Problem Set 2, CSCI1101, Sections 1—3, Fall 2017

Due: at 6PM, Friday, September 29th 2017, on GitHub.
Number of pages: 6
Extra Credits: Up to 4 Points.

Note on the format of your files: Each one of your files must start with the following lines, followed by your program/code:

```
# pylint: disable=c
# Problem Set 2, Part I (if the file is for part I); otherwise, Part II
# Name: your last name, your first name
# Collaborators: last name1, first name1; last name2, first name2; etc.
```

**You're welcome to collaborate with anyone you want (or post your questions to Piazza). However, when it comes to sitting down and writing your code, please write your own code. Thank you.

**If you encounter any problems in completing the assignment or in the submission process, please don't hesitate to ask for help (come to the OHs, and/or post your questions to Piazza).

Part I.A: In ***part_i.py*** file, define a function, named `is_divisible`, that takes in two integer arguments and returns `True` if the first argument is divisible by the second argument. Otherwise, it returns `False`.

For example, calling the function, with 4 and 2 passed to it (in this specific order), would return `True`. In other words, `result = is_divisible(4, 2)` defines a new variable `result` whose value is `True`. However, calling the function, with 2 and 4 passed to it (note we just changed the order), results in `False`. i.e., `result = is_divisible(2, 4)` defines a new variable `result` whose value is `False`. Please don't print anything inside the function definition; we'll use this function in **Part I.B** below.

Note: The integer *n* is divisible by the integer *m* if there exists [at least] an integer *k* such that $n = mk$. Now, using this definition, you should determine if 0 is divisible by 0, if 1 is divisible by 0, if 0 is divisible by 3, etc. Your `is_divisible` function should **not** throw any errors for any pair of integer arguments; in particular, it should not throw errors for any of the following pairs:

- 0 and 0
- 1 and 0
- 0 and 3

Part I.B: In your ***part_i.py*** file, write a program that prompts the user to input two integers, one at a time. The program must determine if either number is divisible by the other number and print a message to that effect. **You must use the function you defined in Part I.A in this file**. Also, assume that the user enters only integers. Let's look at some samples that we get by running the ***part_i.py*** file.

Sample Outcome:

```
Please enter your first integer: 1

Please enter your second integer: 2

The second number is divisible by the first one.
```

Let's run the program again to see another sample outcome:

```
Please enter your first integer: 20

Please enter your second integer: 4

The first number is divisible by the second one.
```

Let's run the program again to see another sample outcome:

```
Please enter your first integer: 21

Please enter your second integer: 4

Neither number is divisible by the other.
```

Let's run the program again to see another sample outcome:

```
Please enter your first integer: -5

Please enter your second integer: 5

Both numbers are divisible by each other.
```

Part II (Many Faces): Open the ***manyfaces.py*** file in Atom. Run the whole file and a separate window/application should pop up, where you see a yellow smiley face (this pop up application has an icon looking like a rocket with the Python logo on it). Let us know if this does not work on your computer.

Now, we want to write a program that asks the user how they're doing and then draws a face based on the user's feeling. For this part, our program must react to at least three different emotions (first, the program lets the user know which emotions it can react to, and then it prompts the user for their emotion. If the user enters an emotion that the program doesn't react to, the program must print a message to that effect). Below, I'm showing faces for *happy*, *sad*, and *pensive*; however, feel free to choose any three different emotions that you like.

Feel free to change any features of the face if you like, e.g., change the eyes, add eyebrows, change the color of the face based on the emotion, add ears, hair, chin, etc. (**please be as creative as possible, all the 100+ students in the three sections are going to vote to choose the three coolest programs**). Also, note that Python understands quite a few color names. Please see the attached instructions (PSet02 Note.pdf) on how the `create_oval`**,** `create_arc`**,** and `create_line` functions work.

Your program must give three different results for at least three different feelings. You'll receive an extra credit per any additional feeling your code responds to (the face must be different for any different feeling, and you may receive up to 2 extra points on this problem set if your code responds to five different emotions (or more)).

