Lab 2:

(68 pts) Due Wed, Mar 4

This is a paired programming lab: In this lab you will work in pairs. In lab, you will choose your partner for the next two weeks. Get your partner's name and email address. Most likely you will not complete this lab during the allotted lab time and will need to meet with your partner outside of lab. While working on this lab, one person should work the computer and the other person should "navigate", or coach the computer operator. Every 20 minutes the two partners should switch who is working the computer and who is coaching.

When you turn in the lab, both partners should turn in the lab, and the lab must have both students' names on it. **DO NOT FORGET YOUR PARTNER'S NAME**. Doing so will result in a penalty of 25%. You should submit your lab via Canvas.

The first part of this lab (involving the python code) should be saved in a python file (with a .py extension) and turned in as lab2.py. The very last part of this lab should be saved in lab2turtle.py. All files should have both partners' names on them and should be submitted by only one partner via Canvas.

Contents

Part 1 (59 pts):

Problem 1 (TriangleCheck)(3 pts)

Write a program that takes as input parameters 3 integers: the three angles of a triangle and checks whether it's a valid triangle or not. The function should return "yes" or "no"

Problem 2 (AbsDiff) (4 pts) Write a function that takes two integers as input parameters and returns the absolute difference between the two integers.

Problem 3(Min)(3 pts) Write a function that takes as input parameters 2 integers. It returns the minimum of those two integers.

Problem 3b(MinOfFive) (5 pts) Using the above function, write a function that takes as input 5 integers and returns the minimum of the 5 integers

Problem 4 (MathFn) (4 pts) Given the math function:

If_fun(x,y) =

$(y* 2)^3/x$	if y is less than 5 and x is not equal to 0
$(y*3)^2/x$	if y is greater than 5 and x is not equal to 0
(y-2)/x	if y is equal to 5 and x is not equal to 0
-1	otherwise

Write the function in python (including comments)

Problem 5(MonthDays)(4 pts) Write a function that takes as an input parameter the number of a month (i.e., 11 would be for November, 3 would be for March, etc.). The function should return the number of days in that month this year (so February gets 29 this year).

Problem 6(Calories) (3 pts) Including comments, write a function that calculates and returns how many calories you have burned when running. To calculate the number of calories burned, use the following formula:

w/2.2 * 7.5 * m/60

Where:

w is your weight (in pounds)
m is the number of minutes you ran

Problem 7(CarPayments)(7 pts) Calculating your monthly car payments. Write a function (including comments) that calculates your monthly car payments. To calculate your monthly car payments, you'd use the following formula:

$$\frac{P\,(\frac{r}{12})}{(1-\left(1+\frac{r}{12}\right)^{-m})}$$

Where:

- P is the principle (the amount you're borrowing to pay for the car, a.k.a. the amount the car cost),
- r is the interest rate,
- m is the number of months in which you intend to pay it off.

So, for instance (and yes, this can be one of your test cases), if you borrowed $15,000 at a rate of 7% and wanted to pay it off over 3 years (or 36 months), you'd calculate your monthly payments as follows:

$$\frac{15000\ (\frac{0.07}{12})}{(1-\left(1+\frac{0.07}{12}\right)^{-36})}$$

Or 463.16 per month.

Problem 8 (Electricity)(7 pts) Write a function that takes as an input parameter an integer representing the number of electricity units and calculates the total electricity bill according to the given condition:
For first 50 units, the cost is 0.50/unit
For next 100 units the cost is 0.75/unit
For next 100 units after that the cost is 1.20/unit
For unit above 250 the cost is Rs. 1.50/unit

It returns the bill (the total cost)

Problem 9(MagicNum) (6 pts) Write a magic number function: The function takes as an input parameter a positive integer. In the function, add 1 to the number. Multiply that number by 2. Add 4 to that number. Divide this number by 2. Now subtract the input parameter from this number. Return this number.

Problem 9b(MagicCheck)(3 pts) Now write a second function that takes as an input parameter an integer. This second function should print, "Is the number 3?". It should then call the first function and see whether the number returned is 3 or not. If it is, this second function should return, "I guessed it!" Otherwise, it should return "I'm not a prognosticator!" (Note: it should always return "I guessed it!")

