How to Create a Image Editor App Using Python

As human beings in general, many times we won't be impressed by the outcome of our Images and so adding some retouching and features to make it bright and professional becomes necessary. When it comes to editing Images we usually opt for different professional tools such as Imageshop, Inshot etc… Even though these tools will still have the features to adjust our Images, getting them fully may require paying for the features.

However, if you are a programmer, you can always utilize your skills by developing tools that can serve the same purpose as other software's you need, like in this case a Image editor app. Doing so will make it much easier to edit your Images in Python.

We will be using Python to demonstrate how to achieve most of the Image editing features which can help you accomplish the same basic tasks as any other software available out there.

How to Edit an Image

We live in a society where as human beings we want our Images to look more impressive than our personalities, where our social media timelines are filled with beautiful Images. In order to achieve this understanding the basics of how to edit an image becomes very important.
When it comes to editing Images, some of the key features many people sort to go after include:

- Cropping
- Adding beauty filters
- Resizing the image
- Adjusting contrast

- Adding the blur effect

The good news is, you can achieve all this with Python and with the help of some image processing libraries.

Python Libraries Needed

In order for us to have our application up and running, we will need to make use of some Python libraries that will make it easier develop the application.
They include:
Pillow – It is used to manipulate, read, and save images. It also adds the image processing capabilities to the Python interpreter.
Tkinter – Used to create the GUI of our app.

Creating a Image-Editor App

- Now, before we get started with writing the code for our app, we need to make sure that we have all necessary tools for this task set in place.
- That includes confirming that you have Python installed in your local machine and also that you have access to a code editor of your choice.
- To check whether Python is installed, No matter the operating system you are using, the command below works across both Windows, Mac and Linux systems:

```
python --version
```

- Having confirmed that, we can move ahead and begin with writing out code. Earlier we listed some of the libraries that we will need to create our app, now let's begin by first installing them into our system.

HARDWARE AND SOFTWARE REQUIRED

Hardware:-

- A modern operating system.
- 64 Bit CPU.
- The Device should have minimum 2 GB RAM.
- Internet Connectivity

Software:-

- Operating System should be Windows 7 or higher.
- Python 3.11.0.
- PyCharm

- **Step 1:** Installing the Libraries
- **Step 2:** Import Required Modules
- **Step 3:** Create a Tkinter Window
- **Step 4:** Loading an Image
- **Step 5**.1The Blur Effect
- **Step 5.2:** The Brightness Effect
- **Step 5.3:** The Contrast Effect
- **Step 5.4:** The Rotation Effect
- **Step 5.5:** The Flip Effect
- **Step 5.6:** Adding Borders
- **Step 5.7:** Filter effect
- **Step 5.8**:Button effect
- **Step 5.9:**Color effect
- **Step 5.10** Crop Image
- **Step 6:** Saving Edited Image
- **Step 7:** Creating Interactive Labels and Buttons
- **Step 7.1:** Buttons and Create canvas to display image
- **Step 8:** Run the App
- The Output
- Conclusion

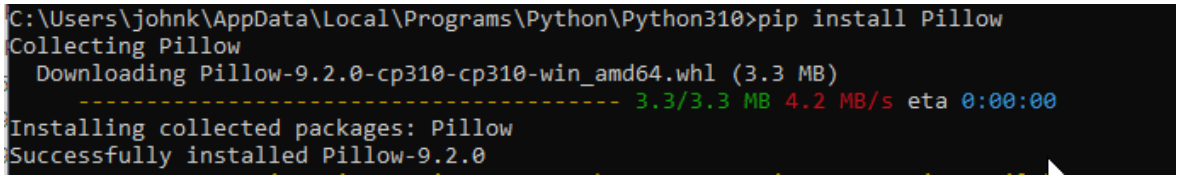
Step 1: Installing the Libraries

- We need to install the pillow library.
- We will use PIP to run the installation command.

```
pip install pillow
```

- Similarly to install tkinter
- we will make use of PIP by executing the command below:

```
pip install tk
```



Step 2: Import Required Modules

- In order for us to use the modules that we have installed, we will need to import them, specifically some packages that align with our project. The code below allows us to do this. Remember, all the code is written in a python file, make sure you already have one created:

```
1. from tkinter import *
2. from tkinter import ttk, IntVar
3. from tkinter import filedialog
4. from tkinter.filedialog import askopenfilename,
   asksaveasfilename
5. from PIL import Image, ImageTk, ImageFilter,
   ImageEnhance, ImageOps,ImageDraw, ImageFont
6. import os
7. import tkinter as tk
```

- The above code adds the modules directly into our project by making use of the import statement.

