```
# In this tutorial I finish the text editor by covering toolbars, fix some previous bugs, show how
to work with images and more.

from tkinter import *
import tkinter.filedialog
import ast

# NEW Type from Pillow import Image, ImageTkClick
# and let PyCharm install Pillow
# Image allows you to load images from files
# ImageTk provides ways to create and modify images
from PIL import Image, ImageTk
# --- END NEW ---


class TextEditor:
    # NEW
    # Used for font size, type
    font_size = 28
    font_type = "Georgia"
    # END NEW

    # Quits the TkInter app when called
    @staticmethod
    def quit_app(event=None):
        root.quit()

    def open_file(self, event=None):
        # Open dialog and get chosen file
        txt_file = tkinter.filedialog.askopenfilename(parent=root)
        # If the file exists
        if txt_file:
            self.text_area.delete(1.0, END)

            # Holds list of tuples
            file_list = []

            # Open file and put text in the text widget
            with open(txt_file) as _file:
                # self.text_area.insert(1.0, _file.read())

                # Processes the list of tuples into a list
                file_list = list(ast.literal_eval(_file.read()))
                print(file_list)

                # Search for text in the list and put it in the right
                # index position
                for data in file_list:
                    if data[0] == "text":
                        self.text_area.insert(data[2], data[1])

                # Cycle through the list looking for tagon, but ignore sel
                i = 0
                while i < len(file_list):
```

```
            # NEW Remove the sel option
            if file_list[i][0] == "tagon":
                # Get the styling tag
                styling = file_list[i][1]
                # Get the index where styling begins
                start_of_style = file_list[i][2]
                # Used to get the index where styling ends
                # but set as end of file by default
                end_of_style = END
                # Make sure I'm not searching beyond the end
                # of the list

                # NEW Change the step to 2 because
                # we got rid of sel and mark
                if (i+2) < len(file_list):
                    # If not find the end index
                    end_of_style = file_list[i+2][2]

                # Add styling provided along with the start
                # and ending index
                self.text_area.tag_add(styling,
                                       start_of_style,
                                       end_of_style)
            i += 1

        # Update the text widget
        root.update_idletasks()

def save_file(self, event=None):

    # Opens the save as dialog box
    file = tkinter.filedialog.asksaveasfile(mode='w')
    if file is not None:
        # Get list of tuples
        text_area_list = self.text_area.dump('1.0', END + '-1c')

        # --- NEW ---
        # Remove all tuples if 'sel' or 'mark' is in it
        text_area_list = [i for i in text_area_list if i[1] != 'sel' and i[0] != 'mark']
        # --- END NEW ---

        # Write list of tuples to file
        file.write(' '.join('("{}", "{}", "{}")'.format(x[0],
                    x[1], x[2])
                    for x in text_area_list))
        file.close()

def make_bold(self):
    self.text_area.tag_add("bt", "sel.first", "sel.last")

# NEW Make selected text italic
def make_italic(self):
    self.text_area.tag_add("ital", "sel.first", "sel.last")
```

```
# --- END NEW ---

def __init__(self, root):

    self.text_to_write = ""

    # Define title for the app
    root.title("Text Editor")

    # Defines the width and height of the window
    root.geometry("600x550")

    frame = Frame(root, width=600, height=550)

    # Create the scrollbar
    scrollbar = Scrollbar(frame)

    # yscrollcommand connects the scroll bar to the text
    # area
    self.text_area = Text(frame, width=600, height=550,
                          yscrollcommand=scrollbar.set,
                          padx=10, pady=10, font=(self.font_type, self.font_size))

    # Call yview when the scrollbar is moved
    scrollbar.config(command=self.text_area.yview)

    # Put scroll bar on the right and fill in the Y direction
    scrollbar.pack(side="right", fill="y")

    # Pack on the left and fill available space
    self.text_area.pack(side="left", fill="both", expand=True)

    # NEW Moved this below the toolbar
    # frame.pack()

    # ----- FILE MENU CREATION -----

    # 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
    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)
```

```
# ----- EDIT MENU CREATION -----

edit_menu = Menu(the_menu, tearoff=0)
edit_menu.add_command(label="Bold", command=self.make_bold)

# --- NEW ---
# Add italic option to menu bar
edit_menu.add_command(label="Italic", command=self.make_italic)
# --- END NEW ---

the_menu.add_cascade(label="Edit", menu=edit_menu)

self.text_area.tag_config("bt", font=(self.font_type, self.font_size, "bold"))

# --- New Configure italic ---
self.text_area.tag_config("ital", font=(self.font_type, self.font_size, "italic"))

# Create our tool bar by creating a frame, defining the border
# width, and relief=RAISED draws a line under the toolbar
toolbar = Frame(root, bd=1, relief=RAISED)

# Get our tool bar images
open_img = Image.open("open.png")
save_img = Image.open("save.png")
copy_img = Image.open("copy.png")
cut_img = Image.open("cut.png")
paste_img = Image.open("paste.png")
bold_img = Image.open("bold.png")
italic_img = Image.open("italic.png")

# Create TkInter image to be used in buttons
open_icon = ImageTk.PhotoImage(open_img)
save_icon = ImageTk.PhotoImage(save_img)
copy_icon = ImageTk.PhotoImage(copy_img)
cut_icon = ImageTk.PhotoImage(cut_img)
paste_icon = ImageTk.PhotoImage(paste_img)
bold_icon = ImageTk.PhotoImage(bold_img)
italic_icon = ImageTk.PhotoImage(italic_img)

# Create buttons for the toolbar
open_button = Button(toolbar, image=open_icon,
                     command=self.open_file)
open_button.image = open_icon
save_button = Button(toolbar, image=save_icon,
                     command=self.save_file)
save_button.image = save_icon
copy_button = Button(toolbar, image=copy_icon,
                     command=lambda: root.focus_get().event_generate('<<Copy>>'))
copy_button.image = copy_icon
cut_button = Button(toolbar, image=cut_icon,
                    command=lambda: root.focus_get().event_generate('<<Cut>>'))
cut_button.image = cut_icon
paste_button = Button(toolbar, image=paste_icon,
                      command=lambda: root.focus_get().event_generate('<<Paste>>'))
```

```
        paste_button.image = paste_icon
        bold_button = Button(toolbar, image=bold_icon,
                             command=self.make_bold)
        bold_button.image = bold_icon
        italic_button = Button(toolbar, image=italic_icon,
                               command=self.make_italic)
        italic_button.image = italic_icon

        # Place buttons in the interface
        open_button.pack(side=LEFT, padx=2, pady=2)
        save_button.pack(side=LEFT, padx=2, pady=2)
        copy_button.pack(side=LEFT, padx=2, pady=2)
        cut_button.pack(side=LEFT, padx=2, pady=2)
        paste_button.pack(side=LEFT, padx=2, pady=2)
        bold_button.pack(side=LEFT, padx=2, pady=2)
        italic_button.pack(side=LEFT, padx=2, pady=2)

        # Put toolbar at the top of the window
        # and fill horizontally
        toolbar.pack(side=TOP, fill=X)

        # Moved from the top
        frame.pack()

        # --- END NEW ---

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


root = Tk()

# Create the menu object
the_menu = Menu(root)

text_editor = TextEditor(root)

root.mainloop()
```