PLTW COMPUTER SCIENCE

Project 1.4.5

Image Algorithms

Introduction

You've learned the basics of writing computer programs. But most programming builds on code that has already been written. Knowing how to find and use code from other people will help make you an efficient and successful software developer.

You could create your own algorithms, for example, to rotate an image or to identify the objects in an image. But others have already solved those problems! There are many advantages to using existing code. You save time, of course. But you also connect to a community of people, making it easier for them to help you, and making it more likely they will be able to use what you create. How will you put other people's code to use?

Materials

- Computer with Enthought Canopy distribution of Python
- Webcam or other way to capture a digital picture
- Image files and Python files for Activity 1.4.5

Resources

1.4.5 sourceFiles.zip

1.4.5 Images.zip

Reference Card for Pyplot and PIL

Procedure

pairs as directed by your teacher. Meet or greet each other to practice professional

Set team norms.

ch Canopy. Open an editor window. Set the working directory to your folder. Create a Python file. Save the file as `JDoe_JSmith_1_4_5.py`.

e of your work in this assignment will involve trial and error using the IPython session. night want a record of commands you have tried, so begin logging the session.

```
]: %logstart -ort studentNames_1_4_5.log
```

n and unzip `1.4.5 sourceFiles` and `1.4.5 imageFiles`. Open `mask.py` provided source files. Execute the code in the code editor. This code defines three new functions oes not actually call any of them, so nothing will visibly occur. Examine the code in the py code editor. What are the names of the three functions?

IPython session, change your working directory to the unzipped folder `1.4.5 Images`. ine the contents of that folder using Windows Explorer.

IPython session, execute the following command. The function will take a moment to ute.

```
]: round_corners_of_all_images()
```

unction will create a new folder `modified` in the `1.4.5 Ijmges` folder. Examine the older's contents. What did the function do?

tion of `mask.py` is shown below. This is the code for the first function, `d_corners_one_image()`, defined in the program file. Answer the questions below the code.

```
round_corners_one_image(original_image, percent_of_side=.3):
""" Rounds the corner of a PIL.Image

original_image must be a PIL.Image
Returns a new PIL.Image with rounded corners, where
0 < percent_of_side < 1 is the corner radius as
portion of shorter dimension of original_image
"""
# Set the radius of the rounded corners
width, height = original_image.size
radius = int(percent_of_side * min(width, height)) #radius in pixels

###
# Create a mask
###

#start with transparent mask
rounded_mask = PIL.Image.new('RGBA', (width, height), (127,0,127,0))
drawing_layer = PIL.ImageDraw.Draw(rounded_mask)

# Overwrite the RGBA values with A=255.
# The 127 for RGB values was used merely for visualizing the mask
```

```
# Draw two rectangles to fill interior with opaqueness
drawing_layer.polygon([(radius,0),(width-radius,0),
                       (width-radius,height),(radius,height)],
                       fill=(127,0,127,255))
drawing_layer.polygon([(0,radius),(width,radius),
                       (width,height-radius),(0,height-radius)],
                       fill=(127,0,127,255))

# Draw four filled circles of opaqueness
drawing_layer.ellipse((0,0, 2*radius, 2*radius),
                       fill=(0,127,127,255)) #top left
drawing_layer.ellipse((width-2*radius, 0, width,2*radius),
                       fill=(0,127,127,255)) #top right
drawing_layer.ellipse((0, height-2*radius, 2*radius,height),
                       fill=(0,127,127,255)) #bottom left
drawing_layer.ellipse((width-2*radius, height-2*radius, width, height),
                       fill=(0,127,127,255)) #bottom right

# Uncomment the following line to show the mask
# plt.imshow(rounded_mask)

# Make the new image, starting with all transparent
result = PIL.Image.new('RGBA', original_image.size, (0,0,0,0))
result.paste(original_image, (0,0), mask=rounded_mask)
return result
```

The function `round_corners_one_image()` was one we made up. It is defined here to take _____ arguments. According to the function's docstring (lines 7 – 13), what type of variable is each argument? What type of variable is returned by the function?

Argument 1:

Argument 2:

Return value:

Line 23 creates a new image filled with a single color. What color is it?

Line 24 creates a new `ImageDraw` object associated with the new `PIL.Image` object from line 23. What are the names of these two objects?

Object created in line 23:

Object created in line 24:

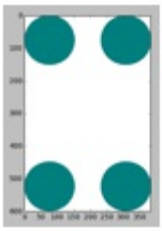
In Step 17g of the last activity, you used the `PIL.Image.paste()` documentation to identify the purpose of the mask argument of `paste()` in that program. Refer to your answer to that question.