Step 3: Create a Tkinter Window

This is the point where we get to create our interactive interface. It's going to act like the layout of our app housing all components needed including different functional buttons for interactivity. With the code below we are able to set the window title and also the dimensions. (*feel free to try out different dimensions you are comfortable with*).

```
1. root = Tk()
2. root.title('Image Editor App With GUI")
3. root.geometry("640x640")
4. opened_image = ""
5. image_height=0
6. image_width=0
7. img18=""
8. img19=""
9. bg=PhotoImage(file="back.png")
10.        background_label= Label(root, image=bg)
11.        background_label.place(x=0, y=0, relwidth=1,
   relheight=1)
```

Here , line 9 define the image
line 10 pass the image to background
line 11 set the place of a image

Step 4: Loading an Image

First we begin by creating a function
def select(), which will allow us to pass in the code that will be used to load an image for editing.

```
1. def select():
2.     global img_path, img, image_height, image_width
3.     img_path =
   filedialog.askopenfilename(initialdir=os.getcwd())
```

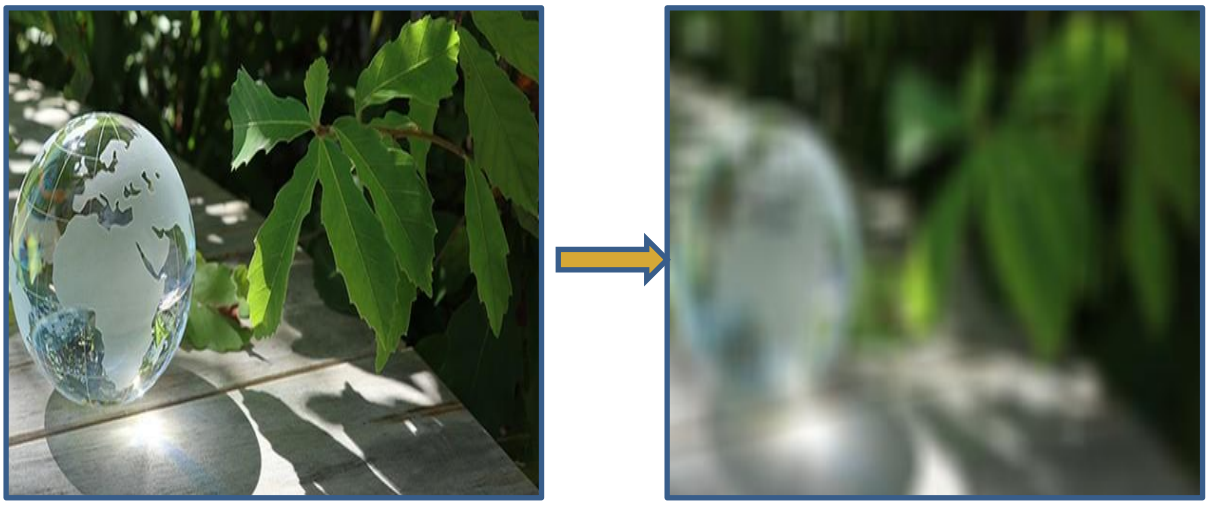
```
4.    img = Image.open(img_path)
5.    img.thumbnail((350, 350))
6.    img1 = ImageTk.PhotoImage(img)
7.    canvas2.create_image(300, 210, image=img1)
8.    canvas2.image = img1
9.    if img:
10.           opened_image =
   ImageTk.PhotoImage(Image.open(img_path))
11.           image_width = opened_image.width()
12.           image_height = opened_image.height()
```

- First line we have the function that describes what we will be handling in this code block, that is selecting an image.
- Second line, three global variables are declared which will assist us in defining different attributes related to our image.
- Third line, the code allows us to get the path of the image that we intend to use.
- Line four, opens the selected image and sets it ready for editing.
- Line five, using the img.thumbnail() :-the image is converted into a thumbnail version of itself.
- Line Six, The Tkinter PhotoImage widget supports different file formats. In order for it to support other file formats such as JPG or JPEG, the use of Pillow is necessary to enable the conversion that is understood by PhotoImage widget. That's what line 6 in our code block does.
- Line seven and eight, The create_image, is used to create an image object on the canvas, this is because it does not accept an image directly it uses an object created by the PhotoImage() method.

Step 5.1: The Blur Effect

When talking about image blurring, it's basically reducing the sharpness of the image. There isn't much to add under this section except creating a function similar to the precious one only this time round we will name it
def blur(event), and embed the option within a
for statement, add a one liner code which will add the blur effect widget
imgg = img.filter(ImageFilter.BoxBlur(m)).

