

# Instrucciones Copilot (S1 : req4, req6)

## Usuario

Hola requiero un script de python que lea la información de un archivo de documentación markdown y la muestre en consola, como ves el .md se divide en capítulos (titulares h2 o ##) requiero que el programa solicite al usuario la elección de un capítulo debe tener mensaje de bienvenida y despedida así como opción para finalizar el ciclo de peticiones al usuario, adapta automáticamente la ruta de ingreso del archivo.

## Copilot

Excelente

Tu requerimiento está claro: quieres un programa en Python que:

1. Lea el contenido del archivo Markdown ( .md ).
2. Identifique los capítulos (líneas que empiezan con ## ).
3. Permita al usuario elegir uno para mostrar su contenido.
4. Tenga un mensaje de bienvenida y despedida.
5. Funcione en bucle hasta que el usuario decida salir.

Aquí tienes el **script Python completo** y comentado, listo para ejecutarse:

```
# programa: leer_markdown.py
# Descripción: Lee el archivo Documentacion.md y permite al usuario elegir un
# capítulo (##) para visualizar su contenido.

def leer_markdown(ruta_archivo):
    """Lee el archivo markdown y devuelve su contenido línea por línea."""
    with open(ruta_archivo, 'r', encoding='utf-8') as archivo:
        return archivo.readlines()

def obtener_capitulos(lineas):
    """Devuelve un diccionario con los títulos de nivel 2 (##) y su
    contenido."""
    capitulos = {}
    titulo_actual = None
    contenido_actual = []

    for linea in lineas:
        if linea.startswith("## "):
```

```

        # Si ya había un capítulo previo, lo guardamos antes de iniciar el
nuevo
        if titulo_actual:
            capitulos[titulo_actual] = "".join(contenido_actual).strip()
            contenido_actual = []
            titulo_actual = linea.strip().replace("## ", "")
        elif titulo_actual:
            contenido_actual.append(linea)

# Guarda el último capítulo
if titulo_actual:
    capitulos[titulo_actual] = "".join(contenido_actual).strip()

return capitulos

def mostrar_menu(capitulos):
    """Muestra el menú de capítulos disponibles."""
    print("\nCapítulos disponibles:\n")
    for i, capitulo in enumerate(capitulos.keys(), 1):
        print(f"{i}. {capitulo}")
    print(f"{len(capitulos) + 1}. Salir")

def main():
    print("👋 Bienvenido al lector de capítulos del archivo Documentacion.md")
    ruta = "Proyecto Aurelion/Documentacion.md"

    try:
        lineas = leer_markdown(ruta)
        capitulos = obtener_capitulos(lineas)

        if not capitulos:
            print("⚠️ No se encontraron capítulos (##) en el archivo.")
            return

        while True:
            mostrar_menu(capitulos)
            opcion = input("\nSeleccione un capítulo por número: ")

            try:
                opcion = int(opcion)
            except ValueError:
                print("❌ Por favor ingrese un número válido.\n")
                continue

```

```

        if opcion == len(capitulos) + 1:
            print("\n👋 Gracias por usar el lector de Markdown. ¡Hasta
pronto!")
            break
        elif 1 <= opcion <= len(capitulos):
            titulo = list(capitulos.keys())[opcion - 1]
            print(f"\n--- {titulo} ---\n")
            print(capitulos[titulo])
            print("\n" + "-" * 50 + "\n")
        else:
            print("⚠️ Opción fuera de rango.\n")

    except FileNotFoundError:
        print("❌ No se encontró el archivo README.md en el directorio
actual.")

if __name__ == "__main__":
    main()

```

## Usuario

indicame como verifico si los valores de una columna son unicos

## Copilot

Para verificar si los valores de una columna son únicos en un DataFrame de pandas, hay varias formas. Te muestro las más comunes:

1. Usando el método `is_unique`:

```

# Retorna True si todos los valores son únicos, False si hay duplicados
columna_es_unica = df_clientes['nombre_columna'].is_unique
print(columna_es_unica)

```

2. Usando `nunique()` vs `len()`:

```

# Compara el número de valores únicos con el total de registros
total_valores = len(df_clientes['nombre_columna'])
valores_unicos = df_clientes['nombre_columna'].nunique()
print(f"Total valores: {total_valores}")
print(f"Valores únicos: {valores_unicos}")
print(f"¿Son todos únicos?: {total_valores == valores_unicos}")

```

3. Para ver los valores duplicados:

```
# Muestra las filas que tienen valores duplicados en la columna
duplicados = df_clientes[df_clientes['nombre_columna']].duplicated()
print(duplicados)
```

