```
from tkinter import *
import tkinter.font
from tkinter.colorchooser import *
from tkinter import simpledialog

# NEW Used to save image data and fonts
from PIL import Image, ImageDraw, ImageTk

# NEW Used to get the file name for saving
import os

# NEW The save dialog
import tkinter.filedialog

# Create main window
root = Tk()
root.geometry("800x600")


class PaintApp:
    text_font = StringVar()
    text_size = IntVar()
    bold_text = StringVar()
    italic_text = StringVar()

    # Stores current tool we are using
    drawing_tool = StringVar()

    # STORE DRAWING SETTINGS
    stroke_size = IntVar()
    fill_color = StringVar()
    stroke_color = StringVar()

    # Tracks whether left mouse is down
    left_but = "up"

    # x and y positions for drawing with pencil
    x_pos, y_pos = None, None

    # Tracks x & y when the mouse is clicked and released
    x1_line_pt, y1_line_pt, x2_line_pt, y2_line_pt = None, None, None, None

    # NEW Empty image to draw on with width, height and color
    my_image = Image.new("RGB", (800, 600), (255, 255, 255))

    # NEW Used to draw shapes
    draw = ImageDraw.Draw(my_image)

    # NEW created so I can draw files to canvas
    drawing_area = Canvas(root, width=800, height=600)

    # Quits the TkInter app when called
    @staticmethod
    def quit_app():
```

```
    root.quit()

# New saves PIL image as PNG
# NOTE : If your image is blurry it is best to create an image
# that is twice as large as what you want to save and then use
# resize to 1/2 the size and use filter to smooth out the drawings
# More can be found here
# https://pillow.readthedocs.io/en/stable/reference/Image.html#PIL
# .Image.Image.resize
def save_file(self, event=None):
    # Opens the save as dialog box
    file = tkinter.filedialog.asksaveasfile(mode='w', defaultextension=".png")
    if file:
        file_path = os.path.abspath(file.name)
        self.my_image.save(file_path)

# NEW Opens the PIL image
def open_file(self, event=None):
    file_path = tkinter.filedialog.askopenfilename(parent=root)
    if file_path:

        # Load the image
        my_pic = Image.open(file_path)

        # Put the image in a canvas class
        self.drawing_area.image = ImageTk.PhotoImage(my_pic)

        # Draw the image starting in the upper left
        self.drawing_area.create_image(0, 0,
                    image=self.drawing_area.image,
                    anchor='nw')


def make_menu_bar(self):
    # Create the menu object
    the_menu = Menu(root)

    # ---- FILE MENU ----
    # Create a pull down menu that can't be removed
    file_menu = Menu(the_menu, tearoff=0)

    # Add items to the menu that show when clicked
    # compound allows you to add an image

    # NEW Add option to open and save files
    file_menu.add_command(label="Open",
                    command=self.open_file)
    file_menu.add_command(label="Save",
                    command=self.save_file)

    # Add a horizontal bar to group similar commands
    file_menu.add_separator()

    # Call for the function to execute when clicked
```

```
file_menu.add_command(label="Quit", command=self.quit_app)

# Add the pull down menu to the menu bar
the_menu.add_cascade(label="File", menu=file_menu)

# ---- FONT MENU ----
font_menu = Menu(the_menu, tearoff=0)
font_type_submenu = Menu(font_menu)
font_type_submenu.add_radiobutton(label="Times",
                    variable=self.text_font)
font_type_submenu.add_radiobutton(label="Courier",
                    variable=self.text_font)
font_type_submenu.add_radiobutton(label="Ariel",
                    variable=self.text_font)
font_menu.add_cascade(label="Font Type",
                  menu=font_type_submenu)

font_size_submenu = Menu(font_menu)
font_size_submenu.add_radiobutton(label="10",
                            variable=self.text_size,
                            value=10)
font_size_submenu.add_radiobutton(label="15",
                            variable=self.text_size,
                            value=15)
font_size_submenu.add_radiobutton(label="20",
                            variable=self.text_size,
                            value=20)
font_size_submenu.add_radiobutton(label="25",
                            variable=self.text_size,
                            value=25)
font_menu.add_cascade(label="Font Size",
                  menu=font_size_submenu)

# NEW FIX THE ON AND OFF VALUES FOR BOLD & ITALIC
font_menu.add_checkbutton(label="Bold",
                    variable=self.bold_text,
                    onvalue='bold',
                    offvalue='normal')
font_menu.add_checkbutton(label="Italic",
                    variable=self.italic_text,