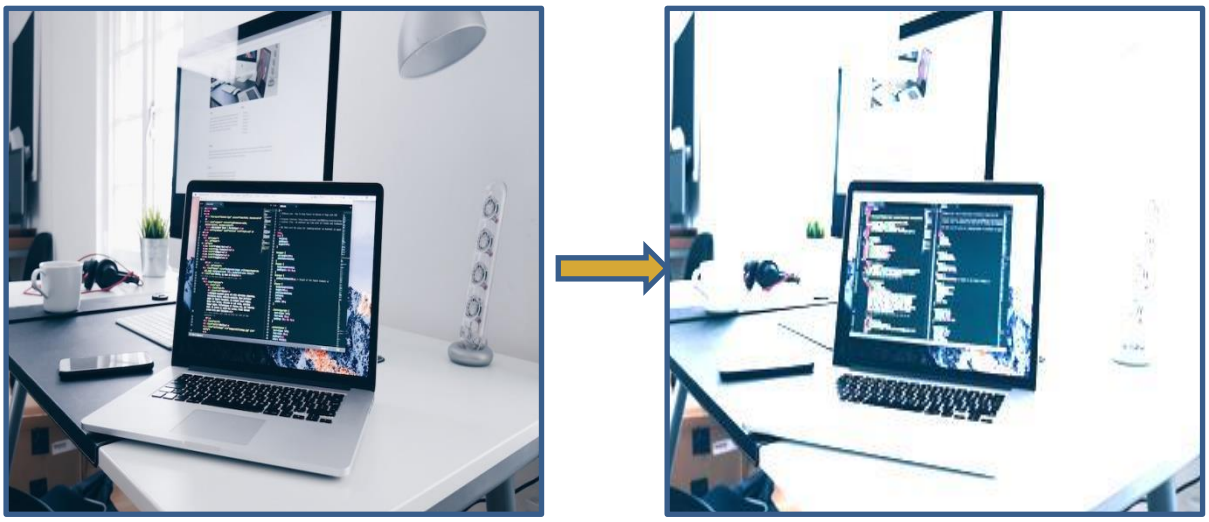
```
1. def blur(event):
2.     global img_path, img1, imgg,img
3.     for m in range(0, v1.get()+1):
4.         img = Image.open(img_path)
5.         img.thumbnail((350, 350))
6.         imgg = img.filter(ImageFilter.BoxBlur(m))
7.         img1 = ImageTk.PhotoImage(imgg)
8.         canvas2.create_image(300, 210, image=img1)
9.         canvas2.image = img1
```



Step 5.2: The Brightness Effect

This code block is used to control the brightness of an image.

```
1. def brightness(event):
2.     global img_path, img2, img3
3.     for m in range(0, v2.get()+1):
4.         img = Image.open(img_path)
5.         img.thumbnail((350, 350))
6.         imgg = ImageEnhance.Brightness(img)
7.         img2 = imgg.enhance(m)
8.         img3 = ImageTk.PhotoImage(img2)
9.         canvas2.create_image(300, 210, image=img3)
10.                canvas2.image = img3
```

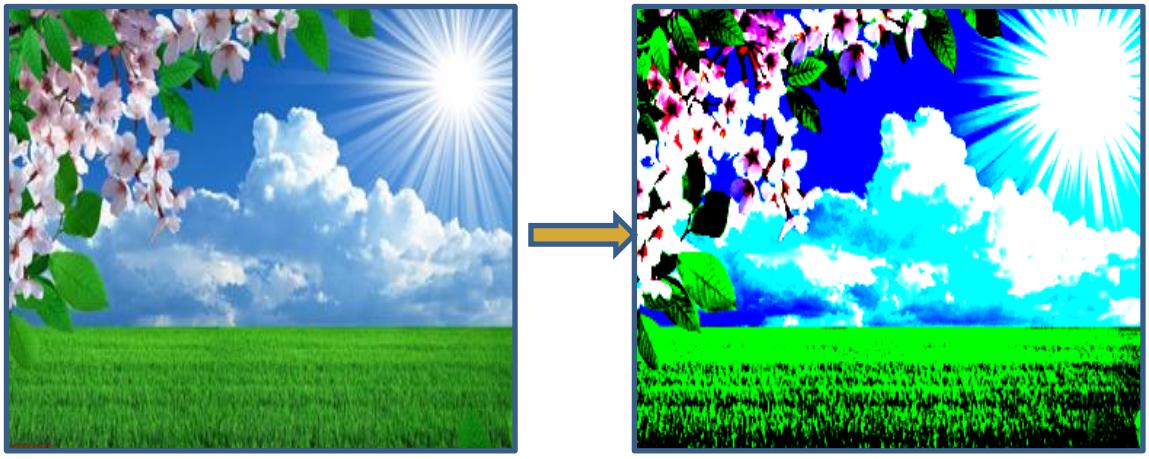


Step 5.3: The Contrast Effect

This code block is used to control the contrast of an image, similar to the contrast control on a TV set. An enhancement factor of 0.0 gives a solid grey image. A factor of 1.0 gives the original image.

```
1. def contrast(event):
2. global img_path, img4, img5
3. for m in range(0, v3.get()+1):
4. img = Image.open(img_path)
```

```
5. img.thumbnail((350, 350))
6. imgg = ImageEnhance.Contrast(img)
7. img4 = imgg.enhance(m)
8. img5 = ImageTk.PhotoImage(img4)
9. canvas2.create_image(300, 210, image=img5)
10.        canvas2.image = img5
```



