Enhance Python Apps With Modern Icons Using Iconipy

What is iconipy?

Iconipy is a versatile Python package designed for creating and managing icons directly from your code.

It supports multiple icon sets and allows seamless integration with various GUI frameworks like Qt (PyQt6 / PySide6), Tkinter, PySimpleGUI, Kivy, DearPyGUI and more.

With Iconipy, you can customize icon properties such as size, color, and background, making it an essential tool for developers looking to enhance their application’s visual appeal.

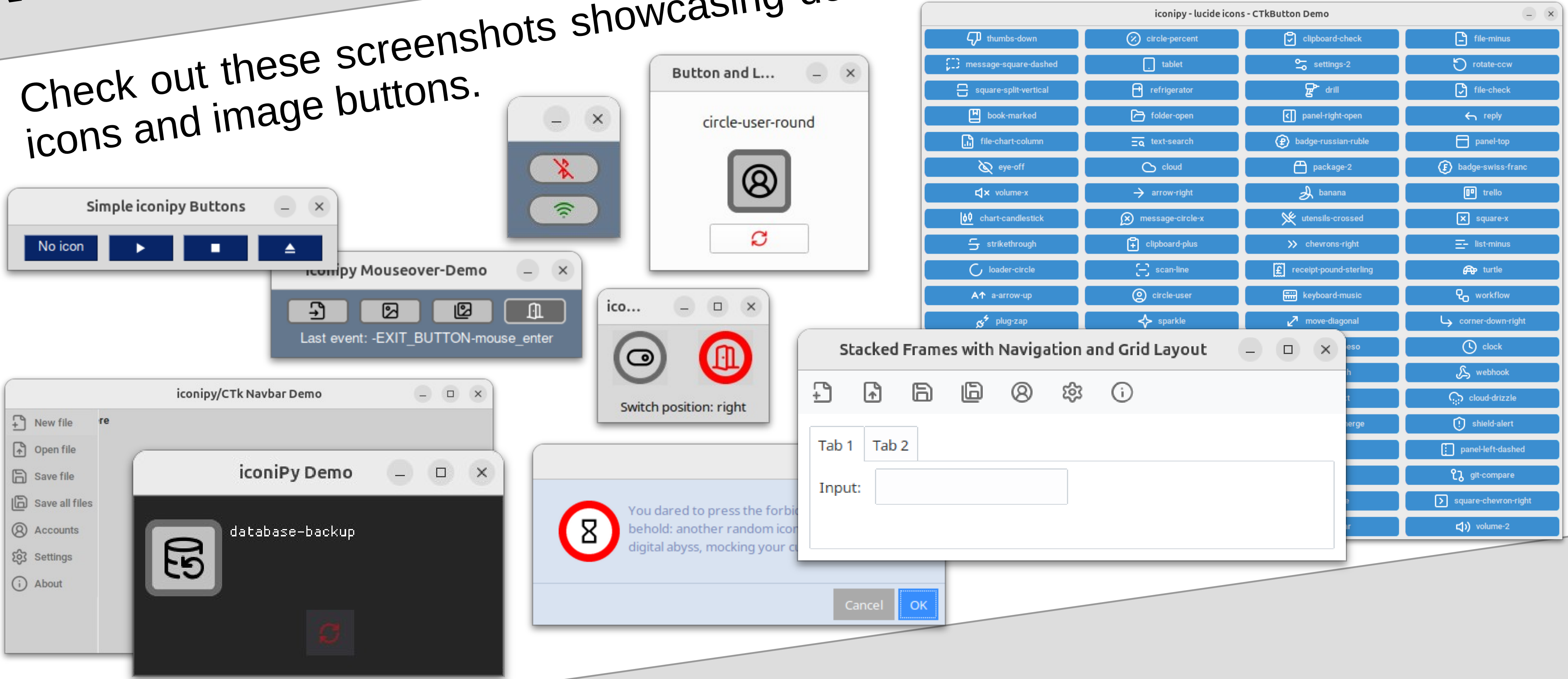
Its ease of use and flexibility make it a popular choice for both beginners and experienced developers

