

Comentarios

Edison Achalma

Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga

Resumen

Este abstract será actualizado una vez que se complete el contenido final del artículo.

Tabla de contenidos

Introduction	2
0.1 Motivación	2
0.2 Intuición	2
0.3 Modelo mental	2
0.4 Desarrollo	2
0.4.1 1. Comentario de una línea	2
0.4.2 2. Comentarios al final de línea	2
0.4.3 3. Desactivar código con comentarios	2
0.4.4 4. Comentarios de múltiples líneas	3
0.4.5 5. Comentarios en bloque y documentación	3
0.5 Ejemplos	3
0.6 Errores comunes	4
0.7 Buenas prácticas	4
0.8 Ejercicio	5
0.9 Síntesis	6
0.10 Qué sigue	6
1 Publicaciones Similares	6

Edison Achalma  <https://orcid.org/0000-0001-6996-3364>

El autor no tiene conflictos de interés que revelar. Los roles de autor se clasificaron utilizando la taxonomía de roles de colaborador (CRediT; <https://credit.niso.org/>) de la siguiente manera: Edison Achalma: conceptualización, metodología, análisis formal, investigación, recursos, curación de datos, redacción, visualización, supervisión, administración del proyecto

La correspondencia relativa a este artículo debe dirigirse a Edison Achalma, Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga, Portal Independencia N° 57, Ayacucho, AYA 5001, Perú, Email: elmer.achalma.09@unsch.edu.pe

Comentarios

0.1 Motivación

El código se escribe una vez, pero se lee muchas veces. Los comentarios son la forma en que te comunicas con otros programadores (y contigo mismo en el futuro). Un buen comentario explica el “por qué” detrás del “qué”, haciendo que tu código sea más fácil de entender, mantener y depurar. Además, los comentarios son una herramienta poderosa para probar código temporalmente sin eliminarlo.

0.2 Intuición

Imagina que estás escribiendo un libro de cocina. Las instrucciones (el código) son los pasos de la receta. Los comentarios son como notas al margen: “este paso es importante porque evita que se queme”, “aquí puedes sustituir el ingrediente si no lo tienes”. El chef (Python) ignora completamente tus notas al margen y solo sigue las instrucciones principales.

0.3 Modelo mental

Los comentarios en Python:

1. **Comienzan con #:** todo lo que sigue en esa línea es ignorado por Python.
2. **Pueden estar al inicio de la línea o al final** de una línea de código.
3. **No tienen efecto en la ejecución:** son solo para humanos.
4. **No existe sintaxis oficial para comentarios de múltiples líneas**, pero hay formas de simularlos.

0.4 Desarrollo

0.4.1 1. Comentario de una línea

El comentario comienza con # y Python ignora todo lo que sigue hasta el final de la línea:

```
# Esto es un comentario
print("Hola mundo") # Esto también es un comentario
```

0.4.2 2. Comentarios al final de línea

Útiles para explicar una línea específica:

```
precio = 100      # Precio base sin impuestos
iva = 0.21        # IVA del 21%
total = precio * (1 + iva) # Cálculo del precio final
```

0.4.3 3. Desactivar código con comentarios

Comentar una línea evita que se ejecute:

```
# print("Esta línea no se ejecuta")
print("Esta línea sí se ejecuta")
# Salida: Esta línea sí se ejecuta
```

Útil para depuración: puedes desactivar temporalmente partes del código.

0.4.4 4. Comentarios de múltiples líneas

Python no tiene un símbolo especial para comentarios de varias líneas, pero puedes:

Opción 1: Usar # en cada línea

```
# Esto es un comentario
# que ocupa varias líneas
# y explica algo complejo
print("Hola")
```

Opción 2: Usar cadenas de texto con triple comilla (no asignadas)

```
"""
Este es un comentario
que ocupa varias líneas
Python lo ignora porque no está asignado a una variable
"""
print("Hola")
```

Importante: técnicamente esto es una cadena de texto (string), no un comentario. Aunque Python la ignora, la buena práctica es usar # para comentarios oficiales.

0.4.5 5. Comentarios en bloque y documentación

- **Comentarios de bloque:** explican secciones enteras de código.
- **Docstrings:** comentarios especiales para documentar funciones y clases (usando """ y asignados a la función).

```
def calcular_area(base, altura):
    """
    Calcula el área de un rectángulo.

    Args:
        base (float): La base del rectángulo
        altura (float): La altura del rectángulo

    Returns:
        float: El área calculada
    """
    return base * altura
```

0.5 Ejemplos

Mínimo — Comentarios simples:

```
# Saludo inicial
print("Hola") # Muestra un saludo
```

Progresivo — Comentarios para explicar lógica:

```
# Cálculo de descuento
precio_original = 120
descuento = 0.15 # 15% de descuento

# El precio final es el original menos el descuento
precio_final = precio_original - (precio_original * descuento)
print(precio_final) # Muestra: 102.0
```

Real — Documentación en un script de configuración:

```
# =====
# CONFIGURACIÓN DEL SISTEMA
# =====

# Configuración de la base de datos
# Usamos SQLite para desarrollo local
DB_HOST = "localhost"
DB_PORT = 5432

# Límites del sistema
MAX_USERS = 100 # Máximo de usuarios simultáneos
TIMEOUT = 30 # Tiempo de espera en segundos

# DEBUG = True # Descomentar para modo depuración
DEBUG = False # Modo producción
```