Step 5.4: The Rotation Effect

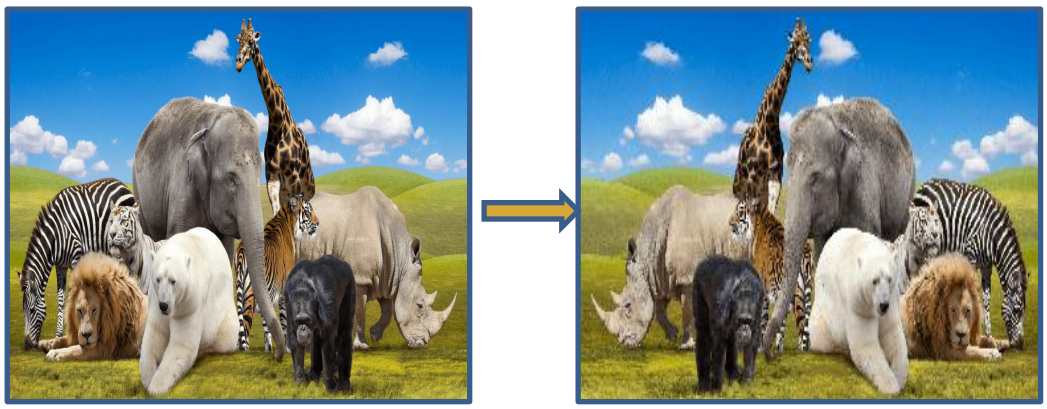
In this code block, the newly added line of code is
img.rotate(int(rotate_combo.get()))
which is used to rotate the image either from left to right or top to bottom.

```
1. def rotate(event):
2.     global img_path, img6, img7
3.     img = Image.open(img_path)
4.     img.thumbnail((350, 350))
5.     img6 = img.rotate(int(rotate_combo.get()))
6.     img7 = ImageTk.PhotoImage(img6)
7.     canvas2.create_image(300, 210, image=img7)
8.     canvas2.image = img7
```

Step 5.5: The Flip Effect

Image flipping is basically a time-reversed image on a horizontal axis. You can either flip your image in different directions that is either from top to bottom or left to right

```
1. def flip(event):
2.     global img_path, img8, img9
3.     img = Image.open(img_path)
4.     img.thumbnail((350, 350))
5.     if flip_combo.get() == "FLIP LEFT TO RIGHT":
6.        img8 = img.transpose(Image.FLIP_LEFT_RIGHT)
7.     elif flip_combo.get() == "FLIP TOP TO BOTTOM":
8.         img8 = img.transpose(Image.FLIP_TOP_BOTTOM)
9.     img9 = ImageTk.PhotoImage(img8)
10.          canvas2.create_image(300, 210, image=img9)
11.          canvas2.image = img9
```



Step 5.6: Adding Borders

The ImageOps module contains a number of ready made image processing operations,
it adds a border to the image. It's syntax is as shown below:
PIL.ImageOps.expand(image, border = 0, fill = 0)
In our case, our borders will be set as follows:

```
1. def border(event):
2. global img_path, img10, img11
3. img = Image.open(img_path)
4. img.thumbnail((350, 350))
5. img10 = ImageOps.expand(img,
   border=int(border_combo.get()),   fill=95)
6. img11 = ImageTk.PhotoImage(img10)
7. canvas2.create_image(300, 210, image=img11)
8. canvas2.image = img11
```

Step 5.7: Filter effect

The ImageFilter module contains definitions for a pre-defined set of filters, which we used with Image.filter() method. These filters are used to change the looks and feel of the image.
pillow library provides below mentioned set of predefined image enhancement filters.

