CS Scholars - Programming Hw1 - Written
Due Date: Friday 07/07 EOD

Name:

AndrewID:

For full credit on the assignment, complete either all Review + Core problems (#1-#10) or all Core + Spicy problems (#1-4 & #8-13).

Bonus problems are related to the Advanced Track content, and are optional.

Core Problems - Written

#1 - Writing Algorithms

Can attempt after Introductions and Algorithms lecture

In this problem, you will write plain-language algorithms (not code!) at three levels of abstraction. Assume all of your instructions will be provided verbally (no pictures). **Do not write more than 100 words per question** (and you can write much less!).

First, write an algorithm at a **low** level of abstraction that instructs a person on how to write the capital letter L. Assume the person you are instructing has almost no prior knowledge- they know directions (up/down/left/right), but nothing else about what pen and paper are, or how to write.

Second, write an algorithm at a **medium** level of abstraction that instructs a person on how to write the word 'ALL' in English, in all capital letters. This time you can assume the person you're instructing has a little more prior knowledge - what pen and paper are and how to use them, how to draw basic shapes, etc. - but you still should not assume they know how to write.

Finally, if you wanted to provide an algorithm on how to write the word 'ALL' in English at a **high** level of abstraction, what **additional starting knowledge** would you give the person being instructed?

#2 - Reading Code

Can attempt after Programming Basics lecture

Assume the following lines of code are written in the editor and run as a script. Your job is to figure out what will be shown in the interpreter as a result. Don't just copy and run this code to find the answer- try to determine what the answer is yourself!

Next to each line in the table, write what will appear in the interpreter based on that line of code running from the editor. If nothing should appear, leave the box blank. We've filled in the first two entries for you as an example.

`Code Line`	**Interpreter Output**
`age = 18`	
`print("test")`	test
`print("age")`	
`print(age)`	
`age + 5`	
`print(age / 3)`	
`print(age // 3)`	
`print(age < 18)`	
`# print(age * 3)`	
`print(age - 10 * 2)`	
`age = age + 1`	
`print("Age:", age)`	
`age -= 2`	
`print(age)`	

#3 - Function Components

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. If there is no name / argument / returned value, leave the space blank.

