A second, less elegant, but functional option for all operating systems, is to base64-encode the content of the icons and include them in our source code file. The image data can then be loaded directly from a variable via tk.PhotoImage. To do this, we first need some code that reads the content of an image and encodes it with Base64. This can be easily achieved with a small Python program:

# encode\_icon.py

from base64 import b64encode

import sys

import pyperclip

filename = sys.argv[1]

with open(filename, "rb") as f:

pyperclip.copy(b64encode(f.read()).decode("ascii"))

print("Copied to clipboard.")

This little program receives the name of an icon by console, encodes it with base64 and copies it to the clipboard. The Pyperclip module is required (installable via pip install pyperclip). Now let's start by enocding the small icon by running on the terminal:

python encode\_icon.py icon-16.png

Next, let's paste the data copied to the clipboard into a variable and load it via tk.PhotoImage:

from base64 import b64decode

import tkinter as tk

small\_icon\_data = "paste-data-here"

root = tk.Tk()

root.title("Window With Icon")

root.geometry("300x200")

small\_icon = tk.PhotoImage(data=b64decode(small\_icon\_data))

root.iconphoto(False, small\_icon)

root.mainloop()

Let's do the same with the large icon:

python encode\_icon.py icon-32.png

And finally:

from base64 import b64decode

import tkinter as tk

small\_icon\_data = "paste-small-icon-data-here"

large\_icon\_data = "paste-large-icon-data-here"

root = tk.Tk()

root.title("Window With Icon")

root.geometry("300x200")

small\_icon = tk.PhotoImage(data=b64decode(small\_icon\_data))

large\_icon = tk.PhotoImage(data=b64decode(large\_icon\_data))

root.iconphoto(False, large\_icon, small\_icon)

root.mainloop()

Great! Our icon data is embedded in the code, so there is no need to distribute additional .png files.