BLUR,CONTOUR,DETAIL,EDGE_ENHANCE,EDGE_ENHANCE_MORE,EMBOSS,FIND_EDGES,SHARPEN,SMOOTH,SMOOTH_MORE

```
def flip8(event):
    global img_path, img14, img15
    img = Image.open(img_path)
    img.thumbnail((350, 350))
    if flip_combo5.get() == "COUNTER":
       img14 = img.filter(ImageFilter.CONTOUR)
    elif flip_combo5.get() == "SMOOTH_MORE":
        img14 = img.filter(ImageFilter.SMOOTH_MORE)
    elif flip_combo5.get() == "EMBOSS":
            img14 = img.filter(ImageFilter.EMBOSS)
        elif flip_combo5.get() == "EDGE_ENHANCE":
            img14 = img.filter(ImageFilter.EDGE_ENHANCE)
        elif flip_combo5.get() == "SHARPEN":
            img14 = img.filter(ImageFilter.SHARPEN)
        elif flip_combo5.get() == "DETAIL":
            img14 = img.filter(ImageFilter.DETAIL)
        img15 = ImageTk.PhotoImage(img14)
        canvas2.create_image(300, 210, image=img15)
        canvas2.image = img15
```

Step 5.8:Button effect

To change the color of the Button while hovering on it, we have to bind the <Enter> and <Leave> events. For each event, we will configure the button property such as background color, foreground color, etc.

```
1. def onButton(e):
2.     btn1['bg']='blue'
3. def onButton1(e):
4.     btn2['bg']='blue'
5. def onButton2(e):
6.     btn3['bg']='blue'
7. def onButton3(e):
8.    btn4['bg'] = 'blue'
9. def leaveButton(e):
10.           btn1['bg']='black'
11.       def leaveButton1(e):
12.           btn2['bg']='black'
13.       def leaveButton2(e):
14.           btn3['bg']='black'
15.       def leaveButton3(e):
16.          btn4['bg'] = 'black'
```

Step 5.9:Color effect

```
1. .def colors(event):
2. global img_path, img16, img17
3. img = Image.open(img_path)
4. img.thumbnail((350, 350))
5. if color_combo8.get() == "DARK":
6.    for x in range(img.size range(img.size[0]):
7.     for y in [1]):
```

```
8.          r, g, b = img.getpixel((x, y))
9.        img.putpixel((x, y), (r//2, g//2, b//2))
10.       elif color_combo8.get() == "LIGHT BLUE":
11.           for x in range(img.size[0]):
12.             for y in range(img.size[1]):
13.               r, g, b = img.getpixel((x, y))
14.              img.putpixel((x, y), (0, g, b))
15.       elif color_combo8.get() == "PINK":
16.           for x in range(img.size[0]):
17.             for y in range(img.size[1]):
18.               r, g, b = img.getpixel((x, y))
19.              img.putpixel((x,y),(r,0,b))
20.       elif color_combo8.get() == "YELLOW":
21.                                                   for x in
    range(img.size[0]):
22.               for y in range(img.size[1]):
23.                 r, g, b = img.getpixel((x, y))
24.                img.putpixel((x,y),(r,g,0))
25.       img16 = ImageTk.PhotoImage(img)
26.       canvas2.create_image(300, 210, image=img16)
27.       canvas2.image = img16
```




Original **light blue** **yellow** **pink**

Step 5.10 Crop Image :-

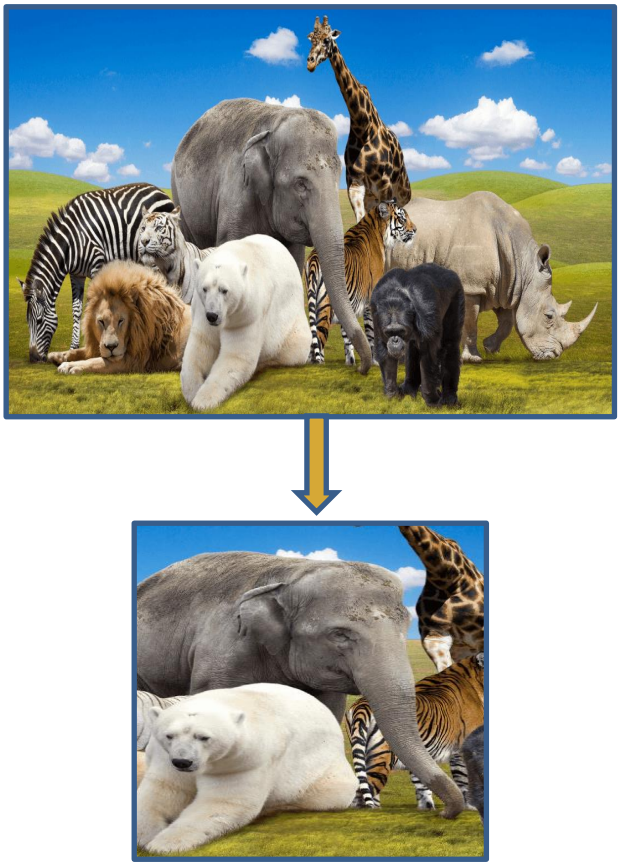
The crop() function of the image class in Pillow requires the portion to be cropped as rectangle.

```
1. def crop_image(left,top,right,bottom):
2. global opened_image,edited_image,img21
3. img= Image.open(img_path)
4. edited_image = img.crop((left,top,image_width-
   right,image_height-bottom))
5. img21 = ImageTk.PhotoImage(edited_image)
6. canvas2.create_image(300, 210, image=img21)
7. canvas2.image = img21
```