I've changed the eyes and added eyebrows. You don't need to create more elaborate eyes as in these drawings. **However, if you do, you'll receive 2 extra points on this pset in addition to the other extra points stated in the previous paragraph**. If you want to draw more elaborate eyes such as the ones in the following pictures, please define a function to draw an eye at a specific location on the face (this function accepts `ps2_canvas` and 4 numbers representing the coordinates of the eye). When you have defined the `draw_eye` function, call the function and pass appropriate arguments to it to draw the left eye. Then, call the function again with appropriate arguments passed to it, this time to draw the right eye. **Please don't simply copy/paste a block of code written for one eye to create the other one, no extra credits will be given in that case. Write a function instead and call it twice to draw both eyes**.

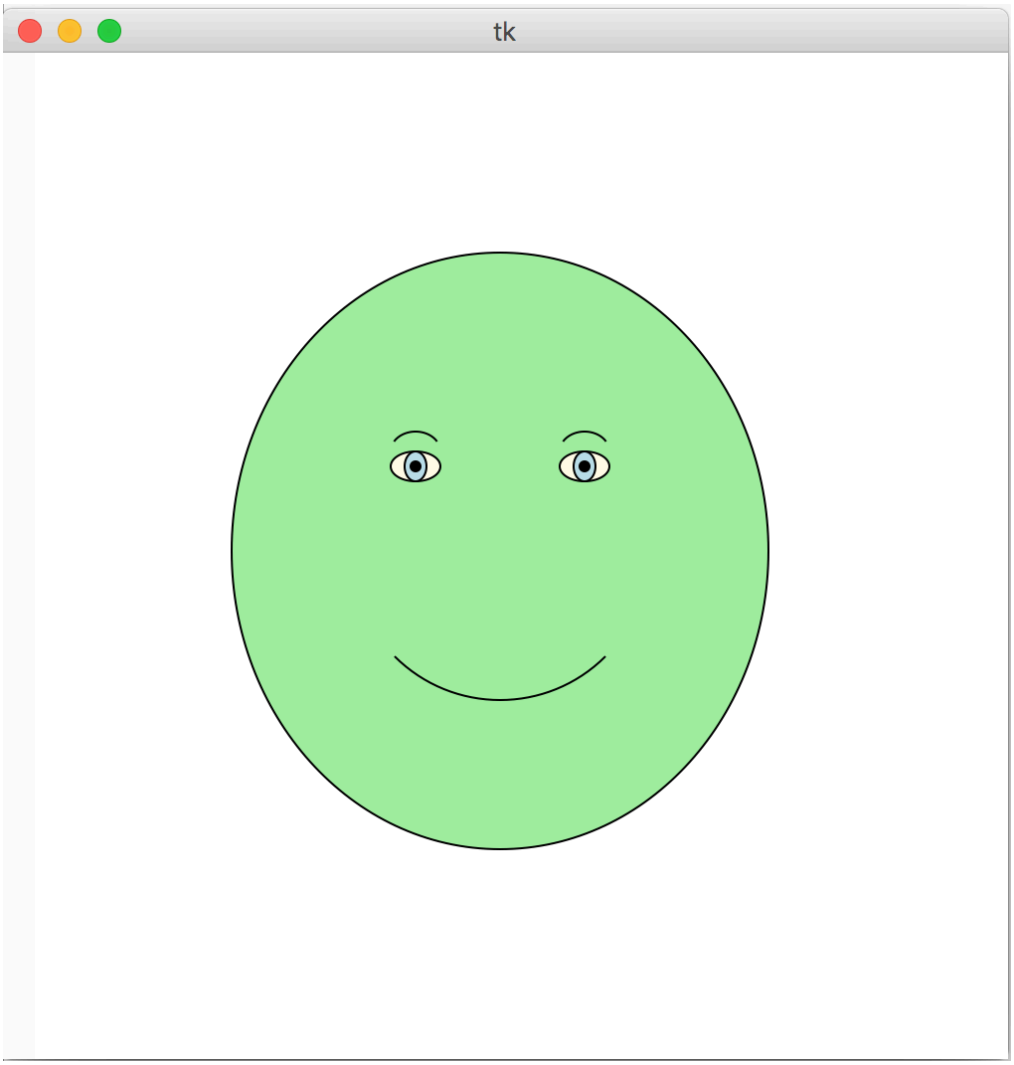
Feel free to change the greeting message and/or the message shown when the user inputs an emotion that the program doesn't recognize.

