Technical Report

Python Deluxe Photo Editor v0.1.1

Team 10: “The Boys”

October 2019

TECHNICAL REPORT

TABLE OF CONTENTS

A. INFORMATION

1.1 Project Description

The Python Deluxe Photo Editor was developed by a quartet of undergraduate engineers at Purdue University. As members of the Engineering Projects In Community Service learning community, we have been given the opportunity to experience a trial curriculum which introduces students to the Python programming language to more gradually ease into the standard engineering curriculum in MATLAB. This group project serves as the final deliverable from this unit.

1.1.1 Objective

The objective of this program is to demonstrate the proficiency of the team members in Python. We were assigned to develop a photo-editing software with a number of specific features and abilities. As we could not use built-in functions provided by modules such as NumPy and MatPlotLib, we must learn how these functions operate and recreate them as best we can.

1.1.2 Team Members

Our team has a strong relationship and productive dynamic that allowed us to easily and evenly distribute tasks between members without conflict. Throughout the work periods in which this program was developed, we frequently helped one another with fixing errors, finding mistakes made, and discovering resources online that helped in troubleshooting. We are all at different skill levels when it comes to Python, but we have all had some sort of programming experience in the past. Thus, we feel relatively comfortable with any aspect of the project, and there are aspects of the assignment that very closely align with what each of us are most comfortable with.

Blake Lowe

Bio: From Indianapolis, Indiana, Blake enjoys aviation and intends to join the college of Aeronautical and Aerospace Engineering.

Contributions: Blake organized the main file and was in charge of blurring the inputted image. He also is responsible for setting up the GitHub repository our team used to share files and stay up to date with one another.

Christos Levy

Bio: Christos is from Detroit, Michigan, and his hobbies include playing guitar, driving and flying. He also intends to join AAE.

Contributions: Christos contributed to the project by developing the code to convert our inputted image to a usable NumPy array, mirror the image, and convert the RGB image to greyscale.

Marcus Lannie

Bio: Arlington Heights, Illinois is the hometown of the one and only Marcus Lannie, who spends his free time hammocking, swimming and yo-yoing. He's the future mechanical engineer of the group.

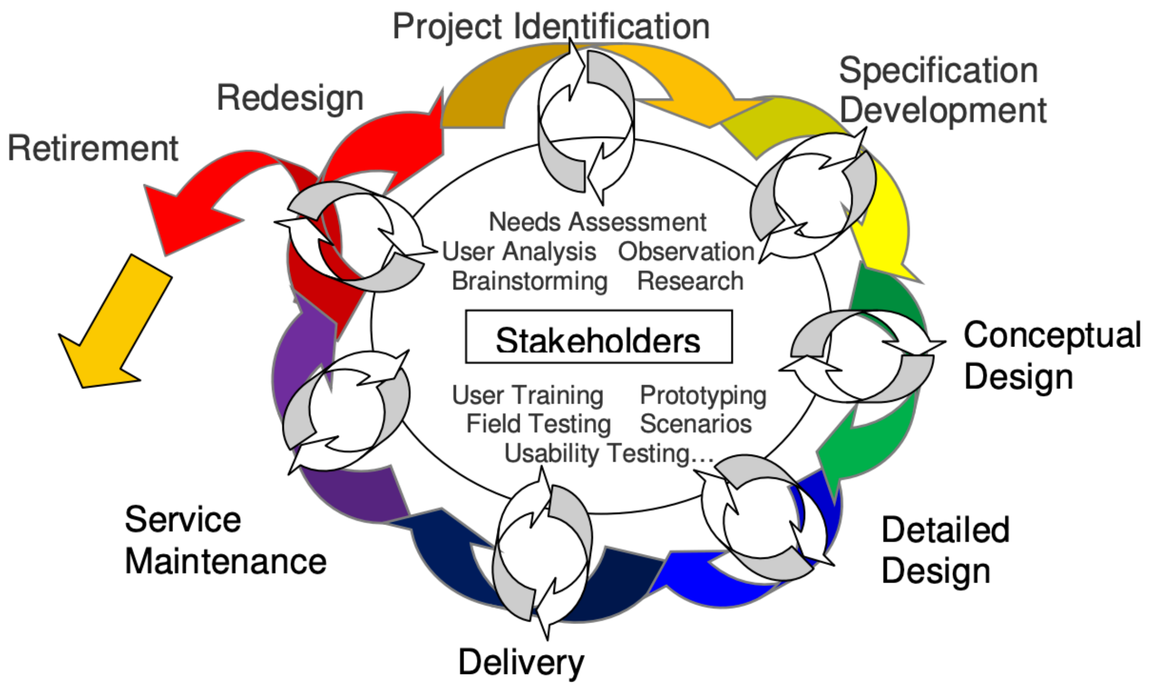
Contributions: Marcus developed the code to rotate the inputted image and assisted other team members in troubleshooting problems.

Thomas Weese

Bio: Thomas is from San Francisco, California, enjoys making coffee for himself and his friends, and intends to major in Industrial Engineering.

Contributions: Thomas wrote the script to output our NumPy array to a .png image saved to the computer. He also was in charge of assembling this report.

1.1.3 Design Process



The EPICS Design Process consists of the following phases:

Project Identification

The first thing our team did was define what exactly needed to be accomplished. We listed out what tasks we needed to accomplish in order to yield a working program. As they are given in the assignment, the required functions are as follows:

1. Image reading/writing. We must take an image (.jpg or .png) as an input and give an image of the same file type as our output.
2. Convert from a color image (RGB) to greyscale. This must be done without use of built in python functions capable of accomplishing this task with minimal effort.
3. Implement some sort of filter. This can be a blur, sharpen, or spatial derivative in either direction.
4. Rotate the image in 90° increments, as defined by the user.
5. Mirror the image
6. Check for likely errors

Aside from the program itself, our team was also required to assemble this report document, detailing how our program functions, how our team distributed work and used the design process, and what resources we used to research for this assignment. Additionally, this document is required to provide an appendix with all of the code found in the attached .py files, and descriptions of how the program is meant to be used with example inputs and expected outputs, including examples of errors.