Key Features And Benefits

- **Versatile Icon Creation:** Supports multiple popular icon sets (Lucide Icons, Google Material Icons, Lineicons, Boxicons) and allows for extensive customization of icon properties such as size, color, and background. (Developers can add their own icon sets as well)
- **Multi-Framework Compatibility:** Seamlessly integrates with various GUI frameworks including Qt, Tkinter, FreeSimpleGUI, NiceGUI, DearPyGUI, and so on - making it adaptable for different projects.
- **Ease of Use:** Simple and intuitive API design that makes it easy for both beginners and experienced developers to create and manage icons.
- **Enhanced Visual Appeal:** Helps in enhancing the visual appeal of applications by providing high-quality, customizable icons.
- **Efficient Development:** Saves time and effort in icon creation, allowing developers to focus more on core functionality and less on design details.
- **Open Source:** Being open-source, it encourages community contributions and continuous improvement.

What does it look like?

Check out these screenshots showcasing demo applications featuring iconipy icons and image buttons.



Last event: -EXIT_BUTTON-mouse_enter

Switch position: right

You dared to press the forbi
behold: another random icon
digital abyss, mocking your c

Install iconipy

To install the iconipy package in Python, you can use the following command in your terminal or command prompt:

```
pip install iconipy
```

Once installed, you can start using it to create icons directly from your code. Here’s a basic example to get you started:

```
from iconipy import IconFactory


# Initialize an IconFactory with desired settings
create_icon = IconFactory(icon_size = 20)


# Create an icon
icon_home = create_icon.asPil('house')  # PIL Image
```

The IconFactory concept

The IconFactory concept is a powerful way to manage and customize the appearance of icons in your application. Here’s how it works:

- **Centralized Design:** By creating an IconFactory for each style, you centralize the design settings in one place of your Python code. This means you can easily update the look and feel of all icons by modifying the IconFactory settings, rather than changing each icon individually.
- **Code Reusability:** Instead of duplicating code for each icon, you define the style once in the IconFactory. This promotes code reusability and reduces redundancy, making your codebase cleaner and more maintainable.
- **Consistency:** Using an IconFactory ensures that all icons created with it share the same style attributes, such as size, color, and outline. This consistency is crucial for a cohesive user interface.
- **Flexibility:** You can create multiple IconFactory instances for different states or themes. For example, one IconFactory for the normal button state and another for the mouse-over state. This allows you to switch styles dynamically based on user interactions or application states.

Customize your icon’s look-and-feel

You can easily customize the appearance of your icons by creating an IconFactory for each style you need. For example, you can have one IconFactory for the normal button state and another for the mouse-over state. This approach allows you to quickly update the design for all icons / buttons by modifying a single location in your Python code.

```
from iconipy import IconFactory
create_button_icon = IconFactory(
                        icon_set = 'lucide',
                        icon_size = (128,64),
                        font_size = 38,
                        font_color = (0, 0, 0, 255), # black solid
                        outline_color = 'dimgrey',
                        outline_width = 6,
                        background_color = '#5500FF',
                        background_radius = 10
)
```

Icon Sets

You can choose from the icon sets that ship with iconipy:

- lucide
- boxicons
- lineicons
- material_icons_regular
- material_icons_round_regular
- material_icons_sharp_regular
- material_icons_outlined_regular

Check out the docs to learn how to add your own icon sets.

```
from iconipy import IconFactory
create_icon = IconFactory(icon_set='lucide')
```

Icon Names

To create an icon you need to know it’s name.

You can search for matching icon names like this:

```
print(create_icon.search('hand'))
```

Or you can print all available icon names for the IconFactory’s icon set:

```
print(create_icon.icon_names)
```

Preview an icon with:

```
create_icon.show('house')
```

```
from iconipy import IconFactory
create_icon = IconFactory(icon_set='lucide')

print(create_button_icon.search('hand'))

print(create_button_icon.icon_names)
```

Icon Formats

Not all GUI frameworks support the same image formats. Therefore, you need to use the appropriate function to create your icons. For instance, some frameworks can use PIL images directly, while others have their own formats (such as QPixmap or PhotoImage) or simply accept paths to actual files.

```
icon_home = create_button_icon.asPil('house') # CustomTkinter, wxPython, NiceGUI
icon_save = create_button_icon.asBytes('save') # PySimpleGUI, FreeSimpleGUI
icon_files = create_button_icon.asTkPhotoImage('files') # tkinter based
icon_folder = create_button_icon.asTkBitmapImage('folder') # tkinter based
icon_reload = create_button_icon.asQPixmap('refresh-cw') # PyQt, PySide
icon_exit_app = create_button_icon.asRawList('log-out') # DearPyGUI
icon_sticker = create_button_icon.asTempFile('sticker') # Kivy and all the rest...
```