- Controls window left, top, right ,bottom scale of Crop function

```
1. def controls_window(w, h):
2. global img_path,image_height,image_width
3. controls = tk.Toplevel(root)
4. controls.geometry("300x300")
5. left_lb = tk.Label(controls, text="left")
6. left_lb.pack(anchor=tk.W, pady=5)
7. left_scale = tk.Scale(controls, from_=0, to=w,
   orient=tk.HORIZONTAL,
            command=lambda x:crop_image(left_scale.get(),
top_scale.get(),
                                        right_scale.get(),
bottom_scale.get())
                                    )
8. left_scale.pack(anchor=tk.W, fill=tk.X)

9. right_lb = tk.Label(controls, text="right")
10.       right_lb.pack(anchor=tk.W, pady=5)
11.       right_scale = tk.Scale(controls, from_=0,
   to=w,orient=tk.HORIZONTAL,
```

```
            command=lambda
x:crop_image(left_scale.get(),top_scale.get(),

right_scale.get(),bottom_scale.get())
                                    )
12.         right_scale.pack(anchor=tk.W, fill=tk.X)

13.         top_to = tk.Label(controls, text="top")
14.         top_to.pack(anchor=tk.W, pady=5)
15.         top_scale = tk.Scale(controls, from_=0, to=w,
    orient=tk.HORIZONTAL,
                command=lambda x:crop_image(left_scale.get(),
top_scale.get(),
                                                right_scale.get(),
bottom_scale.get())
                                    )
16.         top_scale.pack(anchor=tk.W, fill=tk.X)

17.         bottom_to = tk.Label(controls, text="bottom")
18.         bottom_to.pack(anchor=tk.W, pady=5)
19.         bottom_scale=tk.Scale(controls,from_=0,to=w,orient=tk.H
    ORIZONTAL,
20.         command=lambda x:crop_image(left_scale.get(),
    top_scale.get(),
                                        right_scale.get(),
bottom_scale.get())
                                    )
21.         bottom_scale.pack(anchor=tk.W, fill=tk.X)
```



Step 6: Saving Edited Image

Saving an image in Python is easier. The save function here will save the edited image in the same directory as the Python file.

```
1. def save():
2. global img_path, imgg, img1, img2, img3, img4, img5, img6,
   img7, img8, img9, img10, img11
3. ext = img_path.split(".")[-1]
4. file = asksaveasfilename(defaultextension=f".{ext}",
   filetypes=[(
5. "All Files", "*.*"), ("PNG file", "*.png"), ("jpg file", "*.jpg")])
6. if file:
7. if canvas2.image == img1:
        imgg.save(file)
8. elif canvas2.image == img3:
```

```
        img2.save(file)
9. elif canvas2.image == img5:
        img4.save(file)
10.         elif canvas2.image == img7:
        img6.save(file)
11.         elif canvas2.image == img9:
          img8.save(file)
12.         elif canvas2.image == img11:
        img10.save(file)
13.         elif canvas2.image == img15:
        img14.save(file)
elif canvas2
14.         .image == img16:
        img.save(file)
15.         elif canvas2.image == img21:
      edited_image.save(file)
```