Specification Development

In this phase of the process we broke up the needed tasks into assignments for individuals in the group and settled on how the whole program would work together. We decided to set up a GitHub repository to ensure everyone in the group had access to the most updated version of the document regardless of when and where it was updated. We also decided to have each function of the program be its own individual file, importing them into a main file (main.py) as modules. We chose this structure because it makes troubleshooting much easier, as each individual file is thoroughly commented and easy to follow.

Conceptual Design

Once we defined our general structure, we took our individual tasks and researched what we needed to accomplish our goals. Blake suggested we utilize the features offered by the MatPlotLib module to handle image conversion to/from an array, and NumPy to manipulate the array. With this base, we started flowcharting our programs in order to understand what was required to satisfy the expectations of the project.

Detailed Design

Once our team had our structure laid out, we proceeded to assemble our individual files containing the unique functions and developed a congregate main file to handle all the needed operations and user inputs. This process involved prototyping and troubleshooting to come up with a working program, and then going through and thoroughly commenting our code to make its operation and functionality very easy to follow.

Delivery

On Tuesday, October 29, we presented our program to the TA's with zero problems in any of our functions or error checks. Additionally, because of how the program is written, we never had to formally "restart" the script.

Service and Maintenance

We currently do not have any plans on how to conduct service and maintenance on the project, as our assignment only requires us to reach the delivery phase of the process.

1.1.4 Research Summary

The reasons our team opted for the MatPlotLib platform for our image handling system are because of its relative ease of use and, more importantly, its extensive and thorough documentation. Almost every problem we ran into could be solved by referring to the MatPlotLib online user's manual.

For general help with Python troubleshooting and matrix manipulation, we referred often to forums at stackoverflow.com.

1.2 System Overview

A custom-made framework of concise Python scripts that, combined together, perform different photo-editing operations in an efficient, understandable, and easy to operate. Please check to ensure files included in application zip are as follows:

- arraytoimage.py
  - File is responsible for converting the manipulated array into altered image.
- blur.py
  - File provides program the capability to blur the inputted image to a user-defined magnitude.
- grayscale.py
  - File provides program the capability to convert the RGB color inputted image to grayscale.

- image_mirroring.py
    - File provides program the capability to mirror the inputted image.
- import_file_to_array.py
    - File is responsible for converting the inputted image to an array for manipulation in the program.
- main.py
    - File receives user inputs, aggregates the capabilities of all other files, and saves outputted image to a user-controlled file name.
- rotate_picture.py
    - File provides program the capability to rotate the inputted image in 90 degrees increments.

B. SYSTEM SUMMARY

At a fundamental level, our program is an aggregate of several small python files that offer new functionality to our main file, which interfaces with the user.

2.1 How the Functions Work

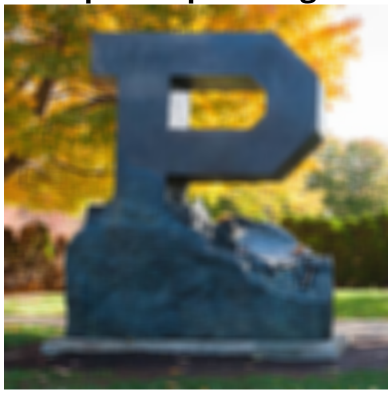
Example input image:

Import File to Array

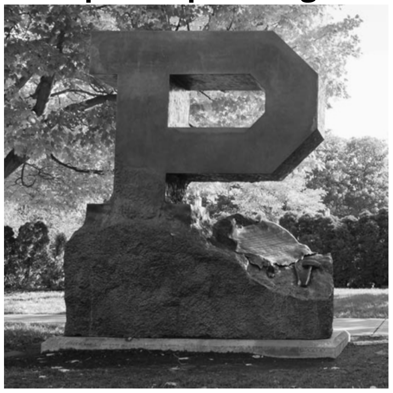
The image to array function takes the path of the image location and using MatPlotLib's image functions it automatically converts the image to an array of arrays of an array (3D array).

Blur

The blur function uses a weighted average for each pixel to make it closer in color to its neighbors. It automatically generates a matrix of weights (kernel) with the Gaussian function, which gives the normal distribution, to allow differing amounts of blur in the final image.
Example output image:



Grayscale

Inside each 3rd level array, there are 4 values. The 4 values represent red, green, blue, and alpha colors. To use the grayscale, we want to manipulate the red, green and blue values AKA the RGB values. Convert to grayscale, we have to take a weighted average of each of the values in a single array and then assign that average to each of the RGB values in the array. After completing the whole image, it will return the new image array.
Example output image:


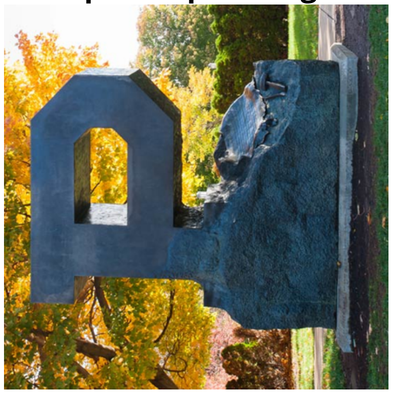
Mirror

The mirror function in essence takes the 3D image array and loops through every single array element and moves it to the opposite side. It then returns that new array with all the values flipped
Example output image:

Rotate

The rotate function begins by initializing variables to store the heights, width, and channel (color) of the image. After these values are stored, a new matrix with height and width opposite to the original matrix is created and is filled with all zeroes. This new matrix will soon be populated with the pixels of the original image in a way that makes the image produced appear to have rotated a specified number of degrees. The user will input a degree measure which will be used to determine the number of times to run the function. For example, entering "180" runs the function twice because the image is rotated 90 degrees twice. The program copies each pixel in the original matrix into its appropriate spot in the new matrix. After iterating through every pixel and adding to the new matrix, the new rotated image is returned.

