4. **Lector de códigos QR:** Genera y lee códigos QR con la librería `qrcode`.

¡Claro! Con gusto te ayudaré a crear una aplicación en Python para generar y leer códigos QR usando la librería qrcode y la librería Pillow para la manipulación de imágenes (necesaria para qrcode en la parte de generación) y cv2 (OpenCV) para la lectura, ya que qrcode en sí misma solo genera y decodifica imágenes ya cargadas, pero cv2 es más práctico para capturar y decodificar desde un archivo o cámara.
Aquí tienes el código completo y las instrucciones de instalación.

?□ⷯ Requisitos e Instalación

Necesitarás instalar las siguientes librerías:

1. qrcode: La librería principal para generar códigos.
2. Pillow (PIL): Necesaria para que qrcode pueda manejar la imagen generada.
3. opencv-python (cv2): La utilizaremos para leer/decodificar el código QR desde un archivo de imagen.

Abre tu terminal o símbolo del sistema y ejecuta:

Bash
```
pip install qrcode[pil] opencv-python
```

? Código Python

Este código define dos funciones: una para **generar** un código QR y otra para **leer** un código QR desde un archivo de imagen.
Guarda el siguiente código como un archivo llamado lector_qr.py:

Python
```
import qrcode
from PIL import Image
import cv2
import os

# --- 1. Generador de Código QR ---
def generar_qr(data, nombre_archivo="codigo_qr.png"):
    """
    Genera un código QR a partir de una cadena de datos y lo guarda como un archivo PNG.

    Args:
        data (str): La información que contendrá el código QR (URL, texto, etc.).
        nombre_archivo (str): El nombre del archivo donde se guardará la imagen.
    """
    try:
        # Crea una instancia de la clase QRCode
        qr = qrcode.QRCode(
            version=1, # Controla el tamaño (1 es el más pequeño y recomendado para URLs cortas)
            error_correction=qrcode.constants.ERROR_CORRECT_L, # Nivel de corrección de error
            box_size=10, # Tamaño de cada "caja" o píxel del QR
            border=4, # Grosor del borde (4 es el mínimo estándar)
        )
```

```
            # Agrega los datos al objeto QR
            qr.add_data(data)
            qr.make(fit=True)

            # Crea una imagen PIL
            img = qr.make_image(fill_color="black", back_color="white")

            # Guarda la imagen
            img.save(nombre_archivo)
            print(f"✅ Código QR generado y guardado como: {nombre_archivo}")

        except Exception as e:
            print(f"❌ Error al generar el QR: {e}")


# --- 2. Lector de Código QR ---
def leer_qr(nombre_archivo):
    """
    Lee y decodifica el contenido de un código QR desde un archivo de imagen.

    Args:
        nombre_archivo (str): La ruta del archivo de imagen del código QR.
    """
    if not os.path.exists(nombre_archivo):
        print(f"❌ Error: El archivo '{nombre_archivo}' no existe.")
        return

    try:
        # Carga la imagen usando OpenCV
        img = cv2.imread(nombre_archivo)

        # Crea el objeto decodificador de códigos de barras (BarcodeDetector)
        detector = cv2.QRCodeDetector()

        # Decodifica la imagen
        data, vertices, _ = detector.detectAndDecode(img)

        if data:
            print("\n============================================")
            print("✅ Contenido del Código QR leído:")
            print(f"   ---> {data}")
            print("============================================")
        else:
            print(f" No se pudo detectar un Código QR válido en '{nombre_archivo}'.")

    except Exception as e:
        print(f"❌ Error al leer el QR con OpenCV: {e}")


# --- Bloque de Ejecución Principal ---
if __name__ == "__main__":

    # ------------------------------------------------
    # DEMOSTRACIÓN 1: Generar un Código QR
    # ------------------------------------------------
```

```
print("--- INICIO DE GENERACIÓN ---")

datos_para_qr = "https://www.google.com/search?q=soy+un+bot+de+google"
archivo_qr_generado = "mi_primer_qr.png"

generar_qr(datos_para_qr, archivo_qr_generado)

# ----------------------------------------------------
# DEMOSTRACIÓN 2: Leer el Código QR que acabamos de generar
# ----------------------------------------------------
print("\n--- INICIO DE LECTURA ---")

# Intentamos leer el archivo que acabamos de crear
leer_qr(archivo_qr_generado)

# Ejemplo de lectura de un archivo que no existe (para prueba de errores)
# leer_qr("archivo_inexistente.png")
```

? Cómo Ejecutarlo

1. Asegúrate de que tienes instaladas las librerías (pip install qrcode[pil] opencv-python).
2. Guarda el código anterior como lector_qr.py.
3. Ejecuta el script desde tu terminal:

```
Bash
python lector_qr.py
```

Resultado Esperado

El script:

1. Generará un archivo llamado mi_primer_qr.png en el mismo directorio con el código QR.
2. Luego, leerá ese mismo archivo.

