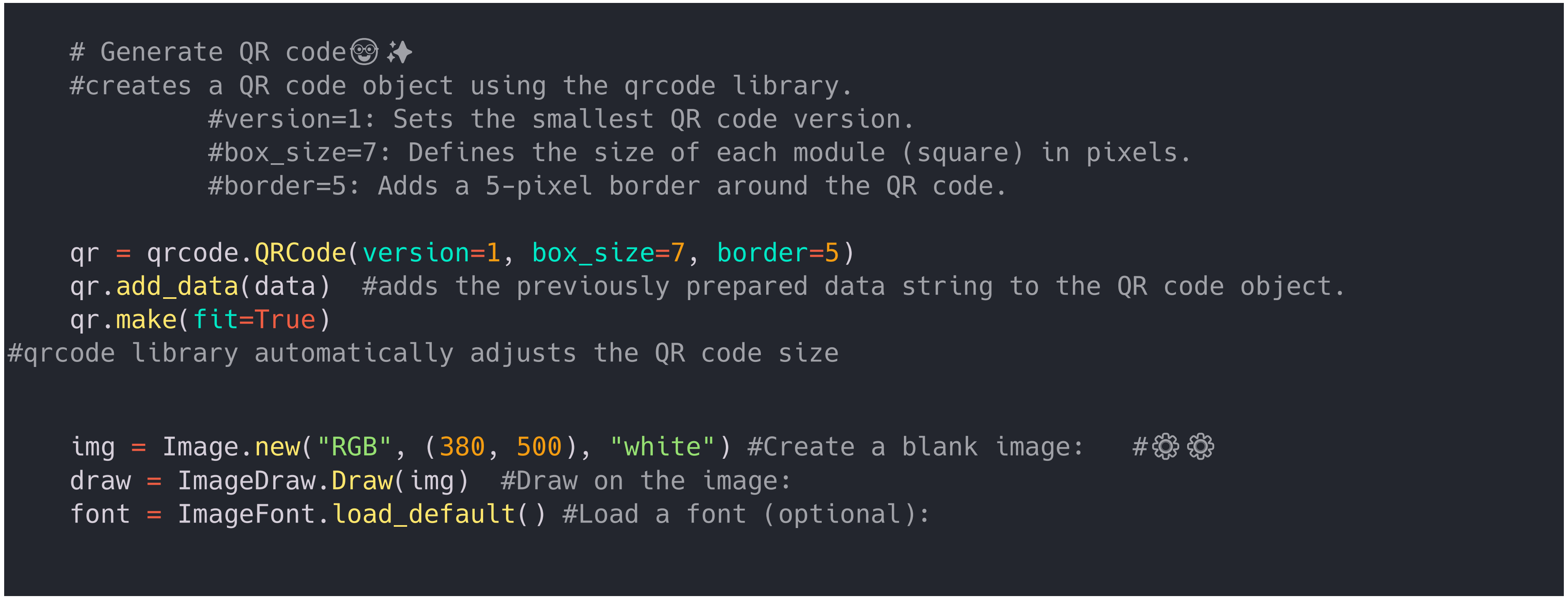
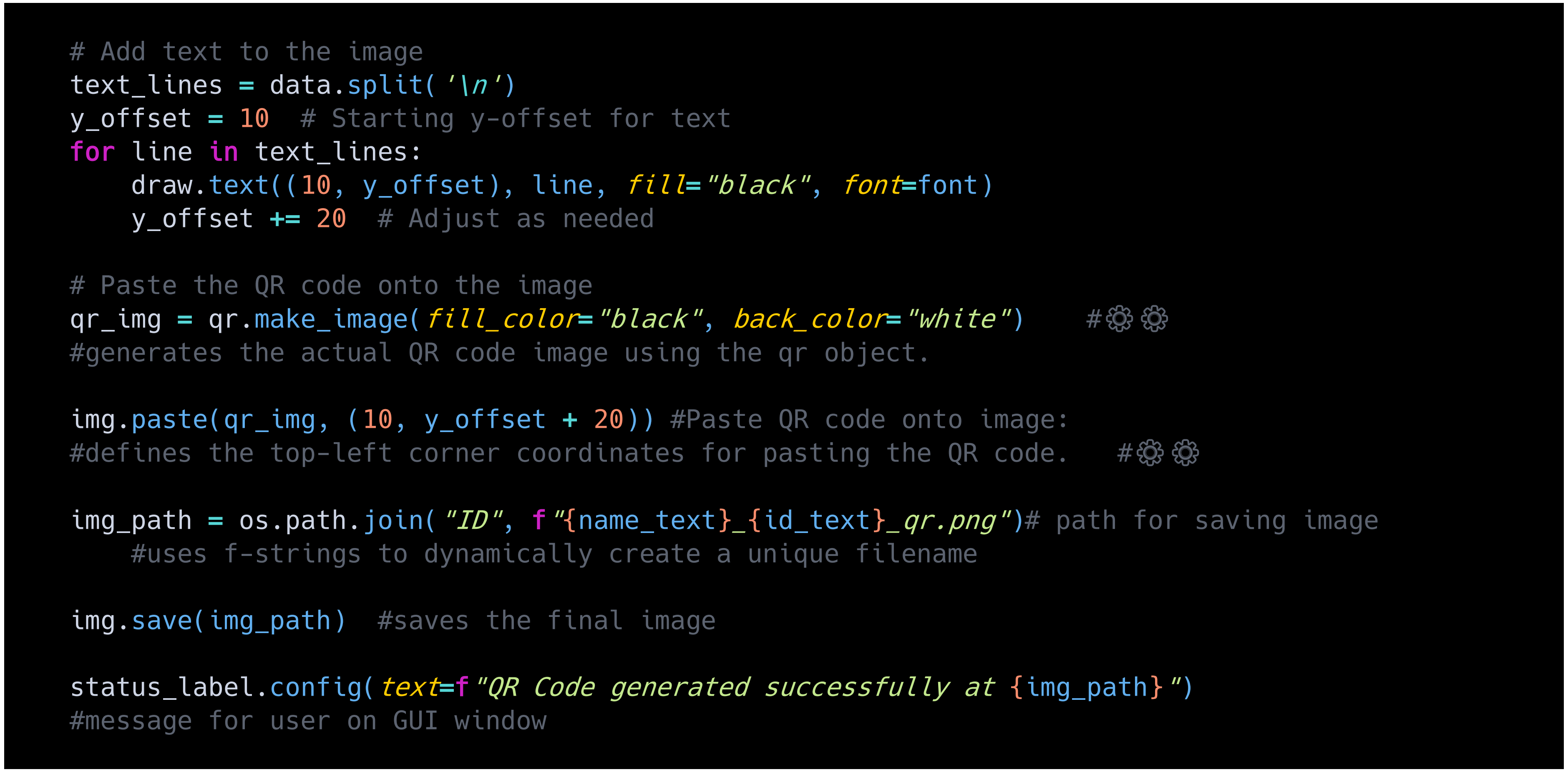
```python
import os                  #  allows you to interact with the (OS) from your Python code
import tkinter as tk       #  GUI toolkit for Python.
from tkinter import ttk #  Themed Tkinter, an extension to Tkinter with themed widgets.
import qrcode              #  A library to generate QR codes.
from PIL import Image, ImageDraw, ImageFont # PIL used for creating, drawing, and manipulating images.
```

```python
#___________________________________________________________________________________________________#
#Gets the input data from the Tkinter Entry widgets (name_entry, id_entry, etc.).using get()
#Creates a string data with the formatted information.
def generate_qr():
    name_text = name_entry.get()
    id_text = id_entry.get()
    mobile_text = mobile_entry.get()
    email_text = email_entry.get()
    blood_group_text = blood_group_entry.get()

    data = f"Name: {name_text}\nID: {id_text}\nPhone No.: {mobile_text}\nMail: {email_text}\nBlood
Group: {blood_group_text}"
#uses f-strings to create a formatted string  by combining retrieved  data

    # Create "ID" folder if not exists
    if not os.path.exists("ID"):
        os.makedirs("ID")
```



```python
    # Generate QR code🤓✨
    #creates a QR code object using the qrcode library.
