios, vamos a ver más de cerca por qué es tan importante comentar el código.

Considere los siguientes dos escenarios en los que un programador decidió no comentar su código.

Escenario 1: Al leer nuestro propio código

Supongamos que un cliente A quiere una implementación de última hora para su servicio web. El plazo de implementación es ajustado, por lo que decidimos simplemente hacer que funcione. Todo ese material "extra" (documentación, comentarios apropiados, etc.) lo agregaremos más adelante.

Llega la fecha límite e implementamos el servicio justo a tiempo.

Hacemos una nota mental para volver y actualizar los comentarios, pero antes de que podamos ponerlo en nuestra lista de tareas pendientes, llega nuestro jefe con un nuevo proyecto que debemos comenzar de inmediato. En unos pocos días, nos olvidamos por completo de que debíamos regresar y comentar correctamente el código que escribimos para el Cliente A.

Ahora avanzamos rápido en el tiempo y seis meses después, nos llega un requerimiento del cliente A para agregar una nueva funcionalidad al servicio web que implementamos anteriormente. Es nuestro trabajo mantenerlo y generar el parche con esa funcionalidad.

Entonces... Abrimos el editor de y... **¡¿Qué es lo que escribimos?!**

Así pasamos horas analizando nuestro antiguo código. En ese momento estábamos tan apurados en ese momento que no nombramos correctamente las variables ni configuramos las funciones en el flujo de control adecuado. Lo peor de todo es que no tenemos ningún comentario que nos indique cómo funciona nuestro programa

Los desarrolladores olvidan lo que hace su propio código todo el tiempo, especialmente si fue escrito hace mucho tiempo o bajo mucha presión. Cuando se acerca rápidamente una fecha límite, esa presión puede reflejarse en forma de código que es más desordenado de lo habitual.

Una vez que se envía el proyecto, muchos desarrolladores simplemente están demasiado cansados para volver atrás y comentar su código. Cuando llega el momento de revisarlo más adelante, pueden pasar horas tratando de analizar lo que escribieron.

Escribir comentarios sobre la marcha es una excelente manera de evitar que suceda el escenario anterior.

Escenario 2: Cuando otros están leyendo nuestro código

Imagina que sos el único desarrollador que trabaja en un pequeño proyecto de Django . Entiendes tu propio código bastante bien, por lo que no sueles usar comentarios ni ningún otro tipo de documentación, y te gusta que sea así. Los comentarios toman tiempo para escribir y mantener, y simplemente no lo ves necesario.

El único problema es que, para fin de año, tu “pequeño proyecto Django” se ha convertido en un proyecto de “20.000 líneas de código”, y tu supervisor está contratando desarrolladores adicionales para ayudar a mantenerlo.

Los nuevos desarrolladores trabajan duro para ponerse al día rápidamente, pero en los primeros días de trabajar juntos, te das cuenta de que están teniendo algunos problemas. Usaste algunos nombres de variables extravagantes y escribiste con una sintaxis súper concisa. Por lo tanto pasan mucho tiempo revisando tu código línea por línea, tratando de descubrir cómo funciona todo.

De esa manera, estarán listos para ayudarte a mantener el código luego de algunos días.

El uso de comentarios en todo el código puede ayudar a otros desarrolladores en situaciones como esta para comprender cómo funciona todo muy rápidamente.

Cómo escribir comentarios en Python

Ahora que comprendimos por qué es tan importante comentar nuestro código, vamos a repasar algunos conceptos básicos para que sepamos hacerlo correctamente.

Conceptos básicos de comentarios de Python

Los comentarios son para los desarrolladores. Describen partes del código para facilitar la comprensión de los programadores, incluidos nosotros mismos

Para escribir un comentario en Python, simplemente tendremos que colocar el símbolo hash **#** antes del comentario

```
In [1]: # Esto es un comentario
```

Python ignora todo después de la marca hash y hasta el final de la línea. Podemos insertarlos en cualquier parte de su código, incluso en línea con otro código.

```
for car in cars: #Imprimir autos
    print(car)
```

Así cuando ejecutemos el código, solo se procesa el for loop.

