Programación con IA Generativa

Prompt engineering para Copilot

Índice

1 Fundamentos de la Ingeniería de Promts

Fundamentos de la Ingeniería de Prompts

¿Qué es un Prompt?

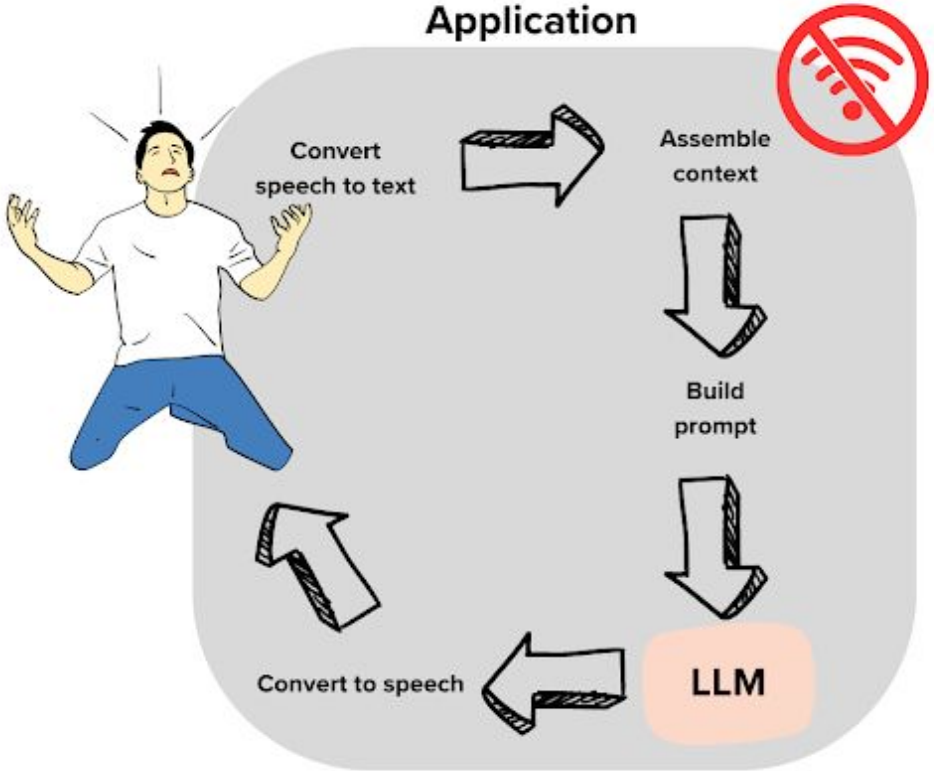
Es una **instrucción** dada a un asistente de programación. Guía la respuesta de la IA.

Claridad

Evita la ambigüedad en tus instrucciones. Facilita la **comprensión** por parte de la IA.

Especificidad

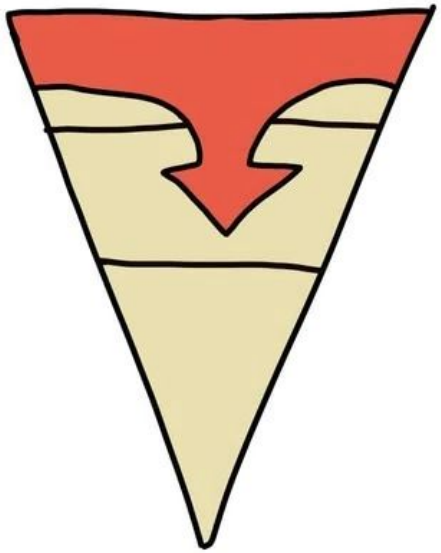
Incluye detalles relevantes para la tarea. Cuanto más **específico**, mejor.


Application
Convert
speech to text
Assemble
context
Build
prompt
LLM
Convert to speech

2 | Técnicas avanzadas

Técnicas Avazadas

De lo general a lo específico



Primero proporcionar una **descripción general** del objetivo o escenario. Luego, enumere los **requisitos específicos**.