Below is the result of a sample program:

The program greets the user and lets them know which emotions it can react to. It then asks the user a question to see how they're doing. The user then enters the answer **happy**.

```
Hello, there. I'm a robot that understands these emotions:
happy, sad, and pensive.
How are you doing today? happy
```

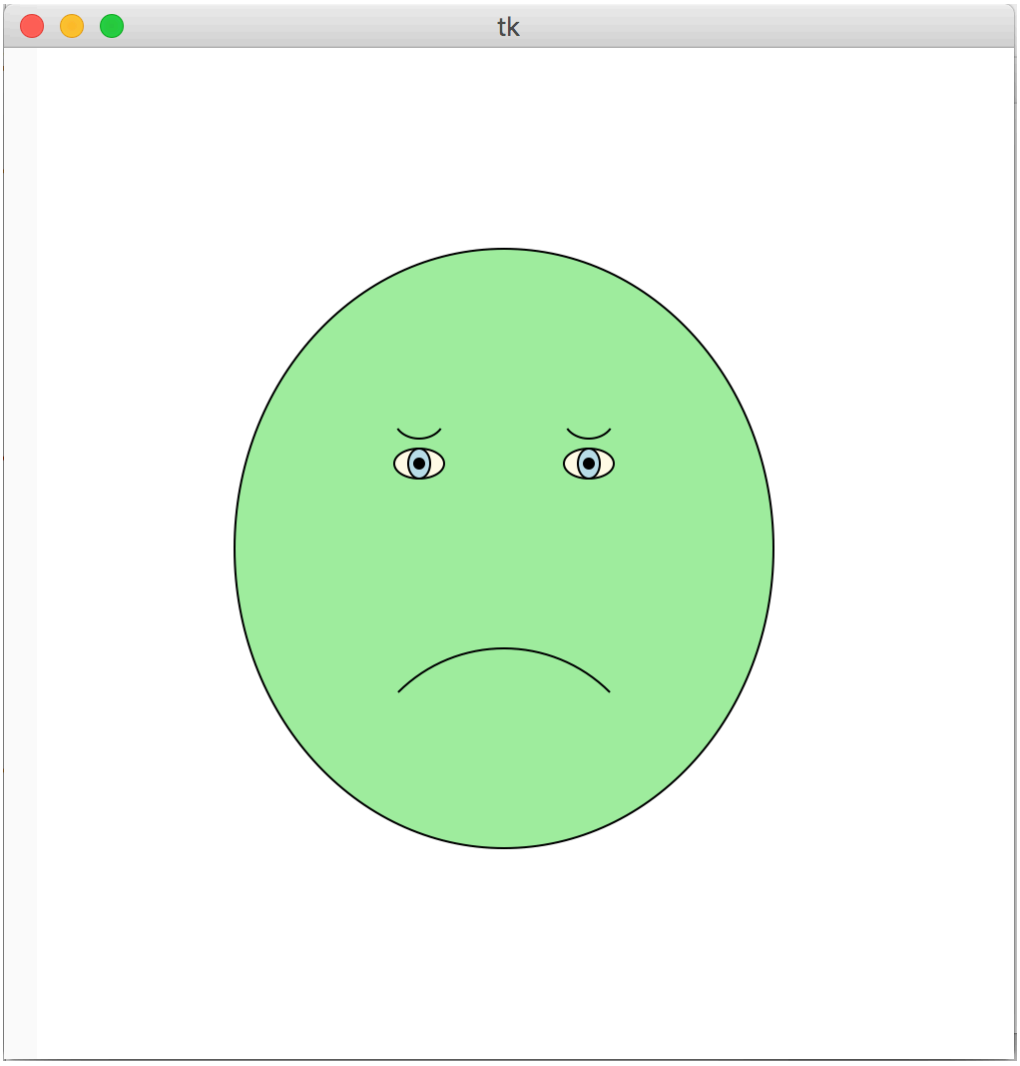
Then the program draws the following face as a response to the user's happy emotion.