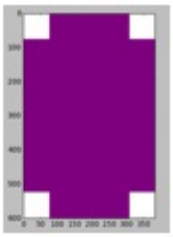
The `rounded_mask` object is used as the mask argument in line 52. The `paste()`

function uses only the alpha channel of the mask argument. It uses this alpha value to decide how to combine the pixels of the two other images. To make an image transparent in the corners, what alpha value would we want for the mask in the corners?

Lines 30 – 35 take advantage of the fact that *Python* allows a line to be continued onto the next line immediately after the comma in a list of arguments. Note the convention to indent the line continuation: the continued line is indented to line up with the parenthesis that begin the argument list.

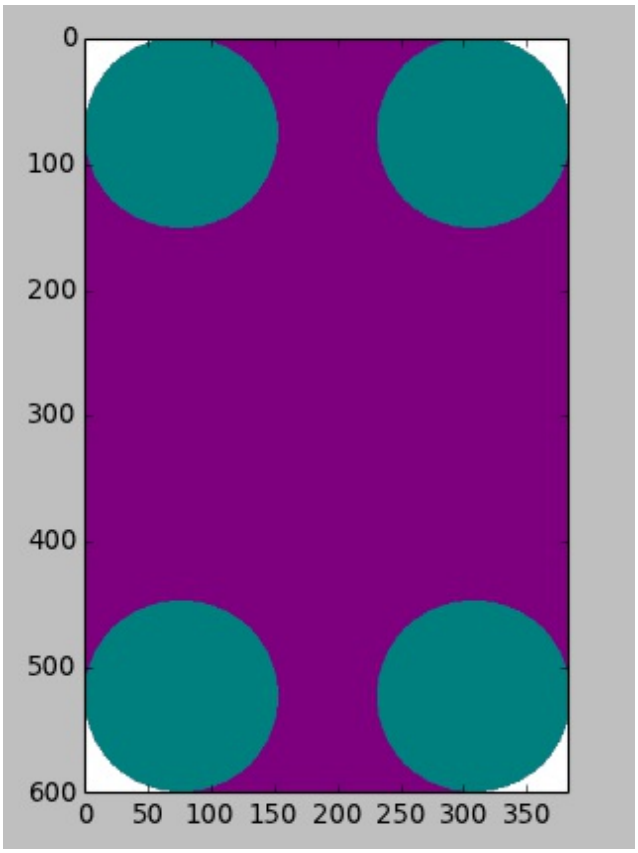
The following images were produced by uncommenting line 48 and using triple single-quotes to comment out either lines 30 – 35 or lines 38 – 45. The result is that all six shapes created in lines 30 – 45 are shown. You don't need to repeat that process; just read the code. In the figures below, label each shape with the corresponding line number used to create it.





Line 51 creates another new `PIL.Image` object called result. It will hold the modified image, but when created, it is filled with a solid color. What color is it?

Line 52 pastes the original image into result. Pixels from the `original_image` are only used if the corresponding pixels from `rounded_mask` have alpha>0. The pixels in the corners are left as-is in `result`. What are the color values in the corners?



ode shown below defines `get_images()`, the second function created by `mask.py`.
to the code and answer the following questions.

```
get_images(directory=None):
""" Returns PIL.Image objects for all the images in directory.

If directory is not specified, uses current directory.
Returns a 2-tuple containing
a list with a PIL.Image object for each image file in root_directory,
and a list with a string filename for each image file in root_directory
"""

if directory == None:
    directory = os.getcwd() # Use working directory if unspecified

image_list=[] # Initialize aggregators
file_list = []

directory_list = os.listdir(directory) # Get list of files
for entry in directory_list:
    absolute_filename = os.path.join(directory, entry)
    try:
        image = PIL.Image.open(absolute_filename)
        file_list += [entry]
        image_list += [image]
    except IOError:
        pass # do nothing with errors tying to open non-images
```

```
return image_list, file_list
```

How many arguments can be passed to the function `get_images()`? Because a default value is specified for directory, that argument is optional, so `get_images()` can be passed either __ or __ arguments.

Read the docstring and examine the return statement on line 79. How many objects and what type are returned by the function?

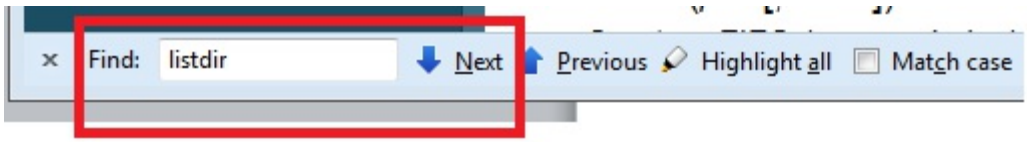
This function calls three functions from the `os` module. Find the three calls to the `os` module used in the code and list the three functions below.