4. Para contar cuántas veces se repite cada valor:

```
# Muestra un conteo de cada valor
conteo = df_clientes['nombre_columna'].value_counts()
print(conteo)
```

## Usuario

recuerdame como hago un heatmap de correlaciones

## Copilot

Usa seaborn sobre la matriz de correlación de las columnas numéricas. Ejemplos cortos:

```
numeric = aurelion.select_dtypes(include=['int64', 'float64'])
corr = numeric.corr()

# heatmap simple
plt.figure(figsize=(10,8))
sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm", vmin=-1, vmax=1,
            linewidths=.5)
plt.title("Matriz de correlación")
plt.show()
```

```
# heatmap con máscara (oculta triángulo superior)
import numpy as np
mask = np.triu(np.ones_like(corr, dtype=bool))

plt.figure(figsize=(10,8))
sns.heatmap(corr, mask=mask, annot=True, fmt=".2f", cmap="vlag", center=0,
            linewidths=.5)
plt.title("Heatmap (triángulo superior oculto)")
plt.show()
```

## Usuario

recuerdame como hago boxplot con seaborn

# Copilot

Ejemplos cortos para hacer boxplots con seaborn (reemplaza aurelion/columnas según tu DataFrame):

```
# 1) Boxplot simple de una columna numérica
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x=aurelion['importe'].dropna())
plt.title('Boxplot de importe')
plt.show()
```

```
# 2) Boxplot por categoría (comparar distribución por grupo)
sns.boxplot(x='categoria', y='importe', data=aurelion)
plt.xticks(rotation=45)
plt.title('Importe por categoría')
plt.tight_layout()
plt.show()
```

```
# 3) Horizontal + ocultar outliers (showfliers=False)
sns.boxplot(x='importe', y='categoria', data=aurelion, showfliers=False)
plt.title('Importe por categoría (sin outliers)')
plt.tight_layout()
plt.show()
```

```
# 4) Grid de boxplots por otra variable con catplot (facets)
sns.catplot(kind='box', x='categoria', y='importe', col='medio_pago',
            data=aurelion, col_wrap=3, sharey=True)
plt.tight_layout()
plt.show()
```

# Usuario

recuerdame como hago boxplot con seaborn

# Copilot

Ejemplos cortos para hacer boxplots con seaborn (reemplaza aurelion/columnas según tu DataFrame):

```
# 1) Boxplot simple de una columna numérica
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x=aurelion['importe'].dropna())
plt.title('Boxplot de importe')
plt.show()
```

```
# 2) Boxplot por categoría (comparar distribución por grupo)
sns.boxplot(x='categoria', y='importe', data=aurelion)
plt.xticks(rotation=45)
plt.title('Importe por categoría')
plt.tight_layout()
plt.show()
```

```
# 3) Horizontal + ocultar outliers (showfliers=False)
sns.boxplot(x='importe', y='categoria', data=aurelion, showfliers=False)
plt.title('Importe por categoría (sin outliers)')
plt.tight_layout()
plt.show()
```

```
# 4) Grid de boxplots por otra variable con catplot (facets)
sns.catplot(kind='box', x='categoria', y='importe', col='medio_pago',
            data=aurelion, col_wrap=3, sharey=True)
plt.tight_layout()
plt.show()
```

## Usuario

ayudame con un grafico de dispersion de la relacion entre precio y cantidad vendida

## Copilot

```
plt.figure(figsize=(8,6))
sns.scatterplot(x='precio_unitario', y='cantidad', data=aurelion,
hue='categoria', alpha=0.7)
plt.title('Relación entre precio unitario y cantidad vendida')
plt.xlabel('Precio unitario')
plt.ylabel('Cantidad vendida')
plt.tight_layout()
plt.show()
```

```
correlacion = aurelion['precio_unitario'].corr(aurelion['cantidad'])
print(f" Correlación precio vs cantidad: {correlacion:.2f}")
```

## Usuario

necesito generar una grafica donde se representen los producto con menor rotación

## copilot

```
productos_menos_vendidos = ventas_por_producto.tail(10)
print("Productos con menor rotación:")
print(productos_menos_vendidos)

plt.figure(figsize=(10,6))
sns.barplot(x='cantidad', y='producto_comprado',
data=productos_menos_vendidos, palette='rocket')
plt.title('Productos con menor rotación')
plt.xlabel('Cantidad vendida')
plt.ylabel('Producto')
plt.tight_layout()
plt.show()
```