                    onvalue='italic',
                    offvalue='roman')

the_menu.add_cascade(label="Font", menu=font_menu)

# ---- TOOL MENU ----
tool_menu = Menu(the_menu, tearoff=0)
tool_menu.add_radiobutton(label="Pencil",
                  variable=self.drawing_tool,
                  value="pencil")
tool_menu.add_radiobutton(label="Line",
                  variable=self.drawing_tool,
                  value="line")
tool_menu.add_radiobutton(label="Arc",
```

```
                    variable=self.drawing_tool,
                    value="arc")
    tool_menu.add_radiobutton(label="Oval",
                    variable=self.drawing_tool,
                    value="oval")
    tool_menu.add_radiobutton(label="Rectangle",
                    variable=self.drawing_tool,
                    value="rectangle")
    tool_menu.add_radiobutton(label="Text",
                    variable=self.drawing_tool,
                    value="text")
    the_menu.add_cascade(label="Tool", menu=tool_menu)

    # ---- COLOR MENU ----

    color_menu = Menu(the_menu, tearoff=0)
    color_menu.add_command(label="Fill", command=self.pick_fill)
    color_menu.add_command(label="Stroke", command=self.pick_stroke)
    stroke_width_submenu = Menu(color_menu)
    stroke_width_submenu.add_radiobutton(label="2",
                                variable=self.stroke_size,
                                value=2)
    stroke_width_submenu.add_radiobutton(label="3",
                                  variable=self.stroke_size,
                                  value=3)
    stroke_width_submenu.add_radiobutton(label="4",
                                  variable=self.stroke_size,
                                  value=4)
    stroke_width_submenu.add_radiobutton(label="5",
                                  variable=self.stroke_size,
                                  value=5)
    color_menu.add_cascade(label="Stroke Size",
                    menu=stroke_width_submenu)
    the_menu.add_cascade(label="Color", menu=color_menu)

    # Display the menu bar
    root.config(menu=the_menu)

# ---------- CATCH MOUSE UP ----------

def left_but_down(self, event=None):
    self.left_but = "down"

    # Set x & y when mouse is clicked
    self.x1_line_pt = event.x
    self.y1_line_pt = event.y

# ---------- CATCH MOUSE UP ----------

def left_but_up(self, event=None):
    self.left_but = "up"

    # Reset the line
    self.x_pos = None
```

```
        self.y_pos = None

        # Set x & y when mouse is released
        self.x2_line_pt = event.x
        self.y2_line_pt = event.y

        # If mouse is released and line tool is selected
        # draw the line
        if self.drawing_tool.get() == "line":
            self.line_draw(event)
        elif self.drawing_tool.get() == "arc":
            self.arc_draw(event)
        elif self.drawing_tool.get() == "oval":
            self.oval_draw(event)
        elif self.drawing_tool.get() == "rectangle":
            self.rectangle_draw(event)
        elif self.drawing_tool.get() == "text":
            self.text_draw(event)

    # ---------- CATCH MOUSE MOVEMENT ----------

    def motion(self, event=None):
        if self.drawing_tool.get() == "pencil":
            self.pencil_draw(event)

    # ---------- DRAW PENCIL ----------

    def pencil_draw(self, event=None):
        if self.left_but == "down":

            # Make sure x and y have a value
            if self.x_pos is not None and self.y_pos is not None:
                event.widget.create_line(self.x_pos, self.y_pos, event.x, event.y, smooth=TRUE,
fill=self.stroke_color.get(), width=self.stroke_size.get())

                # NEW Draw to PIL image for saving
                self.draw.line([(self.x_pos, self.y_pos),
                                (event.x, event.y)],
                               fill=self.stroke_color.get(),
                               width=self.stroke_size.get())

            self.x_pos = event.x
            self.y_pos = event.y

    def line_draw(self, event=None):

        # Shortcut way to check if none of these values contain None
        if None not in (self.x1_line_pt, self.y1_line_pt, self.x2_line_pt, self.y2_line_pt):
            event.widget.create_line(self.x1_line_pt, self.y1_line_pt, self.x2_line_pt, self.y2_line_pt,
smooth=TRUE, fill=self.stroke_color.get())

            # NEW Draw to PIL image for saving
            self.draw.line([(self.x1_line_pt, self.y1_line_pt),
                            (self.x2_line_pt, self.y2_line_pt)],
```

```
                    fill=self.stroke_color.get(),
                    width=self.stroke_size.get())

def arc_draw(self, event=None):