`os._____()`

`os._____()`

`os.path._____()`

Use an Internet search engine to find the official documentation for the `os` module. You could try “os Python” for your search terms. You can identify the official documentation because it will come from a webserver in the python.org domain. Use the find-in-document utility (**Ctrl-F**, then repeatedly choose **Next**) to find the documentation for one of the functions above. Paste a sentence about that function above.



Lines 73 – 78 demonstrate some statements new to you. This is a `try-except` structure, which is the *Python* exception handler. An exception handler lists the code to be executed if an error occurs.

The `PIL.Image.open(filename)` function can cause an error that would halt the program if the filename does not specify an image file. Specifically, `open()` reports an `IOError` type of error. If that error is reported to the Python interpreter, the program is halted and the error is printed in the traceback at the interpreter prompt. The traceback shows what lines of code caused the error.

By using a `try-except` structure, such an error is caught instead of halting the program. An error that has been caught doesn’t get reported back to the *Python* interpreter. The handler can opt to keep the error invisible to the user and keep the program running. That’s a good thing if the - programmer expected the error and wants the program to keep running. That can be a bad thing if the code accidentally catches other exceptions, like the user trying to quit the program. So the program should only catch the specific class of errors that are expected, such as `IOError` in this case.

Here is how the `try-except` structure controls the program flow. The statements in the `try` block are executed one at a time. If one of those statements causes an error, the interpreter checks to see if the type of error matches the type of errors listed in the `except` statement. If the error type matches the except statement, then the interpreter does not execute the rest of the try block and instead continues execution with the `except` block of code. If the error doesn't match the except statement, then the error is not caught and the program will be halted.

In this code the `except` block only contains the *Python* `pass` statement, which does nothing. It is used when *Python* syntax requires a statement but no action is required. So the `except` block catches the error but doesn't do anything with it.

If the `try` block of code is executed without any errors, the `except` block of code is skipped. Execution continues after the `try-except` structure.

Why does this program use a `try-except` structure to open all images in a directory?

Considering the information above, explain what lines 77 and 78 do. Describe which circumstances allow them to be executed.

ode shown below defines `round_corners_of_all_images()`, the third function ed in `mask.py`. Refer to the code and answer the following questions.

```
round_corners_of_all_images(directory=None):
""" Saves a modified version of each image in directory.

Uses current directory if no directory is specified.
Puts images in subdirectory 'modified', creating it if needed.
New image files are of type PNG and have transparent rounded corners.
"""

if directory == None:
    directory = os.getcwd() # Use working directory if unspecified

# Create a new directory 'modified'
new_directory = os.path.join(directory, 'modified')
try:
    os.mkdir(new_directory)
except OSError:
    pass # if the directory already exists, proceed

# Load all the images
image_list, file_list = get_images(directory)

# Go through the images and save modified versions
for n in range(len(image_list)):
    # Parse the filename
    filename, filetype = os.path.splitext(file_list[n])
```

```
    # Round the corners with default percent of radius
    curr_image = image_list[n]
    new_image = round_corners_one_image(curr_image)

    # Save the altered image, using PNG to retain transparency
    new_image_filename = os.path.join(new_directory,filename + '.png')
    new_image.save(new_image_filename)
```

In line 95, `mkdir()` creates a new directory. Explain why you think this function call needed to be embedded in a `try-except` structure.

In line 103, what is represented by `len(image_list)` ? In other words, what does that number mean?

What is the role being played by `n` in lines 103, 105, and 108?

the code using a different file name and modify it to accomplish one of the following tives. Your code should include two new functions modeled after `d_corners_one_image()` and `round_corners_of_all_images()`.

Create a function `frame_all_images(color, wide)` that makes a framed version of all pictures in a directory, where the frame is specified by a color (`r,g,b`) and has thickness `wide`.

Create a function `alter_all_images()` that makes a new version of all pictures in a directory, with the modification being of your own design.

your *Python* file in the code editor. If you were logging the IPython session, save it with `stop`.

```
]: %logstop
```

Conclusion

1. Icons on the desktop are not usually rectangular. You can see through the desktop behind their irregular edges. How is this accomplished?
2. You have 2000 images and would like thumbnails of all of them so that they will be transparent in their corners. Describe the algorithm you would use to accomplish this.
3. The code provided was divided into three functions. Describe how this made the code reuse easier.