Los comentarios deben ser breves, y directos. Si bien PEP8 recomienda mantener el código en 79 caracteres o menos por línea, sugiere un máximo de 72 caracteres para comentarios en línea y cadenas de documentación. Por lo que si nuestro comentario se acerca o supera esa longitud, intentará distribuirlo en varias líneas.

Comentarios multilínea de Python

Desafortunadamente, Python no tiene una forma de escribir comentarios de varias líneas como en lenguajes como C, Javascript y Go.

```
# No podemos
hacer esto
en Python
```

```
/* Si podemos
hacer esto
en Javascript */
```

En el caso de Javascript, todo lo que se encuentre entre los símbolos /* */ será ignorado por el programa.

Si bien Python no tiene la funcionalidad nativa de comentarios de varias líneas, podemos crear comentarios de varias líneas en Python. Hay dos opciones sencillas de hacerlo.

Opción 1:

Consiste simplemente en presionar la tecla enter después de cada línea y agregar un nuevo hash.

```
def obtener_temperatura():
    # Esta función obtiene la temperatura
    # de la cámara frigorífica
    # y la imprime en la consola
```

Opción 2:

Consiste en utilizar cadenas de varias líneas para envolver el comentario dentro de un conjunto de comillas triples.

```
In [4]: """
        Podemos utilizar esta forma, en vez de
        agregar un hash # por cada línea
        de comentario
        """
Out[4]: '\nPodemos utilizar esta forma, en vez de\nagregar un hash # por cada línea\nde comentario\n'
```

Si bien esto nos permite comentar varias líneas, técnicamente no es un comentario. Es una cadena que no está asignada a ninguna variable, por lo que nuestro programa no la llama ni hace referencia a ella. Aún así, dado que se ignorará en el tiempo de ejecución y no aparecerá en el código de bytes, puede actuar efectivamente como un comentario.

Sin embargo, tenemos que tener cuidado donde colocamos estos "comentarios" de varias líneas. Dependiendo de dónde se encuentren, podrían convertirse en docstrings , que son piezas de documentación que están asociadas con una función o método.

Atajos en Python

A veces puede ser tedioso escribir comentarios en diferentes partes de nuestro programa, por lo que podemos emplear los siguientes atajos para escribir nuestro código más rápido.

Utilizar varios cursores

Resulta de mucha utilidad cuando tenemos que escribir el mismo comentario en diferentes partes de nuestro programa.

Para activar varios cursores tenemos que apretar la tecla Ctrl y luego hacer clic izquierdo en las ubicaciones donde necesitamos agregar los cursores.

Finalmente, cuando comencemos a escribir, veremos que el texto aparece en todos los cursores al mismo tiempo.

Para desactivarlo tenemos que hacer clic en la tecla Esc.

```
def funcion_1:
    # Comentarios

def funcion_1:
    # Comentarios

def funcion_1:
    # Comentarios
```

Comentar varias líneas simultáneamente

Cuando estamos escribiendo código posiblemente necesitemos quitar una parte del programa para hacer alguna determinada prueba y luego volverlo a agregar.

Comentar las líneas de forma simultánea nos puede ser de utilidad en esos casos. Para realizarlo debemos seleccionar todas las líneas a comentar y apretar **Ctrl /**

Para descomentar las líneas procedemos de igual forma.

```
# def funcion_1:
#     # Comentarios

# def funcion_1:
#     # Comentarios

# def funcion_1:
#     # Comentarios
```

Tema 2. Prácticas recomendadas

Objetivos

Una vez finalizado este tema estarás capacitado para las siguientes acciones:

- Conocer sobre buenas prácticas y malas prácticas a la hora de escribir comentarios en Python.
- Comentar tu código Python utilizando diferentes opciones.
- Utilizar atajos para hacer tus comentarios más rápidos.