Ej:

General

Escribe una función que me diga si un número es primo

La función debe tomar un número entero y devolver verdadero si el número entero es primo.

La función debería generar un error si la entrada no es un entero positivo

Específico

Técnicas Avazadas

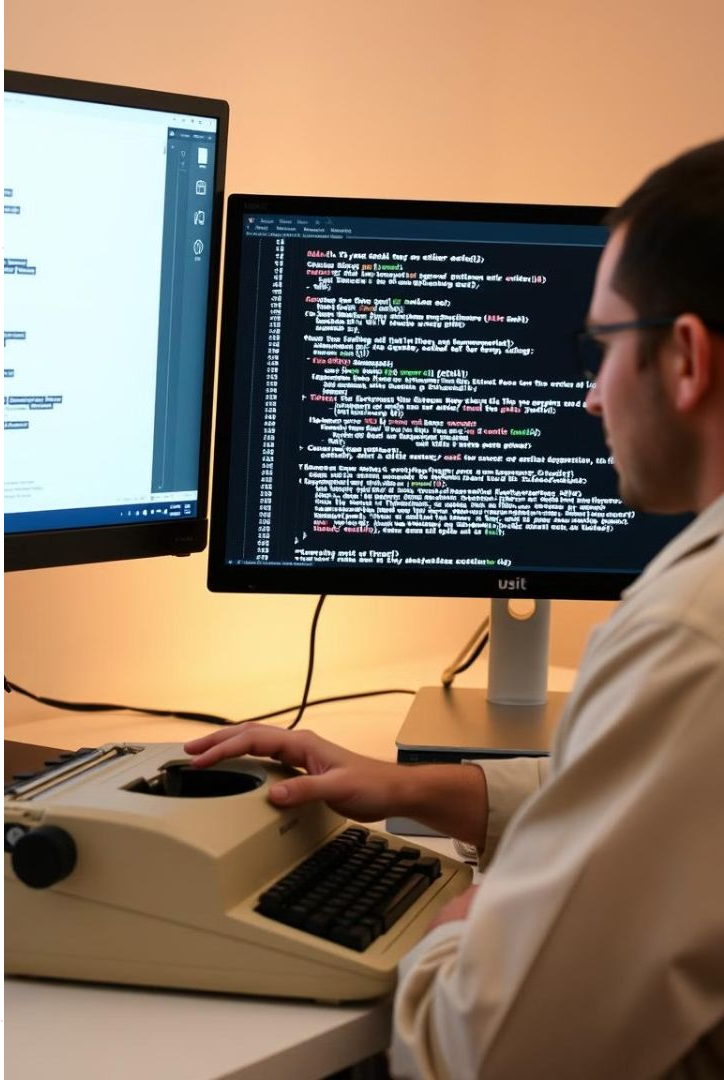
Dar ejemplos

Utilice ejemplos para ayudar a comprender lo que desea. Se puede proporcionar **ejemplos de datos de entrada**, ejemplos de **resultados** y ejemplos de **implementaciones**.

Desarrollar una función que encuentre todas las fechas en una cadena y las devuelva en una matriz. Las fechas se pueden formatear de la siguiente manera:

ejemplos de datos de entrada

- *24/02/05*
- *02/05/2024*
- *2/5/24*
- *2/5/2024*
- *24-02-05*
- *05-02-2024*
- *24 de febrero de 5*
- *5-2-2024*