    # Shortcut way to check if none of these values contain None
    if None not in (self.x1_line_pt, self.y1_line_pt, self.x2_line_pt, self.y2_line_pt):
        coords = self.x1_line_pt, self.y1_line_pt, self.x2_line_pt, self.y2_line_pt

        event.widget.create_arc(coords, start=0, extent=150,
                                style=ARC, fill=self.stroke_color.get())

        # NEW Draw to PIL image for saving
        self.draw.arc([(self.x1_line_pt, self.y1_line_pt),
                       (self.x2_line_pt, self.y2_line_pt)],
                      start=0, end=150,
                       fill=self.stroke_color.get())

def oval_draw(self, event=None):
    if None not in (self.x1_line_pt, self.y1_line_pt, self.x2_line_pt, self.y2_line_pt):
        # fill : Color option names are here http://wiki.tcl.tk/37701
        # outline : border color
        # width : width of border in pixels

        event.widget.create_oval(self.x1_line_pt, self.y1_line_pt, self.x2_line_pt, self.y2_line_pt,
                                 fill=self.fill_color.get(),
                                 outline=self.stroke_color.get(),
                                 width=self.stroke_size.get())

        # NEW Draw oval to PIL image
        self.draw.ellipse([(self.x1_line_pt, self.y1_line_pt),
                           (self.x2_line_pt, self.y2_line_pt)],
                          fill=self.fill_color.get(),
                          outline=self.stroke_color.get())

def rectangle_draw(self, event=None):
    if None not in (self.x1_line_pt, self.y1_line_pt, self.x2_line_pt, self.y2_line_pt):
        # fill : Color option names are here http://wiki.tcl.tk/37701
        # outline : border color
        # width : width of border in pixels

        event.widget.create_rectangle(self.x1_line_pt, self.y1_line_pt,
                                      self.x2_line_pt, self.y2_line_pt,
                                      fill=self.fill_color.get(),
                                      outline=self.stroke_color.get(),
                                      width=self.stroke_size.get())

        # NEW Draw rectangle to PIL image
        self.draw.rectangle([(self.x1_line_pt, self.y1_line_pt),
                             (self.x2_line_pt, self.y2_line_pt),
                             ], fill=self.fill_color.get(),
                                      outline=self.stroke_color.get())

def text_draw(self, event=None):
```

```
    if None not in (self.x1_line_pt, self.y1_line_pt):
        # Show all fonts available
        # print(tkinter.font.families())

        text_font = tkinter.font.Font(family=self.text_font.get(),
                                      size=self.text_size.get(), weight=self.bold_text.get(),
                                      slant=self.italic_text.get())
        # Get the text the user wants to enter
        user_text = simpledialog.askstring("Input",
                                           "Enter Text", parent=root)
        if user_text is not None:
            event.widget.create_text(self.x1_line_pt, self.y1_line_pt,
                                     fill=self.fill_color.get(),
                                     font=text_font,
                                     text=user_text)

            # NEW While you can save text to a PIL file you have
            # to put either True Type Fonts, or PIL fonts in the
            # directory or provide specific paths based on your
            # computer. You could also convert from fonts you have
            # to PIL fonts. This is beyond this tutorial so I leave
            # that task to you for homework

def pick_fill(self, event=None):
    fill_color = askcolor(title='Pick Fill color')
    if None not in fill_color:
        self.fill_color.set(fill_color[1])
        print("Color ", self.fill_color.get())

def pick_stroke(self, event=None):
    stroke_color = askcolor(title='Pick Stroke color')
    if None not in stroke_color:
        self.stroke_color.set(stroke_color[1])

def __init__(self, root):

    self.drawing_area.pack()

    self.text_font.set("Times")
    self.text_size.set(20)
    self.bold_text.set('normal')
    self.italic_text.set('roman')

    self.drawing_tool.set("pencil")
    self.stroke_size.set(3)
    self.fill_color.set('#000000')
    self.stroke_color.set('#000000')

    self.make_menu_bar()

    # Set focus for catching events to the canvas
    self.drawing_area.focus_force()
    self.drawing_area.bind("<Motion>", self.motion)
    self.drawing_area.bind("<ButtonPress-1>", self.left_but_down)
```

```
        self.drawing_area.bind("<ButtonRelease-1>", self.left_but_up)


paint_app = PaintApp(root)
root.mainloop()
```