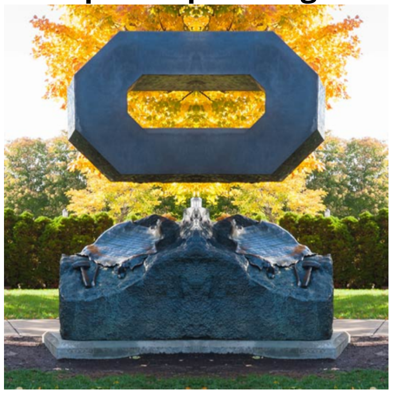
Example output image:



Reflect

The reflect function is similar to the mirror function except that it only moves half the elements over. That way the 2 sides of the image match. This can be done over the horizontal axis where the bottom is the same as the top but inverted, or over the vertical axis where the left side is the same as the right but flipped around.

Example output image:

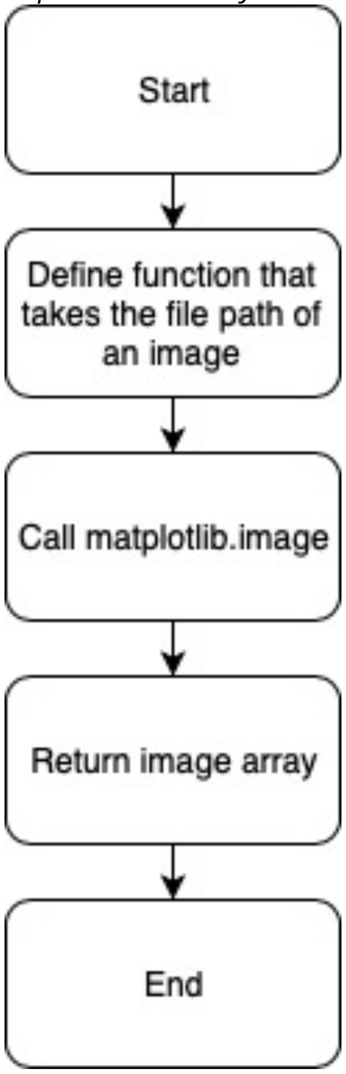


Convert Array to Image

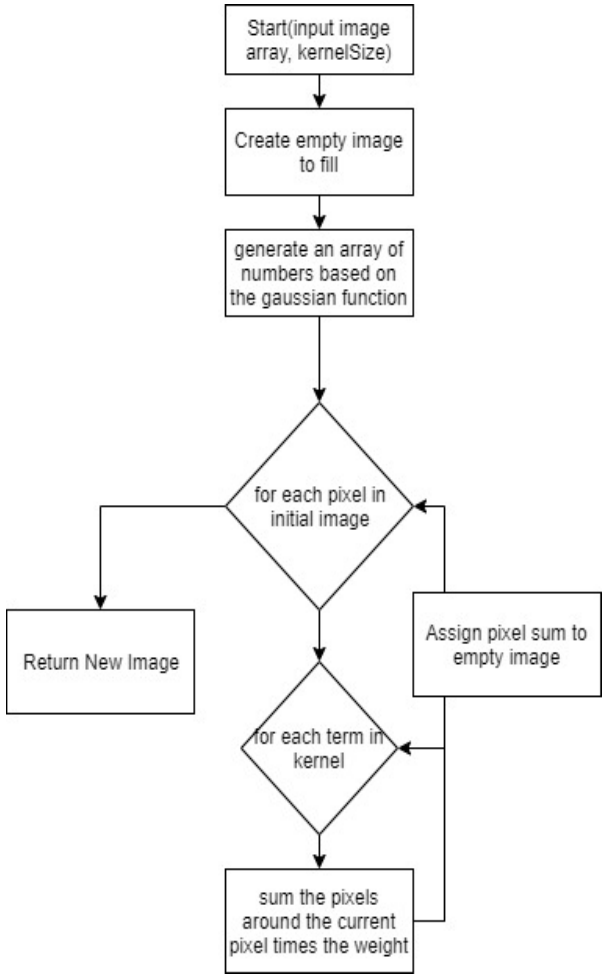
Using tools in the MatPlotLib module, this script converts our manipulated NumPy array into a new image with a desired name.