Técnicas Avazadas

Dar ejemplos

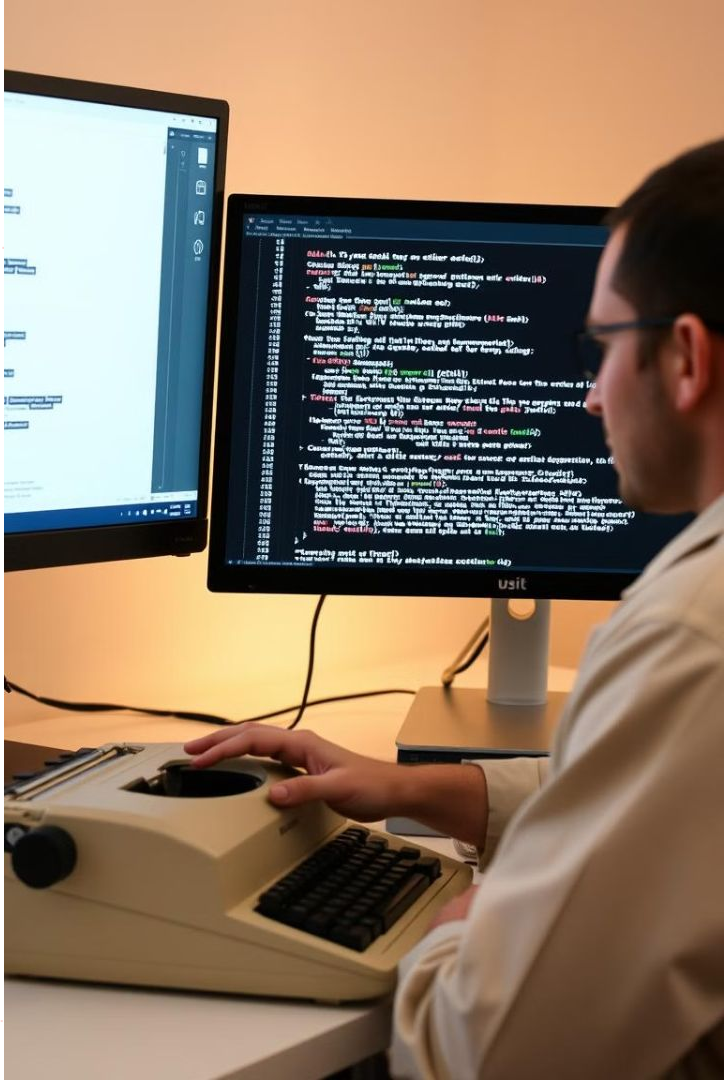
Utilice ejemplos para ayudar a comprender lo que desea. Se puede proporcionar **ejemplos de datos de entrada**, ejemplos de **resultados** y ejemplos de **implementaciones**.

Desarrollar una función que encuentre todas las fechas en una cadena y las devuelva en una lista. Las fechas se pueden formatear de la siguiente manera:

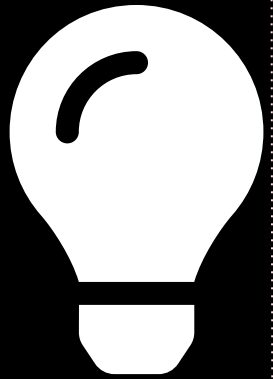
ejemplos de datos de entrada

- *24/02/05*
- *02/05/2024*
- *2/5/24*
- *2/5/2024*
- *24-02-05*
- *05-02-2024*
- *24 de febrero de 5*
- *5-2-2024*

findDates("Tengo una cita con el dentista el 14/11/2023 y un club de lectura el 1/12/23")
Regresa: ["14/11/2023", "1-12-23"]

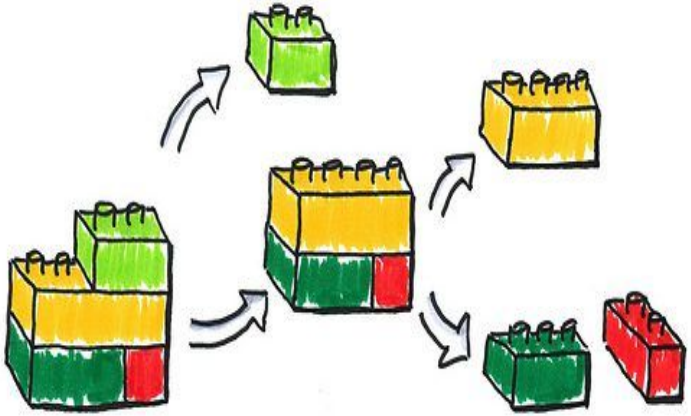


Las pruebas unitarias también pueden servir como ejemplos. Antes de escribir la función, podemos utilizar el asistente para **escribir pruebas unitarias** para la función. Luego, solicitamos que escriba una **función descrita por esas pruebas unitarias**.



