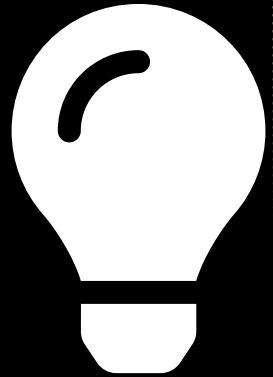


Programación con IA Generativa

Refactorización de código

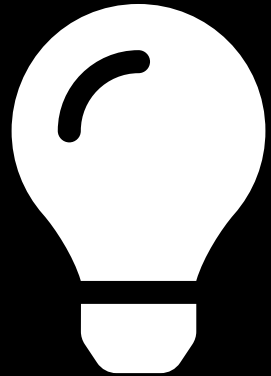


La refactorización de código es el **proceso de reestructuración del código existente sin cambiar su comportamiento.**





Las ventajas de la refactorización incluyen **mejorar la legibilidad del código, reducir la complejidad, facilitar el mantenimiento** del código para permitir que se agreguen más fácilmente nuevas características.



Índice

1. **Mejorar el nombre de variables y funciones**
2. **Optimizar más el código**
3. **Eliminar código repetido**
4. **Reducir la cantidad de líneas de código**
5. **Modularizar más el código**
6. **Reescritura de código condicional**

1 | Mejorar el nombre de variables y funciones



Mejorar el nombre de variables y funciones

Los nombres bien elegidos pueden ayudar a facilitar el mantenimiento del código.

Los asistentes pueden sugerir nombres alternativos para símbolos como variables, funciones, clases, etc.

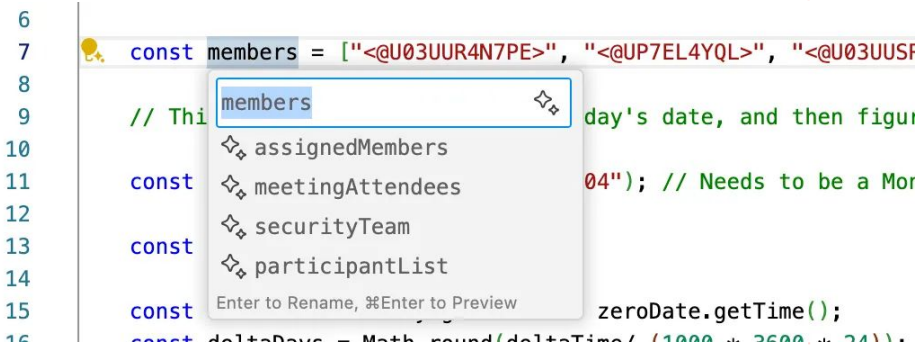
Mejorar el nombre de variables y funciones

1 Seleccionar símbolo

Coloque el cursor en el nombre de la variable, función o clase.

2 Solicitar sugerencias

Presionar la tecla **F2**



The screenshot shows a code editor with a variable named `members` assigned an array of strings. The cursor is positioned on the word `members`. A dropdown menu is open, displaying several suggestions: `assignedMembers`, `meetingAttendees`, `securityTeam`, and `participantList`. At the bottom of the menu, there are instructions: "Enter to Rename, ⌘Enter to Preview". The background code includes comments in Spanish and other variable assignments.

```
6
7  const members = ["<@U03UUR4N7PE>", "<@UP7EL4YQL>", "<@U03UUSF
8
9  // This is the day's date, and then figure out the day's date, and then figure
10
11  const assignedMembers = ["<@U03UUR4N7PE>", "<@UP7EL4YQL>", "<@U03UUSF
12
13  const meetingAttendees = ["<@U03UUR4N7PE>", "<@UP7EL4YQL>", "<@U03UUSF
14
15  const securityTeam = ["<@U03UUR4N7PE>", "<@UP7EL4YQL>", "<@U03UUSF
16  const participantList = ["<@U03UUR4N7PE>", "<@UP7EL4YQL>", "<@U03UUSF
17
18  const zeroDate = new Date(0);
19  const deltaDays = Math.round(deltaTime / (1000 * 60 * 60 * 24));
```

3 Elegir nombre

En la lista desplegable, seleccionar uno de los nombres sugeridos.