            #version=1: Sets the smallest QR code version.
            #box_size=7: Defines the size of each module (square) in pixels.
            #border=5: Adds a 5-pixel border around the QR code.

    qr = qrcode.QRCode(version=1, box_size=7, border=5)
    qr.add_data(data)  #adds the previously prepared data string to the QR code object.
    qr.make(fit=True)
#qrcode library automatically adjusts the QR code size


    img = Image.new("RGB", (380, 500), "white") #Create a blank image:   #⚙️⚙️
    draw = ImageDraw.Draw(img)  #Draw on the image:
    font = ImageFont.load_default() #Load a font (optional):
```



```python
# Add text to the image
text_lines = data.split('\n')
y_offset = 10  # Starting y-offset for text
for line in text_lines:
    draw.text((10, y_offset), line, fill="black", font=font)
    y_offset += 20  # Adjust as needed

# Paste the QR code onto the image
qr_img = qr.make_image(fill_color="black", back_color="white")    #⚙⚙
#generates the actual QR code image using the qr object.

img.paste(qr_img, (10, y_offset + 20)) #Paste QR code onto image:
#defines the top-left corner coordinates for pasting the QR code.   #⚙⚙

img_path = os.path.join("ID", f"{name_text}_{id_text}_qr.png")# path for saving image
    #uses f-strings to dynamically create a unique filename

img.save(img_path)  #saves the final image

status_label.config(text=f"QR Code generated successfully at {img_path}")
#message for user on GUI window
```

```python
#______________________________________________________________________________________________#
# Create main window which holds all other UI elements
window = tk.Tk()
window.title("ID Card Generator") #Sets the title displayed at the top

# Name Entry
name_label = ttk.Label(window, text="Name:") # Creates a label widget with the text "Name:".
#ttk.Label indicates using the ttk theme better than tk.Label

name_label.grid(row=0, column=0, padx=10, pady=10, sticky=tk.W)
#places the label widget on the grid layout manager.
#(first row and column , padding space around Label ,
# sricky : Makes the label stick to the west side (left) of its cell)

name_entry = ttk.Entry(window) # Creates an entry widget (text box) where the user can enter their
name.

name_entry.grid(row=0, column=1, padx=10, pady=10)
#Positions the entry widget next to the label on the grid

# ID Entry
id_label = ttk.Label(window, text="ID:")
id_label.grid(row=1, column=0, padx=10, pady=10, sticky=tk.W)
id_entry = ttk.Entry(window)
id_entry.grid(row=1, column=1, padx=10, pady=10)

# Mobile Entry
mobile_label = ttk.Label(window, text="Mobile:")
mobile_label.grid(row=2, column=0, padx=10, pady=10, sticky=tk.W)
mobile_entry = ttk.Entry(window)
```

```python
mobile_entry.grid(row=2, column=1, padx=10, pady=10)

# Email Entry
email_label = ttk.Label(window, text="Email:")
email_label.grid(row=3, column=0, padx=10, pady=10, sticky=tk.W)
email_entry = ttk.Entry(window)
email_entry.grid(row=3, column=1, padx=10, pady=10)

# Blood Group Entry
blood_group_label = ttk.Label(window, text="Blood Group:")
blood_group_label.grid(row=4, column=0, padx=10, pady=10, sticky=tk.W)
blood_group_entry = ttk.Entry(window)
blood_group_entry.grid(row=4, column=1, padx=10, pady=10)
```

```python
#__________________________________________________________________________________________________#
# Generate QR Button
#Adds a Button widget (generate_button) that triggers the generate_qr() function.
generate_button = ttk.Button(window, text="Generate QR", command=generate_qr)
generate_button.grid(row=5, column=0, columnspan=2, pady=20)

# Status Label
#This label will be used to display messages
#to the user, such as "Generating QR..." or "QR generated successfully!"
status_label = ttk.Label(window, text="")
status_label.grid(row=6, column=0, columnspan=2)

# Run the application
#This keeps the window running and responsive to user interactions
#like button clicks until explicitly closed.
window.mainloop()
```