Técnicas Avazadas

Divide y venceras



Si deseamos que el asistente complete una tarea compleja o grande, **dividamos la tarea en varias tareas pequeñas y simples**.

Por ejemplo, en lugar de solicitar generar una sopa de letras, dividamos el proceso en tareas más pequeñas y pidamos que las complete una por una:

Escriba una función para generar una cuadrícula de letras de 10 por 10.

Escriba una función para encontrar todas las palabras en una cuadrícula de letras, dada una lista de palabras válidas.

Escriba una función que utilice las funciones anteriores para generar una cuadrícula de letras de 10 por 10 que contenga al menos 10 palabras.

Actualice la función anterior para imprimir la cuadrícula de letras y 10 palabras aleatorias de la cuadrícula.

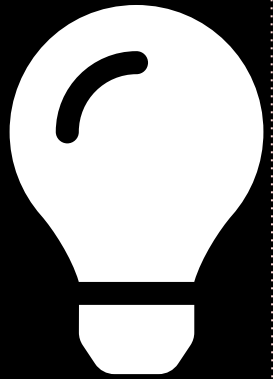
Técnicas Avazadas

Evitar la ambiguedad

Evitar los **términos ambiguos**. Por ejemplo, no preguntar "qué hace esto" si "esto" podría ser el archivo actual, la ultima respuesta del asistente o un bloque de código específico. En lugar de eso, seamos específicos:

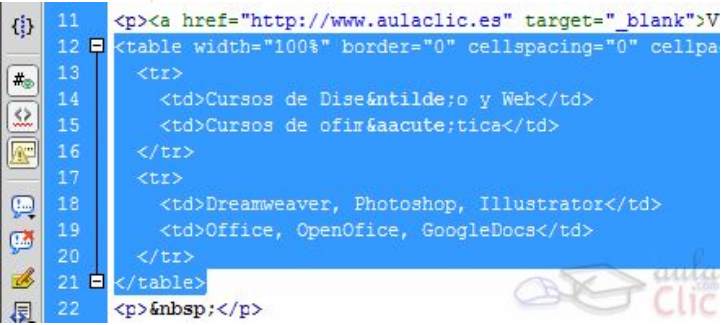
¿Qué hace la createUserfunción?
¿Qué hace el código en tu última respuesta?

Si desea utilizar una biblioteca específica, configure las declaraciones de importación en la parte superior del archivo o **especifique qué biblioteca desea utilizar**.



Técnicas Avazadas

Indicar código relevante



Si utilizamos sugerencias mientras escribimos código, abramos los archivos relevantes y cerremos los que no lo sean. Los asistentes utilizan los archivos abiertos para comprender el requerimiento.

Si usamos el chat, abramos el archivo o resaltemos el código relevante. También podemos usar palabras clave para proporcionar contexto manualmente. Por ejemplo, @workspace para que tenga en cuenta todo el proyecto.