Mejorar el nombre de variables y funciones

1 Seleccionar símbolo

Coloque el cursor en el nombre de la variable, función o clase.



```
def calculateDaysBetweenDates(a, b):  
    return b - a
```

2 Prompt en el chat

Abrir chat con Control+Command+i (Mac) /

Ctrl+Alt+i (Windows/Linux)

Ej prompt: Mejorar los nombres de las variables en esta función

3 Elegir nombre

Tomar la respuesta del chat



```
def calculateDaysBetweenDates(begin, end):  
    return end - begin
```


2 | Optimizar más el código



Optimizar más el código

Los asistentes de IA pueden ayudarnos a optimizar el código; por ejemplo, para que el código se **ejecute más rápidamente** y/o **utilice de manera más eficiente los recursos** computacionales como CPU, GPU, RAM, I/O, etc.

Optimizar más el código

1 Seleccionar código

Seleccionar todo el código a optimizar



```
def buscar_valor(lista, valor):  
    posicion = -1  
  
    for i in range(len(lista)):  
        if lista[i] == valor:  
            posicion = i  
  
    return posicion
```

2 Prompt en el chat

Optimizar este código



```
def buscar_valor(lista, valor):  
    while i < len(lista) && lista[i] != valor:  
        i += 1  
  
    if i == len(lista):  
        i = -1  
  
    return i
```

3 Respuesta

Código optimizado

Optimizar más el código

1 Seleccionar código

Seleccionar todo el código a optimizar



```
def buscar_valor(lista, valor):  
    posicion = -1  
  
    for i in range(len(lista)):  
        if lista[i] == valor:  
            posicion = i  
  
    return posicion
```

} una vez encontrado el valor sigue iterando

2 Prompt en el chat

Optimizar este código



```
def buscar_valor(lista, valor):  
    while i < len(lista) && lista[i] != valor:  
        i += 1  
  
    if i == len(lista):  
        i = -1  
  
    return i
```

3 Respuesta

Código optimizado

Optimizar más el código

1 Seleccionar código

Seleccionar todo el código a optimizar



```
def buscar_valor(lista, valor):  
    posicion = -1  
  
    for i in range(len(lista)):  
        if lista[i] == valor:  
            posicion = i  
  
    return posicion
```

una vez encontrado
el valor sigue
iterando

2 Prompt en el chat

Optimizar este código



```
def buscar_valor(lista, valor):  
    while i < len(lista) && lista[i] != valor:  
        i += 1  
  
    if i == len(lista):  
        i = -1  
  
    return i
```

una vez encontrado
el valor deja de
iterar

3 Respuesta

Código optimizado

3 | Eliminar código repetido



Eliminar código repetido

Evitar la duplicidad permite que el **código sea más fácil de revisar y depurar**. Por ejemplo, si el mismo cálculo se realiza más de una vez en diferentes lugares se puede realizar el cálculo en una función.

Optimizar más el código

1 Seleccionar código

Seleccionar todo el código a optimizar



```
total_sales = 0

apple_price = 3
apples_sold = 100
total_sales += apple_price * apples_sold

orange_price = 5
oranges_sold = 50
total_sales += orange_price * oranges_sold
```

