Comment créer votre QR CODE application sous PYTHON:

pour générer un QR Code avec une interface graphique, ainsi que les liens pour les installer.

### 1. ****qrcode****

* **Description** : Bibliothèque pour générer des QR Codes.
* **Installation** :

bash

Copier le code

pip install qrcode[pil]

* **Documentation** : [qrcode GitHub](https://github.com/lincolnloop/python-qrcode)

### 2. ****Pillow (PIL)****

* **Description** : Bibliothèque pour manipuler des images (format PNG, JPEG, etc.).
* **Installation** :

bash

Copier le code

pip install Pillow

* **Documentation** : [Pillow Documentation](https://pillow.readthedocs.io/en/stable/)

### 3. ****requests****

* **Description** : Bibliothèque pour faire des requêtes HTTP (télécharger des images à partir d'URLs, par exemple).
* **Installation** :

bash

Copier le code

pip install requests

* **Documentation** : Requests Documentation

### 4. ****ttkbootstrap****

* **Description** : Bibliothèque qui fournit des thèmes modernes pour tkinter, rendant les interfaces graphiques plus attrayantes.
* **Installation** :

bash

Copier le code

pip install ttkbootstrap

* **Documentation** : [ttkbootstrap GitHub](https://github.com/israel-dryer/ttkbootstrap)

### 5. ****tkinter**** (inclus avec Python)

* **Description** : Bibliothèque standard de Python pour créer des interfaces graphiques. Généralement, elle est incluse par défaut avec les installations de Python, donc aucune installation supplémentaire n'est nécessaire.
* **Documentation** : [Tkinter Documentation](https://docs.python.org/3/library/tkinter.html)

### Résumé des étapes d'installation

Pour installer toutes les bibliothèques nécessaires, tu peux exécuter les commandes suivantes dans ton terminal :

bash

Copier le code

pip install qrcode[pil]

pip install Pillow

pip install requests

pip install ttkbootstrap

Avec ces bibliothèques, tu peux créer des applications Python modernes et interactives, comme celle pour générer des QR Codes.

**CODE PYTHON AVEC GUI**

import tkinter as tk

from tkinter import messagebox

import qrcode

from PIL import Image

import requests

from io import BytesIO

import ttkbootstrap as ttk # Importer ttkbootstrap

def generate\_qr\_code(data, color="black", logo\_url=None, back\_color="white"):

# Créer un QR Code

qr = qrcode.QRCode(

version=1,

error\_correction=qrcode.constants.ERROR\_CORRECT\_H,

box\_size=10,

border=4,

)

qr.add\_data(data)

qr.make(fit=True)

# Créer une image du QR Code

qr\_img = qr.make\_image(fill\_color=color, back\_color=back\_color)

# Ajouter un logo si spécifié

if logo\_url:

try:

response = requests.get(logo\_url)

if response.status\_code == 200:

logo = Image.open(BytesIO(response.content)).convert("RGBA")

logo\_size = (qr\_img.size[0] // 4, qr\_img.size[1] // 4)

logo = logo.resize(logo\_size)

# Calculer la position pour centrer le logo

qr\_img.paste(logo, ((qr\_img.size[0] - logo\_size[0]) // 2, (qr\_img.size[1] - logo\_size[1]) // 2), logo)

else:

messagebox.showerror("Erreur", f"Erreur lors du téléchargement du logo. Code d'état : {response.status\_code}.")

except Exception as e:

messagebox.showerror("Erreur", f"Une erreur s'est produite lors de l'ajout du logo : {e}")

# Enregistrer le QR Code

qr\_img.save("qr\_code.png")

messagebox.showinfo("Succès", "QR Code généré et enregistré sous 'qr\_code.png'.")

def on\_generate():

data = data\_entry.get()

color\_input = color\_entry.get() or "black"

logo\_url = logo\_entry.get()

back\_color\_input = back\_color\_entry.get() or "white"

# Convertir les couleurs

color = convert\_color(color\_input)

back\_color = convert\_color(back\_color\_input)

generate\_qr\_code(data, color, logo\_url if logo\_url else None, back\_color)

def convert\_color(color):

color\_map = {

"rouge": "red",

"vert": "green",

"bleu": "blue",

"jaune": "yellow",

"noir": "black",

"blanc": "white"