2.2 Program Flowcharts

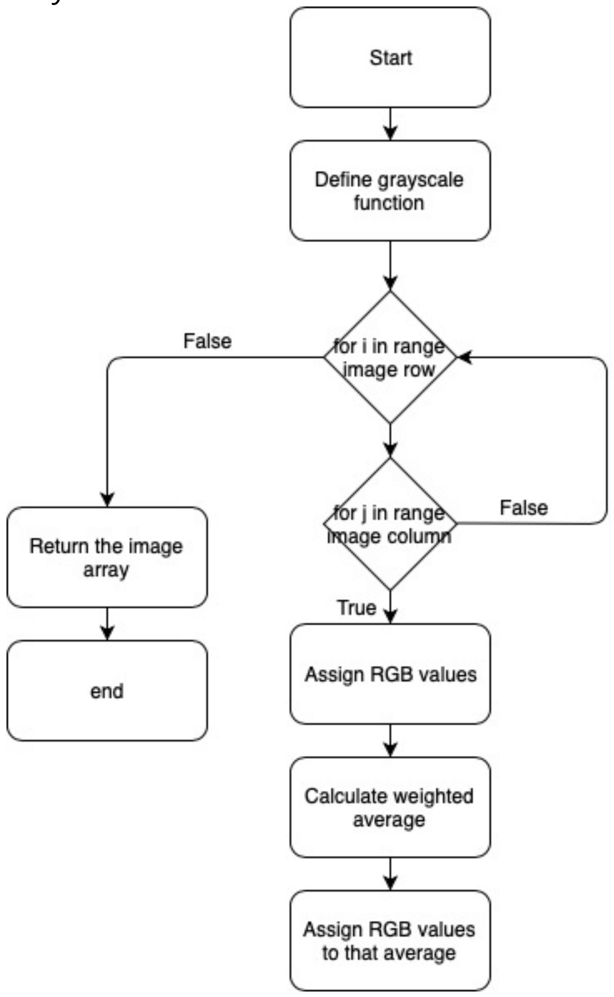
Import File to Array



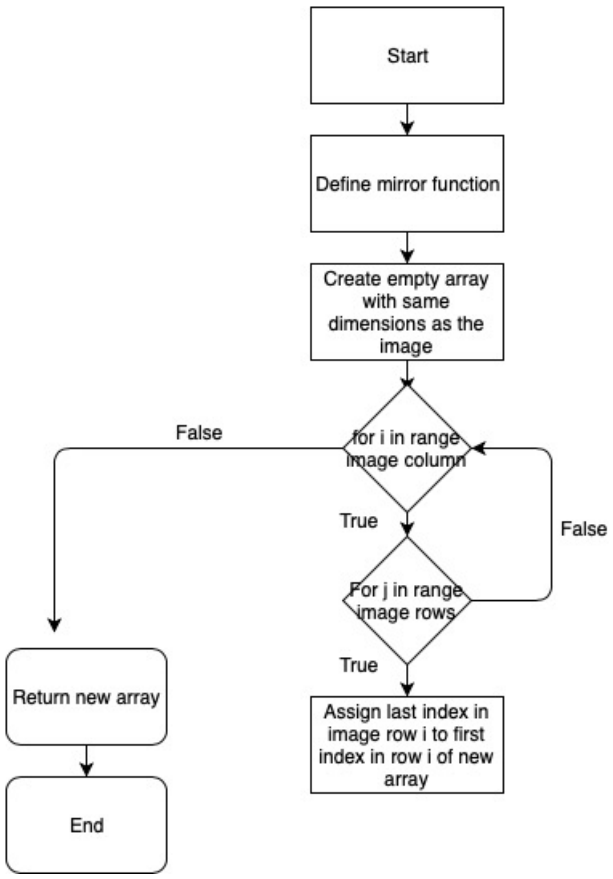
Blur


Start(input image array, kernelSize)
Create empty image to fill
generate an array of numbers based on the gaussian function
for each pixel in initial image
Return New Image
for each term in kernel
sum the pixels around the current pixel times the weight
Assign pixel sum to empty image

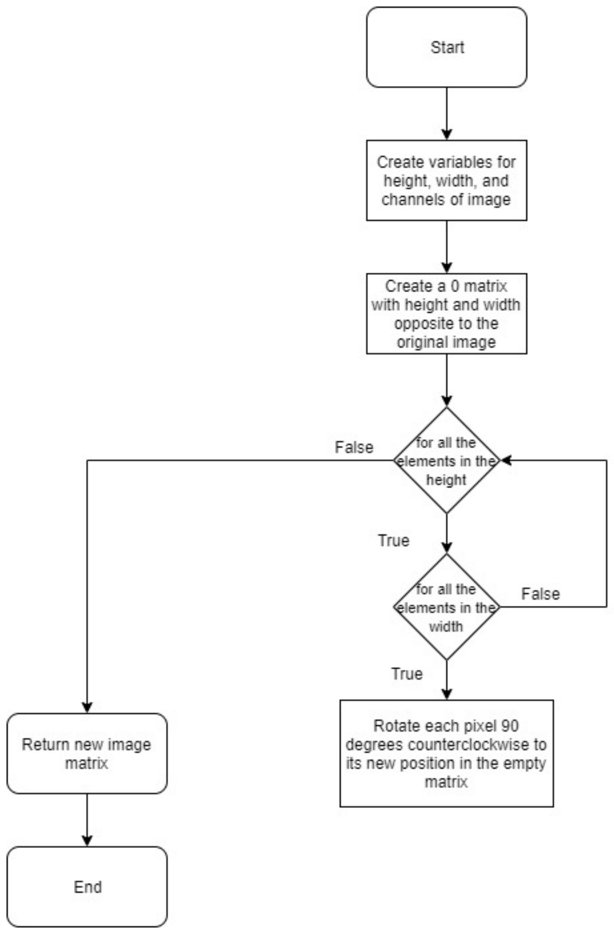
Grayscale


Start
Define grayscale function
for i in range image row
False
for j in range image column
False
True
Return the image array
end
Assign RGB values
Calculate weighted average
Assign RGB values to that average

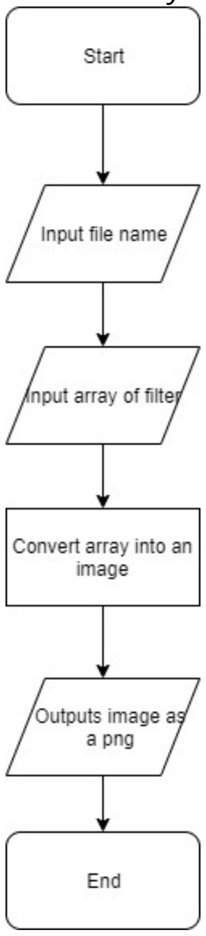
Mirror


Start
Define mirror function
Create empty array with same dimensions as the image
for i in range image column
False
True
For j in range image rows
False
True
Return new array
End
Assign last index in image row i to first index in row i of new array

Rotate


Start
Create variables for height, width, and channels of image
Create a 0 matrix with height and width opposite to the original image
for all the elements in the height
False
True
for all the elements in the width
False
True
Rotate each pixel 90 degrees counterclockwise to its new position in the empty matrix
Return new image matrix
End