Verás una salida similar a esta en tu consola:

```
--- INICIO DE GENERACIÓN ---
✅ Código QR generado y guardado como: mi_primer_qr.png

--- INICIO DE LECTURA ---

==========================================
✅ Contenido del Código QR leído:
    ---> https://www.google.com/search?q=soy+un+bot+de+google
==========================================
```

¿Te gustaría que añadiera una interfaz gráfica simple usando una librería como tkinter, o quieres que modifique las funciones para que, por ejemplo, pueda leer desde la cámara web?

¡Entendido! Si quieres que **añada funcionalidad a la aplicación** para generar y leer códigos QR, te propongo el siguiente desarrollo:

? Opción: Leer Códigos QR desde una URL

Ya que la función leer_qr actual está diseñada para leer desde un archivo local, podemos extenderla para que primero descargue la imagen de un código QR desde una **URL** y luego la decodifique con cv2.
Necesitarás una librería adicional para manejar peticiones web: requests.

?□ Requisitos Adicionales

Abre tu terminal y ejecuta:

Bash
```
pip install requests
```

? Código Python Modificado

Aquí está la función modificada y el bloque principal actualizado con la nueva funcionalidad.

Python
```
import qrcode
from PIL import Image
import cv2
import os
import requests
import numpy as np # Necesaria para la manipulación de arrays de imágenes

# Las funciones generar_qr y leer_qr (desde archivo) son las mismas que antes.
# ... (Mantener las funciones generar_qr y leer_qr originales aquí) ...

# --- Nueva Función: Leer Código QR desde una URL ---
def leer_qr_desde_url(url_imagen):
    """
    Descarga una imagen de código QR desde una URL y decodifica su contenido.

    Args:
        url_imagen (str): La URL directa a la imagen del código QR.
    """
    print(f"⏳ Intentando descargar y leer QR desde: {url_imagen}...")
    try:
        # 1. Descargar la imagen
        response = requests.get(url_imagen, stream=True)
        response.raise_for_status() # Lanza un error para códigos de estado HTTP incorrectos (4xx o 5xx)

        # 2. Convertir la respuesta binaria a un array de numpy
        # Esto es necesario para que OpenCV pueda procesar la imagen
        image_bytes = np.asarray(bytearray(response.content), dtype="uint8")
        img = cv2.imdecode(image_bytes, cv2.IMREAD_COLOR)

        if img is None:
            print("✕ Error al decodificar la imagen descargada. Asegúrate de que la URL es una imagen válida.")
            return

        # 3. Decodificar la imagen usando OpenCV
        detector = cv2.QRCodeDetector()
        data, vertices, _ = detector.detectAndDecode(img)
```

```
            if data:
                print("\n==========================================")
                print("✅ Contenido del Código QR leído (desde URL):")
                print(f"   ---> {data}")
                print("==========================================")
            else:
                print("⚠ No se pudo detectar un Código QR válido en la imagen de la URL.")

        except requests.exceptions.RequestException as e:
            print(f"❌ Error de red o URL: {e}")
        except Exception as e:
            print(f"❌ Error inesperado: {e}")

# --- Bloque de Ejecución Principal ---
if __name__ == "__main__":

    # ... (Mantener las demostraciones de generación y lectura desde archivo) ...

    # ------------------------------------------------
    # DEMOSTRACIÓN 3: Leer un Código QR desde una URL
    # ------------------------------------------------
    print("\n--- INICIO DE LECTURA DESDE URL ---")

    # URL de ejemplo de un QR (puedes sustituirla por la que necesites)
    qr_url = "https://chart.googleapis.com/chart?cht=qr&chs=200x200&chl=DatosDePruebaDesdeGoogleCharts"

    leer_qr_desde_url(qr_url)
```

