Ejercicio 1: Primer contacto con un LLM para generar código sencillo

Objetivo

Ver cómo un LLM puede generar una función completa a partir de una instrucción sencilla, y cómo influye el prompt en la calidad del resultado.

Instrucciones

1. Abre ChatGPT o GitHub Copilot.
2. Escribe el siguiente prompt:
   `Escribe una función en Python que reciba una lista de números y devuelva los elementos únicos ordenados.`
3. Observa la respuesta.
4. Refina el prompt:
   - Añade: "Incluye un ejemplo de uso".
   - Añade: "Comenta el código línea por línea".
   - Añade: "Genera una prueba unitaria".

Qué observar

- ¿El código es correcto?
- ¿Es eficiente?
- ¿Qué cambia al refinar el prompt?

Preguntas para discutir

- ¿Qué parte del código has entendido mejor gracias a los comentarios?
- ¿Qué parte del prompt fue clave para mejorar el resultado?
- ¿Cómo adaptarías el prompt para un caso más complejo?

Resultado esperado

```
def elementos_unicos_ordenados(lista):
    return sorted(set(lista))

# Ejemplo
print(elementos_unicos_ordenados([4, 2, 4, 1]))  # [1, 2, 4]
```

Ejercicio 2: Identificación de errores en código generado por un LLM

Objetivo

Reconocer que los LLMs pueden generar código incorrecto o mal razonado, y aprender a detectar fallos.

Instrucciones

1. Presenta este caso en clase:

```
function intersection(arr1, arr2) {
    let set = new Set(arr1, arr2);
    return Array.from(set);
}
// Ejemplo: intersection([1,2,3], [3,4,5]) → [1,2,3,4,5]
```

2. Pregunta: ¿El código hace lo que promete?
3. Luego, pide que corrijan el código.

Qué observar

- ¿Qué errores detectan?
- ¿Saben por qué `new Set(arr1, arr2)` está mal?
- ¿Confunden intersección con unión?

Preguntas para discutir

- ¿Cómo verificaron que el código era incorrecto?
- ¿Qué prompt usarías para evitar este tipo de errores?

Resultado esperado

```
function intersection(arr1, arr2) {
    return [...new Set(arr1)].filter(x => arr2.includes(x));
}
```

Ejercicio 3: Iteración con un LLM para ajustar una función

Objetivo

Practicar cómo guiar al LLM para modificar una función que inicialmente no cumple con todos los requisitos.

Instrucciones

1. Enviar el prompt inicial:
   `Escribe una función factorial en Python de forma recursiva.`
2. Luego, añade:
   `Ajusta la función para que devuelva 1 si el número es negativo.`
3. Finalmente, pide:
   `Añade docstring y tests con pytest.`

Qué observar

- ¿Cómo responde el modelo a cada modificación?
- ¿Introduce errores nuevos?
- ¿Sabe mantener la función anterior?

Preguntas para discutir

- ¿Cómo validaste que el código final es correcto?
- ¿Te resultó más útil guiar al modelo o modificar tú mismo?
- ¿Qué aprendiste sobre cómo estructurar tus peticiones?

Resultado esperado

```
def factorial(n: int) -> int:
    """Calcula el factorial de n. Para n < 0, retorna 1 por convenio."""
    if n < 0:
        return 1
    if n <= 1:
        return 1
    return n * factorial(n-1)
```