Convert Array to Image



C. HOW TO USE PROGRAM

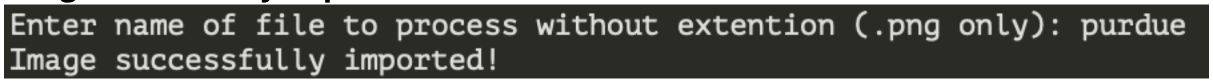
3.1 Setup

Open the main Python file in your preferred IDE and ensure that it is able to successfully import all other files in the program. Check that the images you desire to edit are in the correct directory location. Press run to initiate the program. To operate, you will receive the following prompts:

1. Enter name of file to process without extention (.png only):
2. Functions: blur, grayscale, rotate, mirror, reflect
3. Enter name for output file without extension:
4. Run Again? (Y/N):

3.2 Operations

The first thing the user is prompted to input is the name of image file intended to be edited. If the file exists in the folder, you will receive the following prompt:
Image successfully imported!



```
Enter name of file to process without extention (.png only): purdue
Image successfully imported!
```

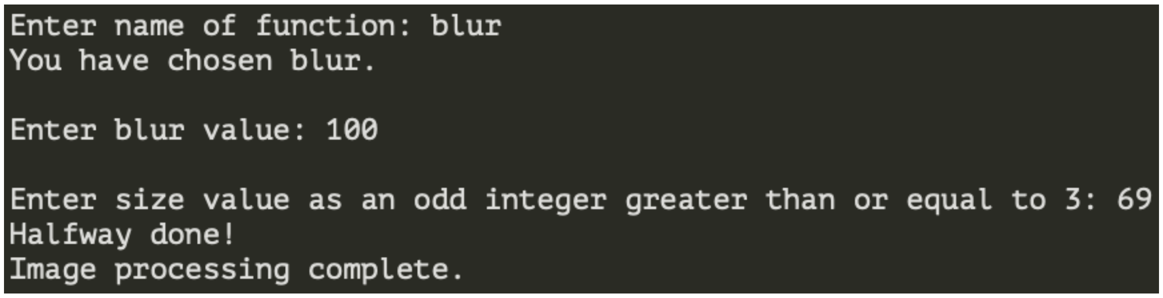
You may next select which operation you would like to use to edit your image. Your options are as follows:

1. Blur
    a. Enter blur value:

The blur value is a value between 0 and 100 that dictates the influence surrounding pixels have on the weighted average of its color. The higher the value, the more alike to its neighbors it will become.

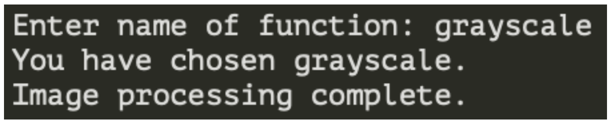
    b. Enter size value as an odd integer greater than or equal to 3:

The size value is a value which dictates the size of the proximity of pixels which influence an individual pixel. As the size value increases, the level of relative blur increases dramatically. *Note: a size value greater than 99 will result in the program taking several minutes to calculate.*



```
Enter name of function: blur
You have chosen blur.

Enter blur value: 100

Enter size value as an odd integer greater than or equal to 3: 69
Halfway done!
Image processing complete.
```

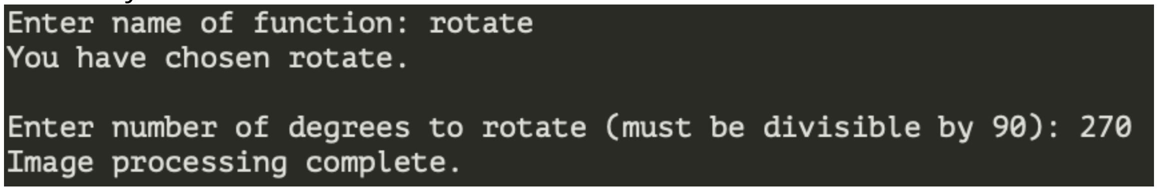
2. Grayscale



```
Enter name of function: grayscale
You have chosen grayscale.
Image processing complete.
```

3. Rotate
    a. Enter number of degrees to rotate (must be divisible by 90):

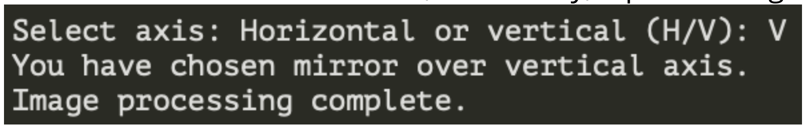
The rotate function rotates the image in the counterclockwise direction, so 270° in the function yields a desired 90° clockwise rotation.



```
Enter name of function: rotate
You have chosen rotate.

Enter number of degrees to rotate (must be divisible by 90): 270
Image processing complete.
```

4. Mirror
    a. Select axis: Horizontal or vertical (H/V):

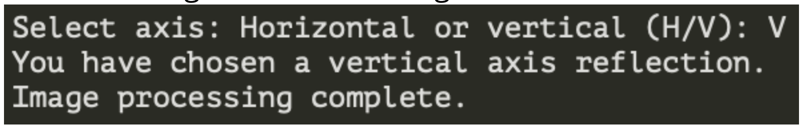
A horizontal mirror will flip the image along its horizontal axis, resulting in an "upside-down" look. The vertical mirror, conversely, flips the image along its vertical axis.



```
Select axis: Horizontal or vertical (H/V): V
You have chosen mirror over vertical axis.
Image processing complete.
```

5. Reflect
    a. Select axis: Horizontal or vertical (H/V):

The reflect function was not a required feature of our code and came about unintentionally. This function sets the mirror axis to the center of the image, and either reflects the right half of the image onto the left or reflects the top half onto the bottom.

