15-110 F25 Hw1 - Written Portion

Name:

AndrewID:

Complete the written problems in the fillable PDF, or print out the PDF, write your answers by hand, and scan the results. Also complete the programming problems in the starter file hw1.py from the course website.

When you are finished, upload your hw1.pdf to **Hw1 - Written** on Gradescope, and upload your hw1.py file to **Hw1 - Programming**. Make sure to check the autograder feedback after you submit!

Written Problems

#1 - Data Representation - 10pts

Can attempt after Data Representation lecture

For each of the following problems, you must **show your work** to receive full credit.

Convert the following number from binary to ascii. You may wish to refer to this chart: www.asciitable.com

01001111 01001011

Work:	
Answer:	

Convert the following three numbers from binary to decimal. Then enter the decimal numbers into the respective R, G, and B values here: https://www.w3schools.com/colors/colors_rgb.asp .

What color does this binary represent?

R: 11100110 **G:** 11110000 **B:** 00111100

Work:	
Answer:	

#2 - Function Calls - 8pts

Can attempt after Function Calls lecture

What does each of the following function calls **evaluate** to? In other words, what is the *returned value* of each function call?

Don't just copy and run the code to find the answer- try to determine what the answer is yourself!

```
int(15.11)
```

```
print("Result:", 5 + (3 * 2))
```

```
round(abs(-19.286), 1)
```

```
math.ceil(6.2) # assume we already imported math
```

#3 - Function Components - 9pts

Can attempt after Function Calls lecture

Each of the following code snippets contains a function call. Write the **function name, argument value(s),** and **returned value** of each call in the relevant table. If there is no name / argument / returned value, leave the space blank.

```
x = 3
y = 2
z = pow(x, y)
```

Name	Argument(s)	Returned Value

```
print("Result:", 10 ** 2)
```

Name	Argument(s)	Returned Value

```
import random
num = random.random()
```

Name	Argument(s)	Returned Value

#4 - Variable Scope - 8pts

Can attempt after Function Definitions lecture

Consider the following code:

```
import random
import tkinter

def randomPos(windowSize):
    tmp = random.randint(0, windowSize)
    return tmp

def randomSize():
    scale = random.randint(1, 5)
    return scale * 20

def drawCircle(canvas, size, color):
    x = randomPos(400)
    y = randomPos(400)
    canvas.create_oval(x, y,
                       x+size, y+size,
                       fill=color)

# ignore graphics setup code here
root = tkinter.Tk()
canvas = tkinter.Canvas(root,
                        height=400,
                        width=400)
canvas.pack()
# stop ignoring now

circleSize = randomSize()
circleColor = "purple"
drawCircle(canvas, circleSize,
           circleColor)

root.mainloop() # ignore this line too
```

For the following two questions, ignore the graphics set-up code (root, canvas, etc).

Are there any **global** variables in this code? If yes, what are they?

Are there any **local** variables in this code? If yes,
a) what are they, and
b) which functions are they local to?

#5 - Function Call Tracing - 15pts

Can attempt after Function Definitions lecture

Consider the following code:

```
1  def a(x):
2      tmp = x + 2
3      x = "wow"
4      print("a:", x)
5      return tmp
6
7  def b(x):
8      x = x + 3
9      y = x * 2
10     print("b:", a(y))
11     return x
12
13 tmp = 5
14 result = b(tmp)
15 print("End:", tmp, result)
```

Write in the space below what this block of code will print once it stops running.

List all the function calls that occur in this code block with their **name, argument value(s),** and **returned value**. If there is no name / argument / returned value, leave the space blank. You might not need to use the whole table. **Do not** include any calls to built-in functions (like `print`) in the table.

Function name	Argument value(s)	Returned value

Programming Problems

For each of these problems (unless otherwise specified), write the needed code directly in the Python file, under the comment and print statement that correspond to the problem. Do not delete the provided print statements - we're using them to autograde.

If you find yourself struggling to get your code to work, remember your resources! Office hours in particular are useful for debugging problems.

Before you submit: click 'Run current script' to make sure your code runs without raising an error message. Any syntax or runtime errors left in the code will result in a deduction on the assignment grade. You should do this for all future programming assignments as well.

Cautionary note: the autograder does not handle calls to `input()` well. Please do not use this function unless a problem specifically asks for it to be used.