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

Name	Argument Value(s)	Returned Value

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

Name	Argument Value(s)	Returned Value

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

Name	Argument Value(s)	Returned Value

#4 - Function Call Tracing

Can attempt after Function Definitions lecture

Consider the following code:

```python
def a(x):
    tmp = x + 2
    x = "wow"
    print("a:", x)
    return tmp

def b(x):
    x = x + 3
    y = x * 2
    print("b:", a(y))
    return x

tmp = 5
result = b(tmp)
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

Each of these problems should be solved in the starter file available on the course website. They should be submitted to the Gradescope assignment Hw1 - Programming to be autograded. Make sure to check the autograder feedback after you submit!

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.

Review Problems - Programming

#5 - Basic Data Types

Can attempt after Programming Basics lecture

Write Python code at the top level of the file to do the following:

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.

#6 - Basic Function Calls

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`** by using the `random.randint` function.
3. Convert the integer value in **`x`** to a radian number using `math.radians` and store the result in the variable **`r`**.
4. Write a single print statement that describes the relationship between the values in **`x`** and **`r`** (that one number is the other in radians/degrees). The print statement must use both variables and at least one additional string.

#7 - `slope`

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`**.
- 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`**.

Core Problems - Programming

#8 - Print a Greeting

Can attempt after Programming Basics lecture

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

1. Assign the string `Kelly` to **`prof`**.
2. Assign a string holding another CS Scholar student's name to **`student`**.
3. Write a single print statement that greets both Prof. Kelly and the student by name. The statement must use the variables **`prof`** and **`student`**, as well as at least one additional string.

#9 - Draw the Fence

Can attempt after Function Calls lecture

Add code to the location designated by `# write your code here #` that 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

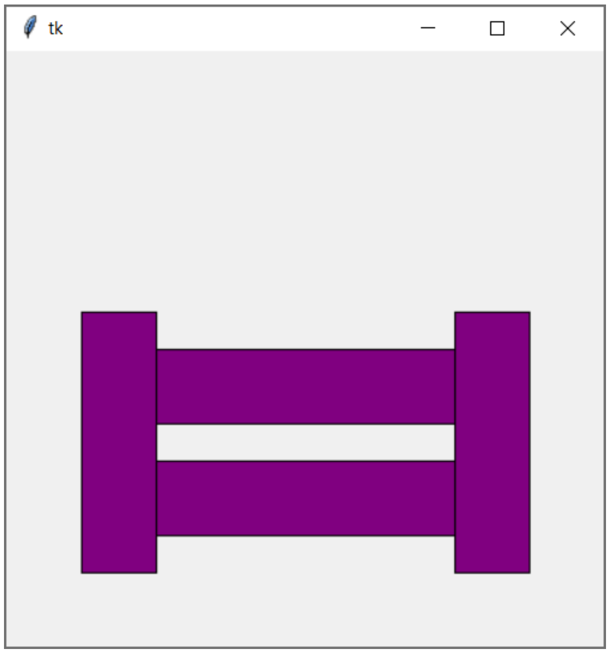
Note: because graphics require set-up code, we have included graphics setup code in the file for you. Do not delete this code!

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!

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



#10 - `addFractions`

Can attempt after Function Definitions lecture

Given four **parameters** `a, b, c, d` that represent two fractions `a/b` and `c/d`, add the two fractions together and **return** the result as a floating point number. For example, if we call `addFractions(1, 2, 3, 4)`, it should return `1.25`, because ½ + ¾ = 1 ¼ = 1.25.

Recall that you can add two fractions with the following formula:

$$\frac{x_1}{y_1} + \frac{x_2}{y_2} = \frac{x_1 y_2 + x_2 y_1}{y_1 y_2}$$

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

Spicy Problems - Programming

#11 - Get kth Digit

Can attempt after Programming Basics lecture

Set up two variables, **`n`** and **`k`**, so that **`n`** holds some integer with at least four digits, and **`k`** is some integer between 0-3. Then write code that finds the **`k`**'th digit of **`n`** from the right. You should count from 0, so **`k=0`** refers to the ones-digit, **`k=1`** refers to the tens digit, etc.

For example, with **`n=9876`** and **`k=2`**, the **`k`**'th digit from the right is 8 (the hundreds digit).

Once you've found the **`k`**'th digit, store it in a variable called **`result`**, then print out `"kth digit:"`, then the result, all on one line.

Note: you only need to run this code on one example, but it should work if you change **`n`** and **`k`** to other numbers. However, you can assume that both numbers are integers, and that both will not be negative. Try testing your code on n = 789 with k equal to 0, 1, 2, and 3 to see if you get the right digits (9, 8, 7, and 0).

Hint: you'll want to use the div and mod operators to modify the number.

#12 - Draw a Robot

Can attempt after Function Calls lecture

Add code to the location designated by `# write your code here - spicy #` to draw a robot of your own design on the Tkinter canvas.

Your robot can look like whatever you want, but for full credit it should use:

- At least 10 shapes total, including at least one oval, one rectangle, one non-rectangular polygon, and one line.
- At least 2 different optional parameters (like fill or width).

Important note: the last line of the Tkinter code, `root.mainloop()`, has been commented out since not all students will attempt this problem. Remove the comment to see your robot.

#13 - nthFibonacciNumber

Can attempt after Function Definitions lecture

Write the function `nthFibonacciNumber` which takes an integer `n` and returns the nth number in the Fibonacci sequence. For example, `nthFibonacciNumber(10)` would return `55`, as the first ten numbers in the Fibonacci sequence are 1, 1, 2, 3, 5, 8, 13, 21, 34, 55.

How can you compute this number? You could use the definition of Fibonacci numbers, which states that fib(n) = fib(n-1) + fib(n-2). But there's an easier way! The nth Fibonacci number can be calculated in a closed-form expression by using the golden ratio, φ.

$$fib(n) = \frac{\varphi^n - (1 - \varphi)^n}{\sqrt{5}}$$

Unfortunately, Python doesn't have φ built in. But you can compute the number yourself with a basic equation:

$$\varphi = \frac{1 + \sqrt{5}}{2}$$

Note that Fibonacci numbers are all **integers**, and this expression will indeed produce a number very close to an integer, but it may sometimes be a bit off due to inaccuracies in Python calculations. Make sure to **round** the result before returning it to get the answer as an integer.

Bonus Problems

Advanced Programming - External Libraries

Choose an external library from the advanced programming slides. Install this library on your machine and write a simple program (at least 5-10 lines of code) to do something interesting with that library.

Your options for libraries might be limited based on your current programming knowledge. Here's a quick breakdown of which libraries you should be able to do something interesting with based on different levels of knowledge.

Just data, variables, and function calls:

- OpenCV
- Pillow

Also function definitions, conditionals, and loops:

- Pygame
- Vpython

Also lists:

- NumPy
- SciPy
- Matplotlib
- Pandas
- Pydub

Also dictionaries:

- nltk
- Beautiful Soup

Also objects:

- Django
- Flask

You should complete this work in a separate file from hw1.py, and upload that file to Hw1 - Bonus, so that you don't break the autograder.

Advanced Computer Science - Data Representation

For each of the following problems, you must **show your work** to receive full credit. For example, to convert 0101 to decimal, you could show 0*8 + 1*4 + 0*2 + 1*1 = 4 + 1 = 5.

Convert 29 from decimal to binary.

Work:	
Answer:	

Convert 1010101 from binary to decimal.

Work:	
Answer:	

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:	