COSC 1336 Lab: 2

Relevant reading: Sections 3.1-3.2 & 3.5

Due: Sep. 18, 2:30 pm

(Late date: Sep. 25, 2:30 pm)

50 Points

I recommend that you stop working on the first two problems (even if you are not done) and take a look at problems 3 and 4 when there are still 10-15 minutes left in lab so that you can get help if you are confused about the overall goal of the problems.

Problem 1. [**10 points**] **Tip table:** Download the file `tip_table.py`.

a. [**4 points**] First, modify the `display_total_due` function so that when it is called with the values `10.825`, `0.0825`, and `0.15`, it displays the result formatted just like this:

```
Total due with 15% tip: $12.32
```

Hint: Look at the sections in Chapter 2 on the `end` and `sep` arguments to `print` (pp. 65-67) on formatting numbers (pp. 68-69) and on formatting numbers as a percentage (p. 72)

b. [**6 points**] Now complete the program so that it reads in the bill for a meal, and then displays the total due with a tip rate of 10%, 15%, 20%, and 25%, using the `display_total_due` function. Since the tax rate should be the same for all of the function calls, you should declare a "constant" variable called `TAX_RATE` to hold the tax rate of 8.25%. Here is a sample interaction with the program:

```
Enter the bill with tax: $16.24
Total due with  10% tip: $17.74
Total due with  15% tip: $18.49
Total due with  20% tip: $19.24
Total due with  25% tip: $19.99
```

Problem 2. [**15 points**] **Average function:** Download the file `average_three_numbers.py`. First, write a function called `display_average` that receives three numbers and calculates and displays their average. After the function definition, write code that will read in three numbers from the user and then display their average using the function you just wrote.

Problem 3. [**15 points**] **Refrigerator art:** For this problem, a special Python package called `pygame` is required. The package is already installed on the lab machines in our lab, as well as in the open lab (and tutor room) on the Rio Grande campus, but it is **not** installed on the other campuses. If you would like to install it on your own computer, go to `http://www.pygame.org` and contact me if you run into trouble.

To complete the problem, do the following steps:

a. Make a folder somewhere (such as your H drive) called `Lab2Drawings`.

b. Download the files `drawing_functions.py` and `draw_picture.py` and put them both in the `Lab2Drawins` folder you just made.

c. Open the `draw_picture.py` file in IDLE. There is no reason to open the `drawing_functions.py` in IDLE unless you are interested.

d. If you run the `draw_picture.py` file, you should see a black window with a yellow smiley face in it. Your job for the lab will be to make your own drawing instead of the smiley face.

e. Delete or modify the code between the line of hash marks (`######...`) to draw a new picture. Here is a list of the functions you can call along with a short description of each. The x-y coordinates assume that the origin of the x-y plane is at the lower-left corner of the window, with `x` specifying the distance across the screen and `y` specifying the height.

- `set_color(color)`: Sets the color of the pen. The color will be used until the `set_color` function is called again with a different color. The color can be chosen from one of the following: `WHITE, GREY, BLACK, BROWN, RED, ORANGE, YELLOW, GREEN, FOREST_GREEN, CYAN, BLUE, MIDNIGHT_BLUE, PURPLE, MAGENTA`.
- `set_background_color(color)`: Sets the background color of the window. The options are the same as for `set_color`. This function should be called before anything else is drawn, and should only be called once.
- `draw_line(x1, y1, x2, y2)`: Receives four coordinates: the x-y coordinate of one end of the line and the x-y coordinate of the other end.
- `draw_rectangle(upper_left_x, upper_left_y, width, height)`: Receives the x-y coordinate of the upper-left corner of the rectangle, and the width and height of the rectangle.
- `draw_triangle(x1, y1, x2, y2, x3, y3)`: Receives the x-y coordinates of the three vertices of the triangle.
- `draw_circle(x, y, radius)`: Receives the x-y coodinate of the center of the circle and the radius of the circle.
- `draw_ellipse(top_left_x, top_right_y, width, height)`: Receives the dimensions of a rectangle (see `draw_rectangle` above) and draws an ellipse inside the rectangle.
- `draw_partial_ellipse(top_left_x, top_left_y, width, height, start_angle, end_angle)`: Receives the dimensions of an ellipse (see `draw_ellipse` above) and two angles, where the angle is measured from the x-axis. That is, zero degrees is straight out to the right, and 90 degrees is straight up.
- `draw_filled_rectangle`, `draw_filled_triangle`, `draw_filled_circle`, `draw_filled_ellipse`: These functions receive the same arguments as their non-filled counterparts, but instead of drawing the outline of the shape, they draw the shape filled in with the pen color.

Note that the functions for drawing shape outlines, as well as the `draw_line` and `draw_partial_ellipse` functions, also have an optional parameter that specifies the thickness of the line. The default is 1 and larger numbers make the line thicker.

Problem 4. [10 points] Art functions: For this problem, you will make a drawing again, but this time you must define at least one function and call it at least once. First, download the file `draw_picture_with_functions.py` and save it in the `Lab2Drawings` folder.

If you'd like, you can simply refactor the code you wrote for the last problem so that it is all inside a function, and then call that function, passing it no arguments. Such a solution will get you full credit on the problem. But you will learn more if you try to take better advantage of the function to draw a more interesting picture.

For instance, in the example code that is in the file when you download it, I took the code for drawing a smiley face, and put it in a function that would allow me to set the x-y position of the center of the face, as well as the face's size, by passing different values into the function. Then I called it three times with randomly chosen values, creating a drawing with three randomly sized and positioned smiley faces.

You might consider doing something similar with your drawing or a piece of your drawing. For example, if you drew a picture that included a tree, make a `draw_tree` function that allows you to set the location of the tree, and then make multiple trees in your drawing.

When you have completed the last two problems, zip up the `Lab2Drawings` folder. On Windows, you can do this by right-clicking the folder and choosing **Send To → Compressed (zipped) Folder**. The resulting .zip file is what you should submit on Blackboard.

For this lab, you should submit the following files:

- `tip_table.py`
- `average_three_numbers.py`
- `Lab2Drawings.zip`