Técnicas Avazadas

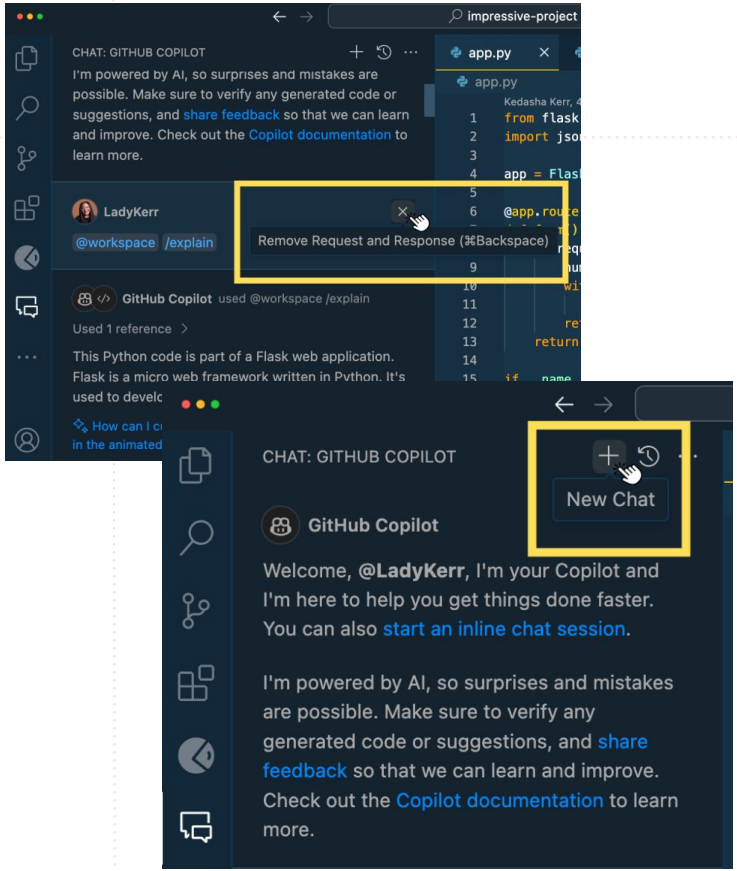
Experimentar e Iterar

Si utilizamos sugerencias mientras escribimos código, podemos eliminar la sugerencia por completo y comenzar de nuevo, o podemos conservar la sugerencia y solicitar modificaciones.

Si usamos los chat en los asistentes, podemos hacer referencia a la respuesta anterior en la próxima solicitud o bien, podemos eliminar la respuesta anterior y comenzar de nuevo.

Técnicas Avazadas

Mantener el historial relevante



Los asistentes utilizan el historial del chat para obtener contexto sobre nuestras solicitudes.
Para proporcionar al asistente solo el historial relevante:

Utilicemos hilos para iniciar una **nueva conversación** para cada nueva tarea.

Eliminemos solicitudes que ya no sean relevantes o que no nos hayan dado el resultado deseado.

Técnicas Avazadas

Utilizar Best Practices

Si no obtienemos las respuestas que deseamos cuando le solicitamos sugerencias o explicaciones al asistente sobre nuestro código base, asegurémonos de que el código existente siga las **mejores prácticas y sea fácil de leer.**

Por ejemplo:

Utilicemos un estilo de **código consistente** y **patrones de diseño**.

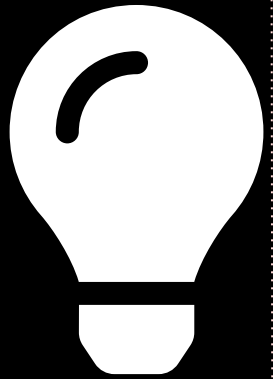
Usemos **nombres descriptivos** para variables y funciones.

Realizar **comentarios** en el código.

Diseñemos **módulos** y **clases**.

Incluyamos **pruebas unitarias**

Utilicemos al **asistente** para ayudar a que nuestro código siga las **mejores prácticas**. Por ejemplo, solicitándole que agregue comentarios o que divida una función grande en funciones más pequeñas.



Actividad en grupo

Armar un ejemplo de prompt utilizando al menos dos de las siguientes técnicas:

- De lo general a lo específico
- Dar ejemplos
- Divide y vencerás
- Evitar la ambiguedad
- Indicar código relevante
- Experimentar e iterar
- Mantener el historial relevante
- Utilizar best practices

? Dudas