#1 - Data Types - 10pts

Can attempt after Programming Basics lecture

Under the line `print("---1---")`, write Python code to:

1. Assign the integer 15 to the variable **`a`**.
2. Assign the float 3.14 to the variable **`b`**.
3. Assign the string "20" to the variable **`c`**.
4. Assign the boolean `True` to the variable **`d`**.
5. Evaluate 5 minus 1.7 and assign that expression to the variable **`e`**.
6. Check whether 8 is less than 5 and assign that expression to the variable **`f`**.
7. Reassign the variable **`a`** to hold the value 45.
8. Concatenate **`c`** and "21" and assign the result to variable **`g`**. Don't change the value in **`c`**.

Feel free to print any of these variables to check your work.

#2 - Printing - 10pts

Can attempt after Programming Basics lecture

Write code at the top level of the file to match the following algorithm.

1. Assign either the string `Kelly` or the string `Franceska` to **`prof`**.
2. Assign a string holding your 110 TA's name to **`ta`**. If you have two TAs, choose one.
3. Write a single print statement that greets both your professor and your TA by name. The statement must use the variables **`prof`** and **`ta`**, as well as at least one additional string.

#3 - Using Function Calls - 10pts

Can attempt after Function Calls lecture

Write code at the top level of the file to match the following algorithm:

1. Import the **`random`** library and the **`math`** library
2. Generate a random integer between [1, 360] and store it in the variable **`x`**.
3. Convert the integer value in **`x`** to a radian number with a function from the math library and store the result in the variable **`r`**.
4. Write a single print statement that briefly describes the relationship between the values in **`x`** and **`r`** (that one number is the other in radians/degrees). The print statement should use both variables and at least one additional string.

#4 - Libraries (Graphics) - 10pts

Can attempt after Function Calls lecture

Add code under the comment `# draw fence here` so that it draws a simple version of the CMU Fence on the canvas. If you have not yet heard of the Fence, learn more here: www.amusingplanet.com/2014/09/the-fence-of-carnegie-mellon-university.html

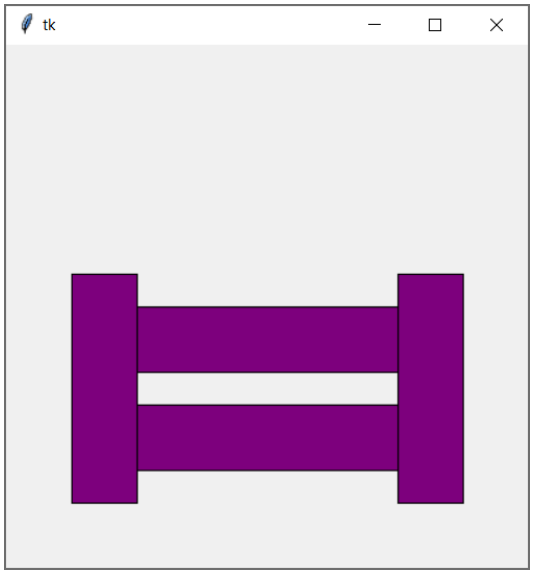
Note: because graphics require set-up code, we have included graphics setup code in the file for you. Do not delete this code!

Your Fence must meet the following basic requirements, but otherwise you may customize it as much as you like.

- The Fence must have at least two columns, with one column ending the left side of the Fence and one ending the right
- The Fence must have two cross-boards that connect all the columns
- There must be a gap above the top cross-board, between the cross-boards, and below the bottom cross-board
- The Fence must be painted at least one color

Otherwise, you're encouraged to 'paint' your fence with your own design or message! You can find bonus instructions on graphics here: www.cs.cmu.edu/~110/slides/week2-graphics.pdf

Here's an example of a simple Fence that meets the requirements:



#5 - Defining Functions - 10pts

Can attempt after Function Definitions lecture

Write a function that implements the following algorithm, which finds the slope of a line between two points.

- The function's name is **slope**
- The function takes four arguments: **`x1`**, **`y1`**, **`x2`**, and **`y2`**, in that order.
- The function should follow this algorithm:
    a. First, compute the difference in y values (using subtraction) and assign it to the variable **`diffY`**.
    b. Second, compute the difference in x values and assign it to the variable **`diffX`**.
    c. Third, compute the slope (**`diffY`** divided by **`diffX`**) and assign it to the variable **`m`**.
- The function should return the variable **`m`**.

After the function definition, write a print call that directly calls slope on four different numbers and displays the result.