Docs

For comprehensive sample code and detailed documentation on the most popular GUI toolkits, visit

GitHub: https://github.com/digidigital/iconipy

or

Iconipy’s official site: https://iconipy.digidigital.de

These resources provide valuable insights and examples to help you create stunning icons directly from your Python code, tailored to various GUI frameworks.

Explore these links to streamline your icon creation process and enhance your user interfaces with ease.

„Hello World“ Code Examples

In the following slides, we’ll explore some basic examples of using iconipy with various frameworks.

Please note that these examples only scratch the surface of the customization options available for your icons.

For more comprehensive and realistic scenarios, follow the links provided at the bottom of each slide to access detailed examples and advanced documentation.

Buttons with icons in tkinter

```
import tkinter as tk

from iconipy import IconFactory

# Initialize IconFactory (default settings except for a custom icon size of 20)
create_icon = IconFactory(icon_size=20)

# Create the main application window
window = tk.Tk()

# Generate an icon of a globe
world_icon = create_icon.asTkPhotoImage('globe')

# Create a button with text and an icon, then add it to the window
button = tk.Button(window, text='Hello World!', image=world_icon, compound=tk.LEFT)
button.grid(row=0, column=0, padx=10, pady=10)

# Run the application's event loop
window.mainloop()
```

Check out more complex examples here: https://github.com/digidigital/iconipy/tree/main/demo_programs/tkinter

PySimpleGUI and FreeSimpleGUI Icon Button

```
try:
    import FreeSimpleGUI as sg
except:
    import PySimpleGUI as sg

from iconipy import IconFactory

create_icon = IconFactory(icon_size=20) # Initialize IconFactory
icon_bytes = create_icon.asBytes('globe') # Create the icon as Bytes

layout = [
    [sg.Text('Hello World Button Demo')],
    [sg.Button('', image_data=icon_bytes, button_color=(sg.theme_background_color(), sg.theme_background_color()),
border_width=0, key='Exit')]
]

window = sg.Window('Hello world button!', layout)

while True:
    event, values = window.read()
    if event in (sg.WIN_CLOSED, 'Exit'):
        Break
```

Check out the other examples here: https://github.com/digidigital/iconipy/tree/main/demo_programs/FreeSimpleGUI_PySimpleGUI

Buttons with image in ttk

```
import tkinter as tk
from tkinter import ttk
from iconipy import IconFactory

# Initialize IconFactory (default settings except for a custom icon size of 20)
create_icon = IconFactory(icon_size=20)

# Create the main application window
window = tk.Tk()

# Generate an icon of a globe
world_icon = create_icon.asTkPhotoImage('globe')

# Create a button with text and an icon, then add it to the window
button = ttk.Button(window, text='Hello World!', image=world_icon, compound=tk.LEFT)
button.grid(row=0, column=0, padx=10, pady=10)

# Run the application's event loop
window.mainloop()
```

Check out more complex examples here: https://github.com/digidigital/iconipy/tree/main/demo_programs/ttk

Add image to button in ttkbootstrap

```
import tkinter as tk
import ttkbootstrap as ttk
from iconipy import IconFactory

# Initialize IconFactory (default settings except for a custom icon size of 20)
create_icon = IconFactory(icon_size=20)

# Create the main application window
window = tk.Tk()

# Apply the ttkbootstrap style
style = ttk.Style(theme="cosmo")

# Generate an icon of a globe
world_icon = create_icon.asTkPhotoImage('globe')

# Create a button with text and an icon, then add it to the window
button = ttk.Button(window, text='Hello World!', image=world_icon, compound=tk.LEFT)
button.grid(row=0, column=0, padx=10, pady=10)

# Run the application's event loop
window.mainloop()
```

Check out more complex examples here: https://github.com/digidigital/iconipy/tree/main/demo_programs/ttkbootstrap