2 Prompt en el chat

mover cálculos repetidos a funciones



```
def calculate_sales(price, quantity):
    return price * quantity

total_sales = 0

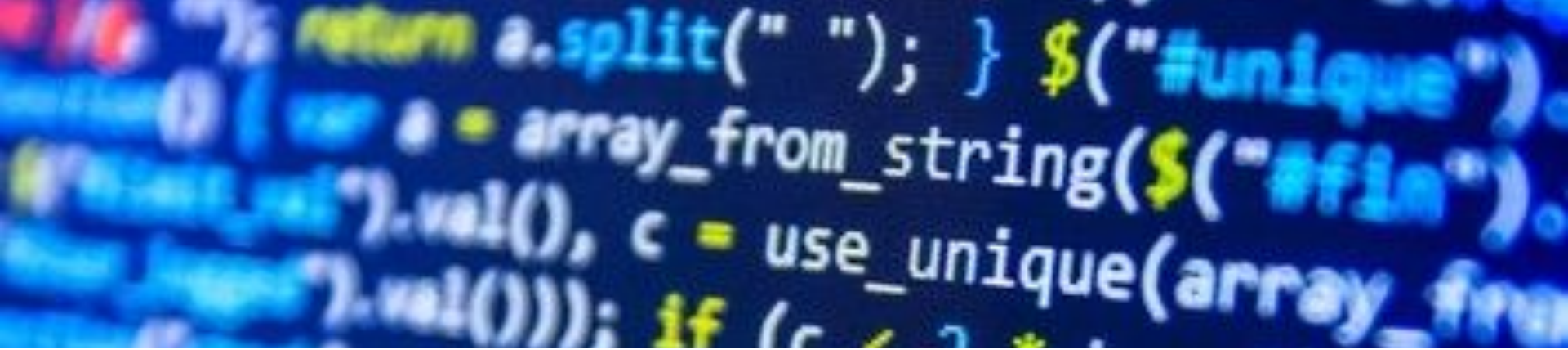
apple_price = 3
apples_sold = 100
total_sales += calculate_sales(apple_price, apples_sold)

orange_price = 5
oranges_sold = 50
total_sales += calculate_sales(orange_price, oranges_sold)
```

3 Respuesta

Código optimizado

4 | Reducir la cantidad de líneas de código



Reducir la cantidad de líneas de código

Si el código es innecesariamente detallado, puede ser difícil de leer y mantener. Los asistentes de código con IA pueden **sugerir una versión más concisa del código** seleccionado.

Reducir la cantidad de líneas de código

1 Seleccionar código

Seleccionar todo el código a optimizar



```
def calculate_area_of_rectangle(length, width):  
    area = length * width  
    return area  
  
def calculate_area_of_circle(radius):  
    area = math.pi * (radius ** 2)  
    return area
```

2 Prompt en el chat

hacer este código más conciso



```
def calculate_area_of_rectangle(length, width):  
    return length * width  
  
def calculate_area_of_circle(radius):  
    return math.pi * (radius ** 2)
```

3 Respuesta

Código optimizado

4 | Modularizar más el código



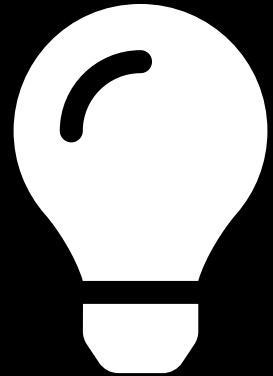
Modularizar más el código

Es probable que los métodos o funciones grandes que realizan varias operaciones ofrezcan menos oportunidades de reutilización que las funciones más pequeñas y sencillas que se centran en realizar una operación determinada. También pueden ser más difíciles de entender y depurar.

“

Los asistentes de código con IA pueden ayudarnos a **dividir bloques complejos** de código **en unidades más pequeñas** que sean más adecuadas **para su reutilización.**

”



Modularizar más el código

1 Seleccionar código

Seleccionar todo el código a optimizar

2 Prompt en el chat

modularizar

3 Respuesta

Código optimizado



```
import pandas as pd

def process_data(item, price):

    item = item.strip()    # Strip whitespace from item
    price = price.strip()  # Strip whitespace from price
    price = float(price)   # Convert price to a float

    data = {'Item': [item], 'Price': [price]}
    df = pd.DataFrame(data)
    print(df.to_string(index=False))
```

Modularizar más el código

1 Seleccionar código

Seleccionar todo el código a optimizar

2 Prompt en el chat

modularizar

3 Respuesta

Código optimizado



```
import pandas as pd

def process_data(item, price):
    # Cleanse data
    item = item.strip() # Strip whitespace from item
    price = price.strip() # Strip whitespace from price
    price = float(price) # Convert price to a float

data = {'Item': [item], 'Price': [price]}
df = pd.DataFrame(data)
print(df.to_string(index=False))
```