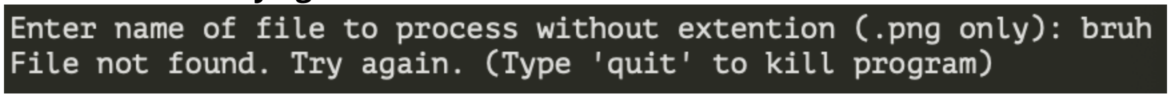


```
Select axis: Horizontal or vertical (H/V): V
You have chosen a vertical axis reflection.
Image processing complete.
```

3.3 Possible Errors

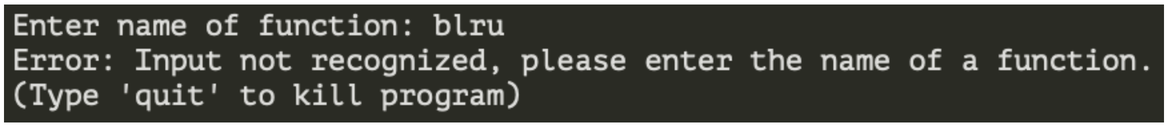
If the file name you've entered does not exist in the folder, you will receive the following prompt:

File not found. Try again.

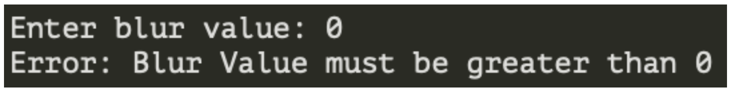


```
Enter name of file to process without extention (.png only): bruh
File not found. Try again. (Type 'quit' to kill program)
```

The possible errors to expect later in the function are as follows:

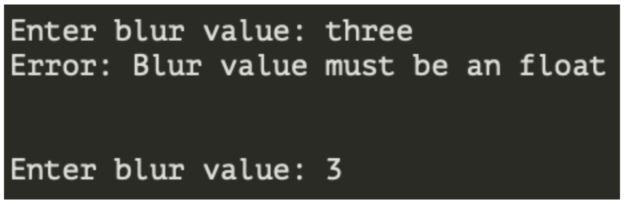
1. Inputted function is not one of the five listed
    a. 

```
Enter name of function: blru
Error: Input not recognized, please enter the name of a function.
(Type 'quit' to kill program)
```

2. Blur
    a. Blur value is 0
        i. 

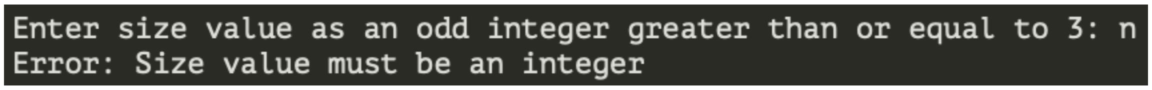
```
Enter blur value: 0
Error: Blur Value must be greater than 0
```

b. Blur value is not numeric

i. 

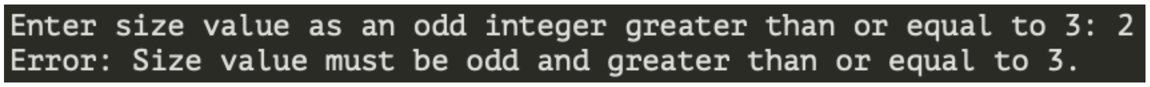
```
Enter blur value: three
Error: Blur value must be an float


Enter blur value: 3
```

c. Size value entered is not numeric

i. 

```
Enter size value as an odd integer greater than or equal to 3: n
Error: Size value must be an integer
```

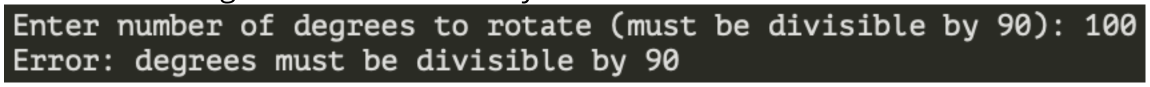
d. Size value entered is even or is less than 3

i. 

```
Enter size value as an odd integer greater than or equal to 3: 2
Error: Size value must be odd and greater than or equal to 3.
```

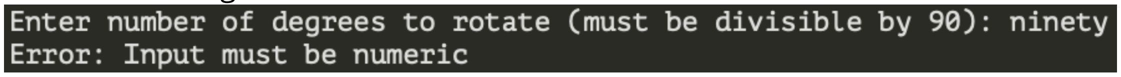
3. Rotate

a. Inputted number of degrees is not divisible by 90

i. 

```
Enter number of degrees to rotate (must be divisible by 90): 100
Error: degrees must be divisible by 90
```

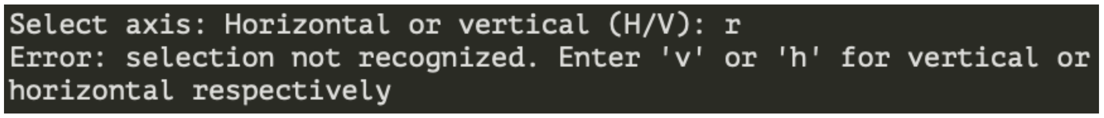
b. Inputted number of degrees is not numeric

i. 

```
Enter number of degrees to rotate (must be divisible by 90): ninety
Error: Input must be numeric
```

4. Mirror

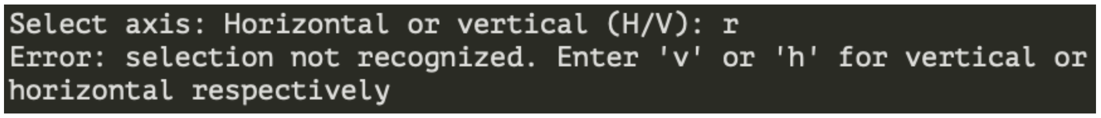
a. Inputted axis of rotation is not H or V

i. 

```
Select axis: Horizontal or vertical (H/V): r
Error: selection not recognized. Enter 'v' or 'h' for vertical or
horizontal respectively
```

5. Reflect

a. Inputted axis of rotation is not H or V

i. 

```
Select axis: Horizontal or vertical (H/V): r
Error: selection not recognized. Enter 'v' or 'h' for vertical or
horizontal respectively
```

3.4 Exiting the Program

If you have just started the file or have only selected your image, you may type ‘quit’ in the terminal to kill the program. Once you have selected an entered a function, however, the system has no way for you to kill the program until it has saved the new image to a file. Upon doing so, you are given the option to run the program again or end the program.