Let's run the program again, this time, the user says they're **sad**.

```
Hello, there. I'm a robot that understands these emotions:
happy, sad, and pensive.
How are you doing today? sad
```

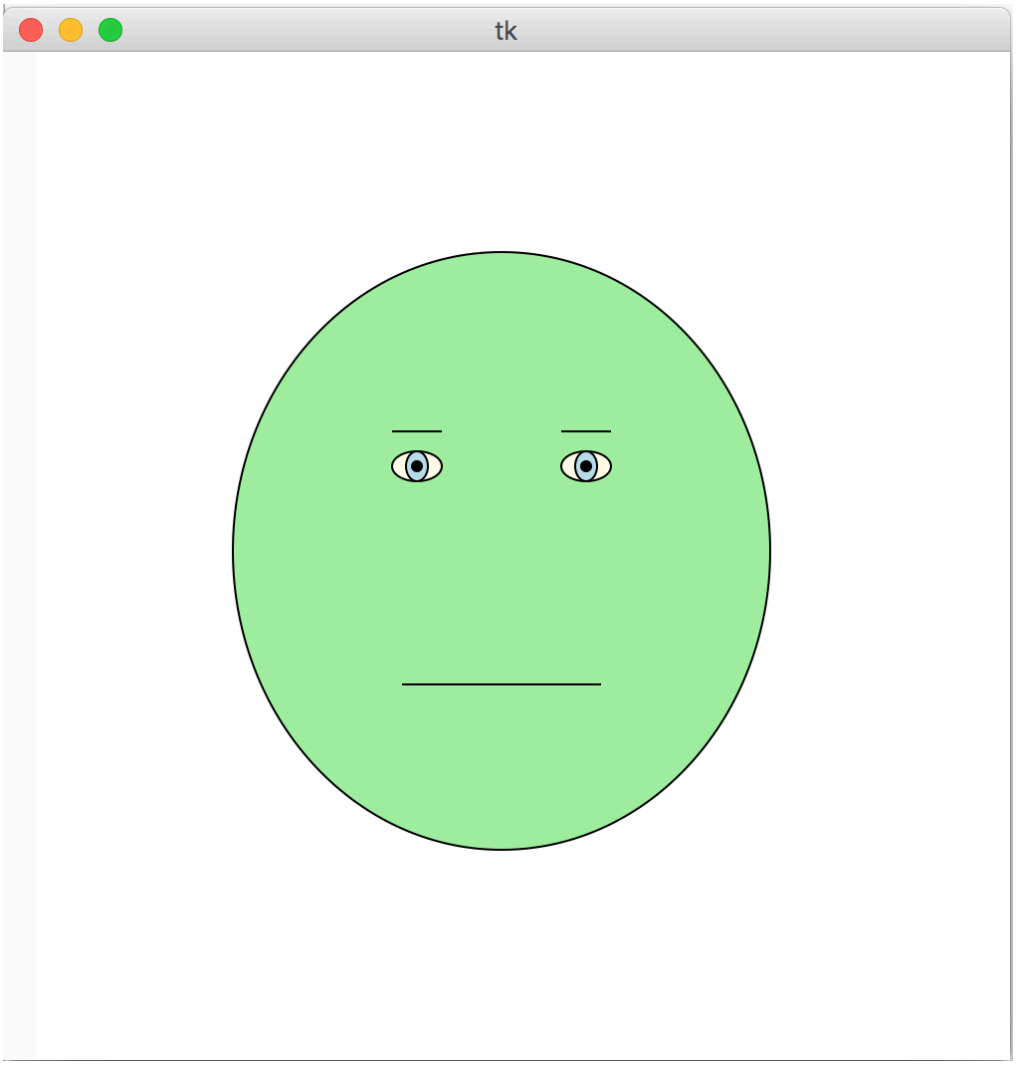
The program draws the following face:



Let's run it again. This time, let's enter **pensive** as our answer:

```
Hello, there. I'm a robot that understands these emotions:
happy, sad, and pensive.
How are you doing today? pensive
```

The program draws the following face:



Let's run the program once again. This time, let's enter a response to which the program doesn't know how to react:

```
Hello, there. I'm a robot that understands these emotions:
happy, sad, and pensive.
How are you doing today? mad
Sorry, I don't recognize your input.
```