Modularizar más el código

1 Seleccionar código

Seleccionar todo el código a optimizar

2 Prompt en el chat

modularizar

3 Respuesta

Código optimizado



```
import pandas as pd

def process_data(item, price):
    # Cleanse data
    item = item.strip() # Strip whitespace from item
    price = price.strip() # Strip whitespace from price
    price = float(price) # Convert price to a float

    # Create and print a DataFrame
    data = {'Item': [item], 'Price': [price]}
    df = pd.DataFrame(data)
    print(df.to_string(index=False))
```

Modularizar más el código

1 Seleccionar código

Seleccionar todo el código a optimizar

2 Prompt en el chat

modularizar

3 Respuesta

Código optimizado



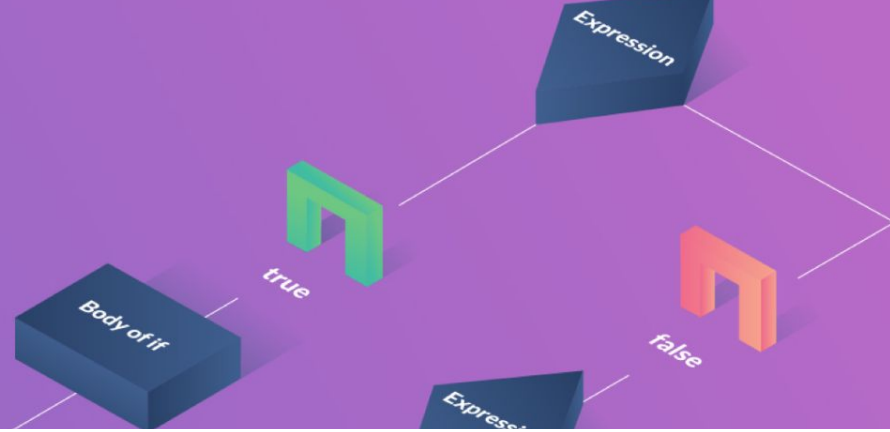
```
import pandas as pd

def cleanse_data(item, price):
    # Cleanse data
    item = item.strip() # Strip whitespace from item
    price = price.strip() # Strip whitespace from price
    price = float(price) # Convert price to a float
    return item, price

def print_data(item, price):
    # Create and print a DataFrame
    data = {'Item': [item], 'Price': [price]}
    df = pd.DataFrame(data)
    print(df.to_string(index=False))

def process_data(item, price):
    item, price = cleanse_data(item, price)
    print_data(item, price)
```

6 | Reescritura de código condicional



Reescritura de código condicional

Las estructuras profundamente anidadas pueden dificultar el seguimiento del código, lo que dificulta a su vez la modificación o ampliación de la lógica del código en el futuro.

Reescritura de código condicional

1 Seleccionar código

Seleccionar todo el código a optimizar

2 Prompt en el chat

Reescribir este código para evitar las declaraciones if/else anidadas

3 Respuesta

Código optimizado



```
def determine_access(user_role, has_permission, is_active):  
    if user_role == "admin":  
        if has_permission:  
            if is_active:  
                "Active admin account with full access."  
            else:  
                "Inactive admin account."  
        else:  
            "Admin account lacks necessary permissions."  
    else:  
        "Access denied."
```

Reescritura de código condicional

1 Seleccionar código

Seleccionar todo el código a optimizar

2 Prompt en el chat

Reescribir este código para evitar las declaraciones if/else anidadas

3 Respuesta

Código optimizado



```
def determine_access(user_role, has_permission, is_active):  
    if user_role != "admin":  
        return "Access denied."  
  
    if not has_permission:  
        return "Admin account lacks necessary permissions."  
  
    if not is_active:  
        return "Inactive admin account."  
  
    return "Active admin account with full access."
```

Actividad en grupo

Armar un ejemplo de refactorización de código solicitando al asistente al menos dos de las siguientes acciones:

- Mejorar el nombre de variables y funciones
- Optimizar más el código
- Eliminar código repetido
- Reducir la cantidad de líneas de código
- Modularizar más el código
- Reescritura de código condicional



?



Dudas