Problem 10(FizzBin)(5 pts) Fizzbin (loosely interpreted for this exercise): Write a fuction that works as follows: It takes as an integer. If the integer is evenly divisible by 5, it returns "fizz" If it is evenly divisible by 3, the function returns "bin" If the integer is evenly divisible by both 5 and 3, it returns "Bizzyfin". Otherwise it returns "wuzzle"

Problem 10b(FizzBin2) (5 pts) Now write a function that takes as input parameters 6 integers. It uses the function above to create the appropriate translation for those numbers. For instance, if the sequence of numbers is:

3,7,5,15,30,2

The returned value from the second string should be: "binwuzzlefizzbizzyfinbizzyfinwuzzle"

Part 2: Intro to Turtle: (9 pts)

Turtle is a python library that provides graphics primitives. To use it, open a second file called lab1turtle.py. The very first line of this file needs to "import" the library, or make the library's functions available to this particular file. So at the top of the file, the first line should be:

```
import turtle
```

Problem 2.0(TurtleDraw)(1 pt) Now add the following lines to the file:

```
turtle.forward(50)
turtle.right(90)
```

```
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
```

Now save the file and run it. A separate window should pop up and a square should be drawn. Each time the command turtle.forward(50) is issued, the "turtle" should move forward 50 pixels. turtle.right(90) rotates the direction in which the turtle is going by 90 degrees.

Problem 2.1(square) (3 pts): Write a function below the lines of code (above) that takes as input a number, and draws a square with sides the length of that number. Note: this is an example of a function that does not return a value. So the last line of this function should simply be:

```
return()
```

Turtle colors

You can change the colors turtle is using to draw a line. You can use different colors to fill the object you drew as well. Colors on the computer are represented in terms of the amount of Red, Green, and Blue (RGB values). In turtle, to set the color to red, I'd use the following command:

```
turtle.color(1,0,0)
```

To set the color to blue, I'd use:

```
turtle.color(0,0,1)
```

and to set it to green, I'd use the following:

```
turtle.color(0,1,0)
```

Thus the first number indicates whether we have red or not, the second whether we have green or not, and the third whether we have blue or not. We can have more than one color:

```
turtle.color(1,0,1)
```

Means we've got red and blue, and will show up as purple.

```
turtle.color(0,1,1)
```

has both green and blue, and results in cyan (blue-green) . And finally,

```
turtle.color(1,1,0)
```

means we've got red and green and results in YELLOW! Yep, yellow.

Finally, if we've got no red, green, or blue, we have black. So the following:

```
turtle.color(0,0,0)
```

Results in black.

And if we have all three colors (red, green, and black), we have white. So the following:

```
turtle.color(1,1,1)
```

Results in white.

Problem 2.20 (colors)(1 pt) To see colors, try the following (below your last function):

```
turtle.color(1,0,0)
turtle.begin_fill()
turtle.forward(100)
turtle.left(90)
turtle.forward(20)
turtle.left(90)
turtle.forward(20)
turtle.right(90)
turtle.forward(20)
turtle.left(90)
turtle.forward(60)
turtle.left(90)
turtle.forward(20)
turtle.right(90)
turtle.forward(20)
turtle.left(90)
turtle.forward(20)
turtle.end_fill()

turtle.color(0,0,0)
turtle.up()
turtle.forward(10)
turtle.down()
turtle.begin_fill()
turtle.circle(10)
turtle.end_fill()

turtle.setheading(0)
turtle.up()
turtle.forward(90)
turtle.right(90)
turtle.forward(10)
turtle.setheading(0)
turtle.begin_fill()
turtle.down()
turtle.circle(10)
turtle.end_fill()
```

Save the file and run it. We've got a couple new commands in here:

```
turtle.begin_fill()
turtle.end_fill()
```

These commands indicate that the shape we are drawing between the two commands should be filled with the color set in turtle.color.

```
turtle.up()
```

lifts the pen off the page. With this command you can move the turtle around the screen without drawing a line.

```
turtle.down()
```

places the pen back on the screen, so from that point on the pen will draw on the page with each movement.

```
turtle.circle(10)
```

Draws a circle with a radius of 10.

And finally,

```
turtle.setheading(0)
```

Sets the heading to 0^{th} degree, meaning, in this case, facing right.

Problem 2.3 (ColoredObject)(4 pts):

Write a function that uses input parameter(s) to determine what color to create an object. Within the function, draw an object, and fill it with the color indicated by the input parameter(s). You can draw whatever you feel like drawing – a robot, a flower, a lightbulb, anything you want.

Hint on the color: you can either use 3 input parameters, one for red, one for green, and one for blue, or you can use one input parameter that might be 0- 7, with 0 being black, 1 being red, 2 being green, 3 being blue, 4 being purple, etc.

To Turn In (via Canvas):

- Part 1 (either as comments in lab1.py or in a separate MS Word document)
- Part 2 and 3 in lab1.py
- Part 4 in lab1turtle.py