}

return color\_map.get(color.lower(), color)

# Créer la fenêtre principale avec un thème moderne

root = ttk.Window(themename="darkly") # Choisir un thème moderne

root.title("Générateur de QR Code")

# Créer les champs d'entrée

ttk.Label(root, text="Données à encoder :").grid(row=0, column=0, padx=10, pady=10)

data\_entry = ttk.Entry(root, width=40)

data\_entry.grid(row=0, column=1, padx=10, pady=10)

ttk.Label(root, text="Couleur du QR Code :").grid(row=1, column=0, padx=10, pady=10)

color\_entry = ttk.Entry(root, width=20)

color\_entry.grid(row=1, column=1, padx=10, pady=10)

ttk.Label(root, text="URL du logo :").grid(row=2, column=0, padx=10, pady=10)

logo\_entry = ttk.Entry(root, width=40)

logo\_entry.grid(row=2, column=1, padx=10, pady=10)

ttk.Label(root, text="Couleur de fond :").grid(row=3, column=0, padx=10, pady=10)

back\_color\_entry = ttk.Entry(root, width=20)

back\_color\_entry.grid(row=3, column=1, padx=10, pady=10)

# Bouton pour générer le QR Code

generate\_button = ttk.Button(root, text="Générer QR Code", command=on\_generate)

generate\_button.grid(row=4, column=0, columnspan=2, pady=20)

# Lancer la boucle principale

root.mainloop()

**CODE PYTHON DIRECTEMENT DANS LE TERMINAL SANS GUI:**

import qrcode

from PIL import Image

import requests

from io import BytesIO

def generate\_qr\_code(data, color="black", logo\_url=None, back\_color="white"):

# Créer un QR Code

qr = qrcode.QRCode(

version=1,

error\_correction=qrcode.constants.ERROR\_CORRECT\_H,

box\_size=10,

border=4,

)

qr.add\_data(data)

qr.make(fit=True)

# Créer une image du QR Code

qr\_img = qr.make\_image(fill\_color=color, back\_color=back\_color) # Change ici pour fill\_color

# Ajouter un logo si spécifié

if logo\_url:

try:

response = requests.get(logo\_url)

print(f"Code d'état du logo : {response.status\_code}") # Afficher le code d'état

if response.status\_code == 200:

logo = Image.open(BytesIO(response.content)).convert("RGBA") # Convertir le logo en RGBA

logo\_size = (qr\_img.size[0] // 4, qr\_img.size[1] // 4) # Redimensionner le logo

logo = logo.resize(logo\_size)

# Calculer la position pour centrer le logo

qr\_img.paste(logo, ((qr\_img.size[0] - logo\_size[0]) // 2, (qr\_img.size[1] - logo\_size[1]) // 2), logo)

else:

print(f"Erreur lors du téléchargement du logo. Code d'état : {response.status\_code}.")

except Exception as e:

print(f"Une erreur s'est produite lors de l'ajout du logo : {e}")

# Enregistrer le QR Code

qr\_img.save("qr\_code.png")

print("QR Code généré et enregistré sous 'qr\_code.png'.")

def convert\_color(color):

color\_map = {

"rouge": "red",

"vert": "green",

"bleu": "blue",

"jaune": "yellow",

"noir": "black",

"blanc": "white"

# Ajoute d'autres couleurs si nécessaire

}

return color\_map.get(color.lower(), color)

# Exemple d'utilisation

if \_\_name\_\_ == "\_\_main\_\_":

data = input("Entrez les données à encoder dans le QR Code : ")

color\_input = input("Entrez la couleur du QR Code (par défaut: noir) : ") or "black"

logo\_url = input("Entrez l'URL du logo (laisser vide si pas de logo) : ")

back\_color\_input = input("Entrez la couleur de fond du QR Code (par défaut: blanc) : ") or "white"

# Convertir les couleurs française en couleur anglaise

color = convert\_color(color\_input)

back\_color = convert\_color(back\_color\_input)

generate\_qr\_code(data, color, logo\_url if logo\_url else None, back\_color)