Mejores prácticas de comentarios en Python

Si bien es bueno saber cómo escribir comentarios en Python, es igualmente vital asegurarse de que los comentarios sean legibles y fáciles de entender.

A continuación veremos algunas buenas prácticas que podemos tener en cuenta:

Al escribir código para nosotros mismos mismos

Podemos facilitarnos la vida comentando nuestro propio código correctamente. Incluso si nadie más lo verá, nosotros lo veremos, y esa es razón suficiente para hacerlo bien. Después de todo, nosotros somos desarrolladores, por lo que el código también debería ser fácil de entender para nosotros.

Una forma extremadamente útil de usar los comentarios consiste en utilizarlos para delinear un pseudocódigo dentro de una función.

```python
def get_top_cities(precio):
    top_cities = defaultdict(int)

    # Para cada cambio de precio
        # Obtener la ciudad para ese precio
        # Contar la cantidad de búsquedas que tuvo esa cidudad
        # Agregar al diccionario

    return dict(top_cities)
```

Así una vez que sepamos lo que debe hacer una función, podremos concentrarnos en traducirlo a código.

Usar comentarios nos puede ayudar a mantener todo en orden y a medida que avanzamos en el programa, sabremos lo que queda por hacer para tener un script completamente funcional. Después de "traducir" los comentarios al código, debemos eliminar cualquier comentario que se haya vuelto redundante para que se mantenga nítido y limpio.

para usted mismo es como un esquema para su código. Si no está seguro de cómo resultará su programa, puede usar los comentarios como una forma de realizar un seguimiento de lo que queda por hacer, o incluso como una forma de realizar un seguimiento del flujo de alto nivel de su programa. Por ejemplo, use comentarios para delinear una función en pseudocódigo:

Al escribir código para otros

A las personas les gusta hojear y saltar de un lado a otro del texto, y cuando leen código no es diferente. La única vez que probablemente tengamos que leer el código línea por línea es cuando no funciona y necesitemos averiguar qué está pasando.

En el resto de los casos, echaremos un vistazo rápido a las definiciones de variables y funciones para entender la esencia. Tener comentarios para explicar lo que sucede en un lenguaje sencillo realmente puede ayudar a un desarrollador en esta posición.

Es importante ser amable con nuestros compañeros desarrolladores y usar comentarios para ayudarlos a hojear nuestro código. Los

comentarios en línea deben usarse con moderación para aclarar fragmentos de código que no son obvios por sí mismos. (Por supuesto, nuestra primera prioridad debe ser que nuestro código se sostenga por sí mismo, pero los comentarios en línea pueden ser útiles en este sentido).

Por ejemplo si tenemos una función compleja, podemos incluir un breve comentario para arrojar algo de luz:

```
def funcion_compleja():
    # Esta función es algo compleja
```

Peores prácticas de comentarios en Python

Así como existen estándares para escribir comentarios de Python, existen algunos tipos de comentarios que no conducen a un código limpio. A continuación vemos algunos de ellos que debemos evitar.

Comentarios obvios

Esto significa que nuestro código debe tener poca o ninguna redundancia. No es necesario comentar un fragmento de código que se explique lo suficiente, como por ejemplo este:

```
return a # retorna a
```

Podemos ver claramente que la variable a se retorna, por lo que no es necesario indicarlo explícitamente en un comentario.

Los comentarios WET pueden ser un simple error, especialmente si usó comentarios para planificar su código antes de escribirlo. Pero una vez que tenga el código funcionando bien, asegúrese de regresar y eliminar los comentarios que se hayan vuelto innecesarios.

Comentarios de relleno

Son aquellos intentan enmascarar los problemas subyacentes de un programa, y los comentarios son una forma de tratar de ocultar esos problemas. Los comentarios deben respaldar nuestro código, no tratar de explicarlo. Si nuestro código está mal escrito, ninguna cantidad de comentarios lo arreglará.