CTkButton with icon / image

```
import customtkinter as ctk
from iconipy import IconFactory

# Initialize IconFactory (default settings except for a custom icon size of 20 and font color 'white')
create_icon = IconFactory(icon_size=20, font_color='white')

# Create the main application window
window = ctk.CTk()

# Generate an icon of a globe
world_icon = create_icon.asTkPhotoImage('globe')

# Create a button with text and an icon, then add it to the window
button = ctk.CTkButton(window, text='Hello World!', image=world_icon, compound='left')
button.grid(row=0, column=0, padx=10, pady=10)

# Run the application's event loop
window.mainloop()
```

Check out more complex examples here: https://github.com/digidigital/iconipy/tree/main/demo_programs/customtkinter

PyQt buttons with image

```
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton, QVBoxLayout, Qwidget
from PyQt6.QtGui import Qicon
from PyQt6.QtCore import Qt
from iconipy import IconFactory
import sys

create_icon = IconFactory(icon_size=18, font_color='dimgrey') # Initialize IconFactory

app = QApplication(sys.argv) # Initialize the application
window = QMainWindow() # Create the main window
central_widget = QWidget() # Create a central widget and set a layout
layout = QVBoxLayout(central_widget)

globe_icon=create_icon.asQPixmap('globe') # Create icon as Qpixmap

button = QPushButton("Hello World!") # Create a button with text and an icon
button.setIcon(QIcon(globe_icon))layout.addWidget(button) # Add the button to the layout

window.setCentralWidget(central_widget) # Set the central widget
window.show() # Show the main window

sys.exit(app.exec()) # Run the application's event loop
```

Check out more complex examples here: https://github.com/digidigital/iconipy/tree/main/demo_programs/PyQt6

PySide – Add image to button

```
from PySide6.QtWidgets import QApplication, QMainWindow, QPushButton, QVBoxLayout, Qwidget
from PySide6.QtGui import Qicon
from PySide6.QtCore import Qt
from iconipy import IconFactory
import sys

create_icon = IconFactory(icon_size=18, font_color='dimgrey') # Initialize IconFactory

app = QApplication(sys.argv) # Initialize the application
window = QMainWindow() # Create the main window
central_widget = QWidget() # Create a central widget and set a layout
layout = QVBoxLayout(central_widget)

globe_icon=create_icon.asQPixmap('globe') # Create icon as Qpixmap

button = QPushButton("Hello World!") # Create a button with text and an icon
button.setIcon(QIcon(globe_icon))layout.addWidget(button) # Add the button to the layout

window.setCentralWidget(central_widget) # Set the central widget
window.show() # Show the main window

sys.exit(app.exec()) # Run the application's event loop
```

Check out the other examples here: https://github.com/digidigital/iconipy/tree/main/demo_programs/PySide6

wxPython BitmapButton

```
import wx
from iconipy import IconFactory

app = wx.App()
frame = wx.Frame(None, -1, 'wxPython Hello World! Button')
frame.SetDimensions(0, 0, 200, 70)

panel = wx.Panel(frame, wx.ID_ANY)

# Initialize IconFactory for a simple icon with transparent background
create_icon = IconFactory(icon_set = 'lucide',
                icon_size = 18,
                font_color = 'grey') # (reg, green, blue. alpha)

# Create the icon as temp file and load it
# You can replace 'globe' with any valid icon name for the icon set
bmp = wx.Bitmap(create_icon.asTempFile('globe'), wx.BITMAP_TYPE_ANY)

# Create a BitmapButton with the loaded image
button = wx.BitmapButton(panel, id=wx.ID_ANY, bitmap=bmp, size=(64, 32))

frame.Show()
app.MainLoop()
```

More examples here: https://github.com/digidigital/iconipy/tree/main/demo_programs/wxPython

More „Hello World“ Code Examples

That’s not all!

Iconipy supports even more frameworks than those covered in this presentation.

To explore how to integrate with additional toolkits, visit these websites:

Kivy → https://github.com/digidigital/iconipy/tree/main/demo_programs/kivy

NiceGUI → https://github.com/digidigital/iconipy/tree/main/demo_programs/NiceGUI

DearPyGui → https://github.com/digidigital/iconipy/tree/main/demo_programs/DearPyGUI

appJar → https://github.com/digidigital/iconipy/tree/main/demo_programs/appJar