A. APPENDIX

ARRAYTOIMAGE.PY

```
import import_file_to_array as ifta
import matplotlib as mpl

## Takes an input name and the array of the filter and outputs an image in the directo
ry
def outimage(name, array):
    mpl.image.imsave(name + '.png', array)
    return
```

BLUR.PY

```
import math
import numpy as np


def process(imageData, blur, size):
    channelCount = len(imageData[0][0])#determine if RGB or RGBA
    kernel = getKernel(size, blur)
    outimage1 = np.empty([len(imageData), len(imageData[0]), channelCount])#create emp
ty image with same dimensions
    outimage2 = np.empty([len(imageData), len(imageData[0]), channelCount])#create emp
ty image with same dimensionsas original
    #vertical blur only
    for i in range(0, len(imageData)):#for each row
        #print(i)
        for j in range(0, len(imageData[0])):#for each column
            weightedAvg = np.zeros(channelCount)

            for h in range(0, len(kernel)):
                dy = -len(kernel)//2 + h#relative change in pixel y

                if j+dy >= 0 and j+dy < len(imageData[0]):
                    pixelY = j+dy
                elif j+dy < 0:
                    pixelY = 0
                elif j+dy > len(imageData[0]):
                    pixelY = -1
                pixelX = i
                pixel = imageData[pixelX][pixelY]#get pixel data for target pixel

                weightedAvg += np.multiply(kernel[h], pixel)#sum the corresponding ARG
B or RGB numbers
```

```
            outimage1[i][j] = weightedAvg#output to temp matrix

    print("Halfway done!")

    #horizontal blur only
    for i in range(0, len(imageData)):#for each row
        #print(i)
        for j in range(0, len(imageData[0])):#for each column
            weightedAvg = np.zeros(channelCount)

            for k in range(0, len(kernel)):
                dx = -len(kernel)//2 + k#relative change in pixel y

                if i+dx >=0 and i+dx < len(imageData):#if pixel is out of bounds, exte
nd the image
                    pixelX = i+dx
                elif i+dx < 0:
                    pixelX = 0
                elif i+dx >= len(imageData):
                    pixelX = -1
                pixelY = j
                pixel = outimage1[pixelX][pixelY]#get pixel data for target pixel

                weightedAvg += np.multiply(kernel[k], pixel)

            outimage2[i][j] = weightedAvg#output to matrix

    return outimage2#return final image

def getKernel(size, stddev):
    kernel = []
    center = (int(size))//2
    for i in range(0, size):
        x = i-center
        kernel.append(gaussian(x, stddev))
    kernel = normalizeArray(kernel)
    return kernel

def gaussian(x, stddev):
    A = 1/(math.sqrt(2*math.pi)*stddev)
    return A*math.exp(-(x*x)/(2*stddev*stddev))

def normalizeArray(array):#1D array input
    total = 0
    for i in range(len(array)):
        total += array[i]
    for i in range(len(array)):
        array[i] /= total

    return(array)

      GRAYSCALE.PY

import numpy as np

##Takes the image array and returns it as a gray image by using the weighted average R
BG formula
def makeGray(image):
    for i in range(len(image)):
        for j in range(len(image[i])):
            ## Finds value of RGB nums
            r = image[i][j][0]
            g = image[i][j][1]
            b = image[i][j][2]

            ## Calculates grayscale by using the weighted average formula
            Avg = 0.2989*r+0.5780*g+0.1140*b
            ## Reassigns colors to the average
            image[i][j][0], image[i][j][1], image[i][j][2] = Avg,Avg,Avg

    return image

      IMAGE_MIRRORING
import numpy as np



def flipIt(image): #FLIPS THE IMAGE ACROSS VERTICAL AXIS
    outimage = np.empty([len(image),len(image[0]),len(image[0][0])])
    for i in range(len(image)):
      for j in range(len(image[i])):
          outimage[i][j] = image[i][-j-1]
    return outimage

def flipH(image): #FLIPS IMAGE ACROSS HORIZONTAL AXIS
    outimage = np.empty([len(image),len(image[0]),len(image[0][0])])
    for i in range(len(image)):
      for j in range(len(image[i])):
          outimage[i][j] = image[-i-1][j]
    return outimage

def mirrorH(image): # "MIRRORS" IMAGE ACROSS HORIZONTAL AXIS
    for i in range(len(image)):
      for j in range(len(image[i])):
          image[i][j] = image[-i-1][j]
    return image

def mirrorV(image): # "MIRRORS" IMAGE ACROSS VERTICAL AXIS
    for i in range(len(image)):
      for j in range(len(image[i])):
          image[i][j] = image[i][-j-1]
    return image
```

IMPORT_FILE_TO_ARRAY.PY

```
import matplotlib.image as mpimg
import numpy as np

## Takes a file path and returns the array form of the image
def importImage(path): ## Where path is the file path as a string
    image = mpimg.imread(path)
    return image
```