Step 7: Creating Interactive Labels and Buttons

This particular step is where we get to design the interface of our app that includes adding the canvas to display the image and buttons too, that is by setting the fonts, color, text position etc…

```
1. blurr =ttk.Label(root, text="Blur:", font=("TIMES-ROMAN 12
   bold"))
2. blurr.place(x=11, y=8)
3. v1 = IntVar()
4. scale1 = ttk.Scale(root, from_=0, to=10, variable=v1,
5. orient=HORIZONTAL, command=blur)
6. scale1.place(x=125, y=10, height=23, width=100)
7. bright = ttk.Label(root, text="Brightness:", font=("TIMES-
   ROMAN 12 bold"))
8. bright.place(x=11, y=50)
```

```
9. v2 = IntVar()
10.        scale2 = ttk.Scale(root, from_=0, to=10, variable=v2,
11.        orient=HORIZONTAL, command=brightness)
12.        scale2.place(x=125, y=50, height=23, width=100)

13.        contrast = ttk.Label(root, text="Contrast:", font=("TIMES-
    ROMAN 12 bold"),)
14.        contrast.place(x=11, y=92)
15.        v3 = IntVar()
16.        scale3 = ttk.Scale(root, from_=0, to=10, variable=v3,
17.        orient=HORIZONTAL, command=contrast)
18.        scale3.place(x=125, y=92, height=23, width=100)

19.        rotate1 = ttk.Label(root, text="Rotate:",font=("TIMES-
    ROMAN 12 bold"))
20.        rotate1.place(x=250, y=8)
21.        values = [0, 90, 180, 270, 360]
22.        rotate_combo = ttk.Combobox(root, value=values,
    font=('TIMES-ROMAN 10 bold'))
23.        rotate_combo.place(x=330, y=10,width=100, height=23)
24.        rotate_combo.bind("<<ComboboxSelected>>", rotate)
```

```
25.        flip1 = ttk.Label(root, text="Flip:", font=("TIMES-
    ROMAN 12 bold"))
26.        flip1.place(x=250, y=50)
27.        values1 = ["FLIP LEFT TO RIGHT", "FLIP TOP TO
    BOTTOM"]
28.        flip_combo = ttk.Combobox(root, values=values1,
    font=('TIMES-ROMAN 10 bold'))
29.        flip_combo.place(x=330, y=50,width=100,height=23)
30.        flip_combo.bind("<<ComboboxSelected>>", flip)
```

```
31.     flip5 = ttk.Label(root, text="Filter:", font=("TIMES-
    ROMAN 12 bold"))
32.     flip5.place(x=250, y=92)
33.     values5 = ["COUNTER", "SMOOTH_MORE",
    "EMBOSS" , "EDGE_ENHANCE", "SHARPEN", "DETAIL" ]
34.     flip_combo5 = ttk.Combobox(root, values=values5,
    font=('TIMES-ROMAN 10 bold'))
35.     flip_combo5.place(x=330, y=92,width=100,height=23)
36.     flip_combo5.bind("<<ComboboxSelected>>", flip8)



37.     border1 = ttk.Label(root, text="Border:", font=("TIMES-
    ROMAN 12 bold"))
38.     border1.place(x=450, y=8)
39.     values2 = [i for i in range(10, 45, 5)]
40.     border_combo = ttk.Combobox(root, values=values2,
    font=("TIMES-ROMAN 10 bold"))
41.     border_combo.place(x=530, y=10,width=100,height=23)
42.     border_combo.bind("<<ComboboxSelected>>", border)

43.     color9 = ttk.Label(root, text="Colors:", font=("TIMES-
    ROMAN 12 bold"))
44.     color9.place(x=450, y=50)
45.     values8 = ["DARK", "LIGHT
    BLUE","PINK","YELLOW"]
46.     color_combo8 =ttk.Combobox(root, values=values8,
    font=('TIMES-ROMAN 10 bold'))
47.     color_combo8.place(x=530, y=50,width=100,height=23)
48.     color_combo8.bind("<<ComboboxSelected>>", colors)
```

Step 7.1: Buttons and Create canvas to display image

Tkinter canvas is the most versatile widget in the Tkinter library. It is used to create images, shapes, arcs, animating objects, and many more other works.
In order to work and process the images, Python supports Pillow Package or PIL.

```
canvas2 = Canvas(root, width="600", height="420", relief=RIDGE,bd=2)
canvas2.place(x=15, y=150)
```

```
# create buttons
btn1 = Button(root, text="Select Image", width=11, bg='black', fg='gold',
                        font=('ariel 15 bold'), relief=GROOVE, command=select)
btn1.place(x=12, y=595,)
btn1.bind('<Enter>', onButton)
btn1.bind('<Leave>', leaveButton)
```

```
btn2 = Button(root, text="Save", width=10, bg='black', fg='gold',
                    font=('ariel 15 bold'), relief=GROOVE, command=save)
btn2.place(x=341, y=595)
        btn2.bind('<Enter>', onButton1)
        btn2.bind('<Leave>', leaveButton1)
```

```
        btn3 = Button(root,text="Exit", width=10, bg='black', fg='gold',
                font=('ariel 15 bold'), relief=GROOVE, command=root.destroy)
```

```
13.     btn3.place(x=492, y=595)
14.     btn3.bind('<Enter>', onButton2)
15.     btn3.bind('<Leave>', leaveButton2)
```

```
16.     btn4 = Button(root, text="Crop", width=12, bg='black',
    fg='gold',
                                    font=('ariel 15 bold'),
relief=GROOVE,
command=lambda:controls_window(w=image_width,
h=image_height))
17.     btn4.place(x=168, y=595)
18.     btn4.bind('<Enter>', onButton3)
19.     btn4.bind('<Leave>', leaveButton3)
```

```
20.     root.resizable(False, False)
```