Si te interesa más el proceso interno, el componente clave en la lectura es el método .QRCodeDetector() de **OpenCV (cv2)**. Este objeto busca los **patrones de localización** que tienen todos los códigos QR (los tres cuadrados grandes en las esquinas) para determinar su posición y perspectiva antes de aplicar el algoritmo de decodificación.
¿Qué te parecería si, como siguiente paso, implementamos la lectura de códigos QR en **tiempo real** usando la cámara web de tu ordenador?
Puedes ver cómo se usan OpenCV y Pyzbar para la detección de códigos de barras y QR en tiempo real en la cámara web. Detectar códigos de barras y QR con OPENCV y camara web [Proyecto Python]

¡Perfecto! El siguiente paso lógico, y uno de los más útiles, es implementar la **lectura de códigos QR en tiempo real** usando tu cámara web.
Para esto, utilizaremos las librerías que ya tienes: **OpenCV (cv2)** para acceder a la cámara y procesar el video, y el decodificador de códigos QR incorporado.

Asegúrate de tener instalado:

Si ya ejecutaste los pasos anteriores, deberías tenerlas:

Bash
```
pip install opencv-python
```

? Lector de QR en Tiempo Real (Cámara Web)

Aquí tienes el código de la función, lista para ser integrada en tu script lector_qr.py:

Python

```python
import cv2
import time

def leer_qr_desde_camara():
    """
    Inicia la cámara web y lee códigos QR en tiempo real.

    Presiona 'q' para salir de la ventana de la cámara.
    """
    # 0 indica la primera cámara web disponible. Usa 1, 2, etc., si tienes varias.
    cap = cv2.VideoCapture(0)

    # Inicializa el detector de códigos QR de OpenCV
    detector = cv2.QRCodeDetector()

    print("\n--- INICIO DE LECTURA EN VIVO ---")
    print("Muestra un código QR a la cámara. Presiona 'q' para salir.")

    # Variable para evitar imprimir repetidamente el mismo código
    ultimo_qr_leido = ""

    while True:
        # Captura un frame (cuadro) del video
        ret, frame = cap.read()

        if not ret:
            print("❌ No se pudo capturar la imagen de la cámara.")
            break

        # Intenta decodificar el QR en el frame
        data, vertices, _ = detector.detectAndDecode(frame)

        if data:
            # Dibuja un recuadro verde alrededor del QR detectado
            if vertices is not None:
                # Los vértices contienen las coordenadas del polígono del QR
                vertices = vertices.astype(int)
                cv2.polylines(frame, [vertices], True, (0, 255, 0), 3)

                # Si el código es nuevo, imprímelo
                if data != ultimo_qr_leido:
                    print("\n============================================")
                    print(f"✅ QR LEÍDO: {data}")
                    print("============================================")
                    ultimo_qr_leido = data # Actualiza el último código leído

        # Muestra la ventana de la cámara
        cv2.imshow('Lector de Codigo QR en Vivo (Presiona Q para salir)', frame)

        # La tecla 'q' detiene el ciclo y cierra la ventana
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Libera la cámara y cierra todas las ventanas de OpenCV
```

```
    cap.release()
    cv2.destroyAllWindows()
    print("--- Lector de cámara terminado ---")

# --- Bloque de Ejecución Principal (Integración) ---
if __name__ == "__main__":
    # Importaciones (asegúrate de que estén al principio de tu archivo)
    # import qrcode
    # from PIL import Image
    # import cv2
    # import os

    # Puedes comentar las demostraciones anteriores si solo quieres probar la cámara:

    # -----------------------------------------------------
    # DEMOSTRACIÓN 4: Lector de QR en Tiempo Real
    # -----------------------------------------------------
    leer_qr_desde_camara()
```

?¬ª‹♂ Cómo Ejecutar la Lectura en Vivo

1. Asegúrate de que la función leer_qr_desde_camara() se haya añadido a tu archivo Python (por ejemplo, lector_qr.py).
2. Ejecuta el script desde tu terminal:

   Bash
   ```
   python lector_qr.py
   ```
3. Se abrirá una ventana que mostrará lo que ve tu cámara web.
4. Muestra un código QR (puede ser uno que generaste, uno en tu móvil o impreso). El código debe ser detectado, rodeado con un recuadro verde y el contenido se imprimirá en tu consola.
5. Presiona la tecla **'q'** en la ventana de la cámara para detener el proceso.
   Esta es una aplicación muy robusta, ya que combina la **generación**, la **lectura de archivos** y la **lectura en tiempo real**.