MAIN.PY

```
## IMPORT FUNCTIONS
import numpy as np
import import_file_to_array as ifta
import image_mirroring as im
import arraytoimage as ati
import rotate_picture as rp
import grayscale as gs
#import blur as bl ## Unimplemented
import fastblur as fbl
import os

## Asks for name of the file
while True:
    isKilled = False
    while True:
        infName = input("Enter name of file to process without extention (.png only):
")

        ## Changes kill variable to true to quit program
        if(infName.lower() == "quit"):
            isKilled = True
            break
        filepath = os.getcwd()+ "/" + infName+ ".png"

        ## Converts image to an array after successful import
        try:
            inImage = ifta.importImage(filepath)
            print("Image successfully imported!")
            break

        ## Repeats if the file isn't found
        except:
            print("File not found. Try again. (Type 'quit' to kill program)\n")
            continue

    ## Quits program when kill variable is set true
    if isKilled == True:
        print("\n\n\n")
        break

    if isKilled == False:
        ## Asks user for desired function
        while True:
            print("Functions: blur, grayscale, rotate, mirror, reflect")
            processName = input("Enter name of function: ").lower()
            if(processName == "quit"):
                isKilled = True
                print("Killing program...")
                break

            ## Runs FastBlur Function and assures input values are (str, float, int)
            elif(processName == "blur"):
                print("You have chosen blur.")
                blurValue = 0
                size = 0
                ## Checks if the values are correct data type and valid
                while True:
                    blurValue = input("Enter blur value: ")
                    try:
                        blurValue = float(blurValue)
                        if blurValue == 0:
                            print("Error: Blur Value must be greater than 0\n")
                            continue
                        else:
                            break
                    except:
                        print("Error: Blur value must be an float\n")
                        continue

                while True:
                    size = input("Enter size value as an odd integer greater than or e
qual to 3: ")
                    try:
                        size = int(size)
                    except:
                        print("Error: Size value must be an integer\n")
                        continue

                    if size%2 == 0 or size < 3:
                        print("Error: Size value must be odd and greater than or equal
 to 3.\n")

                        continue
                    else:
                        break
                ## Creates blurred image
                outImage = fbl.process(inImage,blurValue,size)
                print("Image processing complete.")
                break

            ## Runs Grayscale Function
            elif(processName == "grayscale"):
                print("You have chosen grayscale.")
                outImage = gs.makeGray(inImage)
                print("Image processing complete.")
                break

            ## Runs Rotate Function
            elif(processName == "rotate"):
                print("You have chosen rotate.")
                degrees = 0
                while True:
                    degrees = input("Enter number of degrees to rotate (must be divisi
ble by 90): ")

                    ## Checks to see if input is a number
                    try:
                        degrees = int(degrees)
                    except:
                        print("Error: Input must be numeric\n")
                        continue

                    ## Checks if the number is a factor of 90
                    if degrees%90 != 0:
                        print("Error: degrees must be divisible by 90.\n")
                        continue
                    else:
                        break
                outImage = rp.rotate(inImage, degrees)
                print("Image processing complete.")
                break
            ## Runs mirror function
            elif(processName == "mirror"):
                print("You have chosen mirror.")
                ## Loop to detect axis
                while True:
                    axisSelect = input("Select axis: Horizontal or vertical (H/V): ").
lower()
                    if axisSelect == "h":
                        print("You have chosen mirror over horizontal axis.")
                        outImage = im.flipH(inImage)
                        break
                    elif axisSelect == "v":
                        print("You have chosen mirror over vertical axis.")
                        outImage = im.flipIt(inImage)
                        break
                    ## Prints error if the selection isn't recognized
                    else:
                        print("Error: selection not recognized. Enter 'v' or 'h' for v
ertical or horizontal respectively\n")

                print("Image processing complete.")
                break

            ## Runs reflect function
            elif processName == "reflect":
                print("You have chosen reflect.")
                ## Loop to detect axis
                while True:
                    axisSelect = input("Select axis: Horizontal or vertical (H/V): ").
lower()
                    if axisSelect == "h":
                        print("You have chosen a horizontal axis reflection.")
                        outImage = im.mirrorH(inImage)
                        break
                    elif axisSelect == "v":
                        print("You have chosen a vertical axis reflection.")
                        outImage = im.mirrorV(inImage)
                        break
                    ## Prints error if the selection isn't recognized
                    else:
                        print("Error: selection not recognized. Enter 'v' or 'h' for v
ertical or horizontal respectively\n")
                        continue
                print("Image processing complete.")
                break
            ## Prints an error if a function is not found
            else:
                print("Error: Input not recognized, please enter the name of a functio
n. (Type 'quit' to kill program)\n")

    ## Creates and outputs the file to the directory
    if isKilled == False:
        fileName = input("Enter name for output file without extension: ")
        ati.outimage(fileName,outImage)

        ## Asks if user wishes to run again
        runDecision = input("Run Again? (Y/N): ").lower()
        if runDecision == "n":
            print("Thank you for using the Python Deluxe Photo Editor!\n\n\n")
            isKilled = True
            break
        else:
            continue
```

ROTATE_PICTURE.PY

```
import numpy as np


## Rotates image based on specified degrees
def rotate(image,degrees):

    ## Will rotate image 90 degrees counterclockwise
    if degrees == 90:
        h,w,c = image.shape
        empty_image = np.zeros([w,h,c]) ## Creates a new image that has opposite dimen
sions as original
        for i in range(h):
            for j in range(w):
                empty_image[-j-1,i] = image[i,j]
        return(empty_image)
    ## Rotates image 180 degrees or mirrors over horizontal axis
    elif degrees == 180:
        empty_image = np.empty([len(image),len(image[0]),len(image[0][0])]) ## Creates
 a new image that has same dimensions as original
        for i in range(len(image)):
            for j in range(len(image[i])):
                empty_image[i][j] = image[-i-1][-j-1]
        return empty_image

    ## Rotates image 270 degrees counterclockwise
    elif degrees == 270:
        h,w,c = image.shape
        empty_image = np.zeros([w,h,c]) ## Creates a new image that has opposite dimen
sions as original
        for i in range(h):
            for j in range(w):
                empty_image[j,-i-1] = image[i,j]
        return(empty_image)
```