✓ That is disable the maximization of window

Step 8: Run the App

To execute the app we use the

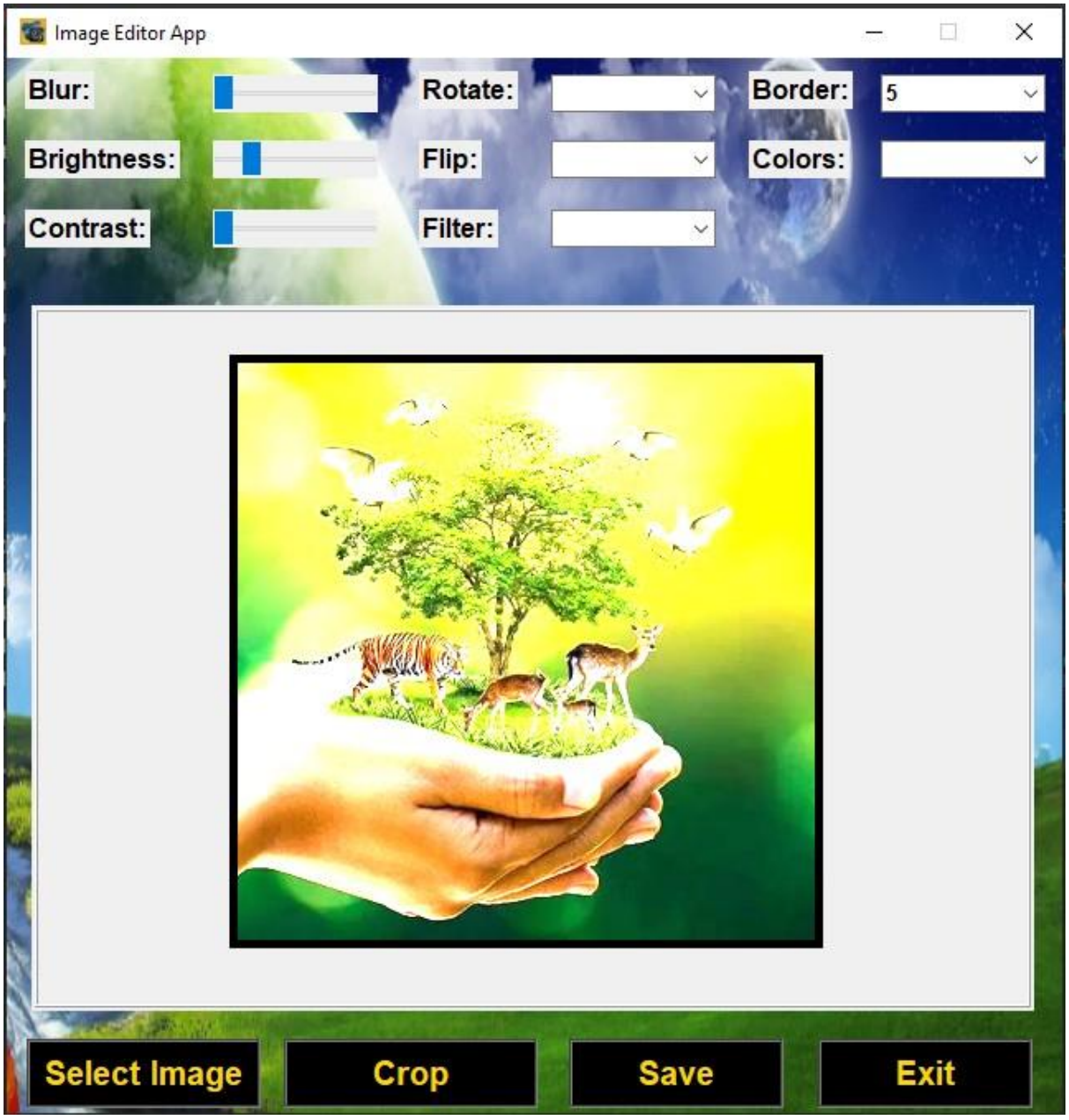
```
mainloop()
```

method. What it does is, it tells Python to run the
tkinter
event loop until the user exits it.

```
root.mainloop()
```

The Output

Running the above code will create a display of our Image editor app that will look as shown below with everything set as defined in the code.
As you can we have successfully added an image into our editor, feel free to play with different features provided and after that go ahead and save the copy of your edited image.



Conclusion:-

Now you can create your own Image Editor App using Python that has most functionality just like other software's out there. There is no limitation on the design of the app. Therefore, you can redesign it to meet your own specification and style. You can even add other interactivity features to it. It's about time you start using your own creations.