0.6 Errores comunes

Error	Síntoma	Cómo detectarlo
Olvidar #	El texto se interpreta como código y da error	Asegúrate de que todo comentario comienza con #
Espacio después de #	No da error, pero es una convención	Pon un espacio después de # para legibilidad: # Esto es un comentario
Confundir comentario con string	"Esto no es un comentario" es código válido	Los comentarios usan #, no comillas
Anidar comentarios	No se puede anidar # dentro de #	Cada línea es independiente

0.7 Buenas prácticas

- Comenta el “por qué”, no el “qué”: el código ya dice qué hace.

```
# Poco útil
x = x + 1 # Incrementa x en 1

# Útil
x = x + 1 # Contador de intentos fallidos
```

- **Mantén los comentarios actualizados:** un comentario desactualizado es peor que ninguno.
- **Usa comentarios de bloque** para secciones largas:

```
# =====  
# SECCIÓN: VALIDACIÓN DE DATOS  
# =====
```

- **Documenta funciones con docstrings** (triple comilla):

```
def saludar(nombre):  
    """Saluda al usuario por su nombre."""  
    print(f"Hola, {nombre}")
```

- **No abusos de los comentarios:** el código limpio y autoexplicativo es mejor que el código lleno de comentarios.

0.8 Ejercicio

Básico — Agrega comentarios apropiados

El siguiente código funciona pero no tiene comentarios. Agrega comentarios que:

1. Expliquen qué hace cada sección
2. Documenten qué hace la función
3. Desactiven temporalmente una línea de código (sin eliminarla)

```
def calcular_total(precio, cantidad):  
    return precio * cantidad  
  
precio_producto = 25.50  
unidades = 4  
total = calcular_total(precio_producto, unidades)  
print(total)  
print("Fin del cálculo")
```

Solución (no mires hasta haberlo intentado):

```
# Función que calcula el total de una compra  
def calcular_total(precio, cantidad):  
    """Multiplica el precio por la cantidad para obtener el total."""  
    return precio * cantidad  
  
# Datos del producto  
precio_producto = 25.50 # Precio unitario en euros  
unidades = 4           # Cantidad de unidades compradas
```

```
# Cálculo del total
total = calcular_total(precio_producto, unidades)

# Mostrar resultados
print(total)          # Muestra el total
# print("Fin del cálculo") # Desactivado temporalmente
```

0.9 Síntesis

- **#** inicia un comentario de una línea.
- **Todo después de #** es ignorado por Python.
- **Comentarios al final de línea** explican código específico.
- **Múltiples #** para comentarios de varias líneas (o triple comilla).
- **Desactivar código** comentando líneas es útil para pruebas.
- **Docstrings** (""") documentan funciones y clases.
- **Buen comentario:** explica el “por qué”, no repite el “qué”.

0.10 Qué sigue

- **Relacionado:** variables (cómo nombrar bien las variables para reducir necesidad de comentarios)
- **Siguiente tema:** Variables — aprenderás a almacenar datos con nombres claros y auto-explicativos.

1 Publicaciones Similares

Si te interesó este artículo, te recomendamos que explores otros blogs y recursos relacionados que pueden ampliar tus conocimientos. Aquí te dejo algunas sugerencias:

1. [!\[\]\(c33cb967c8fc4f5e27188a389b621c8e_img.jpg\) Instalacion De Anaconda](#)
2. [!\[\]\(38e1383487ca0f0e9e2c9378b9dbcae7_img.jpg\) Configurar Entorno Virtual Python Anaconda](#)
3. [!\[\]\(d399648641177ccf0f777d76c74f84ed_img.jpg\) 01 Introducion A La Programacion Con Python](#)
4. [!\[\]\(d32727c446c8638ae1599c3d4f46ad10_img.jpg\) 02 Variables Expresiones Y Statements Con Python](#)
5. [!\[\]\(af3a820412cab4640f1b0ff6288cd856_img.jpg\) 03 Objetos De Python](#)
6. [!\[\]\(12f929b5ec67e02f5e65eeeaf3df99e5_img.jpg\) 04 Ejecucion Condicional Con Python](#)
7. [!\[\]\(7386c1d06e937b48b25a2240ff896668_img.jpg\) 05 Iteraciones Con Python](#)
8. [!\[\]\(ac216e2966b1433b81d3f743e456c9f6_img.jpg\) 06 Funciones Con Python](#)
9. [!\[\]\(d9267fdb68c994fd5365a8e26080ee3f_img.jpg\) 07 Dataframes Con Python](#)
10. [!\[\]\(9f72faae7641ca894b1e3562c818eb20_img.jpg\) 08 Prediccion Y Metrica De Performance Con Python](#)
11. [!\[\]\(19e81320237c07dfaab9b5355b20c5e3_img.jpg\) 09 Metodos De Machine Learning Para Clasificacion Con Python](#)
12. [!\[\]\(456fb49ccbbf407b9d7dd3d48a173498_img.jpg\) 10 Metodos De Machine Learning Para Regresion Con Python](#)
13. [!\[\]\(2abb90db705b89fc778379f2ba2854e3_img.jpg\) Visualizacion De Datos Con Python](#)

Esperamos que encuentres estas publicaciones igualmente interesantes y útiles. ¡Disfruta de la lectura!