Tomemos este ejemplo simple:

```
# Un diccionario de familias que viven en una ciudad
mydict = {
    "Midtown": ["Gilmore", "Young","Brantley"],
    "Norcross": ["Montgomery"],
    "Ackworth": []
}

def a(dict):
    # Para cada ciudad
    for p in dict:
        # Si no hay familias en esa ciudad
        if not mydict[p]:
            # Decimos que no hay familias
            print("Ninguna.")
```

Este código es bastante ineficiente. Hay un comentario antes de cada línea que explica lo que hace el código.

Este script podría haberse simplificado asignando nombres obvios a variables, funciones y diccionarios, así:

```
families_by_city = {
    "Midtown": ["Gilmore", "Young","Brantley"],
    "Norcross": ["Montgomery"],
    "Ackworth": [],
}

def no_families(cities):
    for city in cities:
        if not families_by_city[city]:
            print(f"No hay familias en: {city}.")
```

Al usar convenciones de nomenclatura obvias, pudimos eliminar todos los comentarios innecesarios y también reducir la longitud del código.

Nuestros comentarios rara vez deberían ser más largos que el código que admiten. Si estamos dedicando demasiado tiempo a explicar lo que se hizo, entonces debemos volver atrás y refactorizar para que nuestro código sea más claro y conciso.

Comentarios groseros

Esto es algo que probablemente surja cuando se trabaja en un equipo de desarrollo. Cuando varias personas están trabajando en el mismo código, otras entrarán y revisarán lo que hemos escrito y realizarán cambios. De vez en cuando, es posible que nos encontremos con alguien que se atrevió a escribir un comentario como este:

```
# Aquí reparamos el desastre que hizo Manuel
```

De esta manera podemos ofender a un compañero y además nunca se sabe, quizás esta versión podría enviarse a producción y llegar a los ojos de un cliente.

Cómo practicar los comentarios

¡La forma más sencilla de comenzar a escribir más comentarios es simplemente hacerlo!

Podemos comenzar escribiendo comentarios para nosotros mismos en nuestro propio código. Debemos asegurarnos de incluir comentarios simples de ahora en adelante cuando sea necesario.

Agregar un poco de claridad a las funciones complejas y colocar una cadena de documentación en la parte superior de todos los scripts, también es un buen inicio.

Otra buena manera de practicar es volver atrás y revisar el código antiguo que hemos escrito. Donde veamos que algo podría no tener

sentido y debemos limpiar el código. Si aún necesitamos soporte adicional, podemos agregar un comentario rápido para ayudar a aclarar el propósito del código.

Esta es una idea especialmente buena si nuestro código está en GitHub y las personas están haciendo Fork a nuestro repositorio.

También podemos retribuir a la comunidad comentando el código de otras personas. Si descargamos algo de GitHub y tuvimos problemas para revisarlo, podemos agregar comentarios a medida que comprendamos lo que hace cada parte del código.

"Firma" tu comentario con tus iniciales y la fecha, y luego envía los cambios como una solicitud de extracción. Si tus cambios se fusionan, podrías estar ayudando a docenas, si no a cientos, de desarrolladores como a obtener una ventaja en su próximo proyecto

Cierre

Durante esta unidad aprendimos a comentar nuestro código Python.

Comenzamos desarrollando las razones por las cuales es importante comentar nuestro código y aprendimos diferentes maneras de realizarlo y algunos atajos para hacerlo más sencillo

Luego revisamos algunas buenas prácticas y malas prácticas cuando trabajamos con comentarios.

Finalmente incorporamos algunos tips para comenzar a desarrollar el hábito de comentar nuestro código.

Quedamos a disposición de las consultas que puedan surgir.

Referencias

Comments in Python: Why are They Important And How to Use Them | Simply Learn. Recuperado de: https://www.simplilearn.com/tutorials/python-tutorial/comments-in-python

Writing Comments in Python | Real Python. Recuperado de: https://realpython.com/python-comments-guide/

Python Comments | Geeks for Geeks. Recuperado de: https://www.geeksforgeeks.org/python-comments/

tiimiit by