CS 171
Week 2 Lab

Overview

This lab consists of a collection of problems that you may solve in any order. To earn full credit, you must submit solutions to all problems at one time.

Important:

- You must be present in the lab session to earn credit
    - If you are recorded as absent but still complete the work, you will not earn credit. If you arrived after the CA/TA took attendance, be sure to check in so you are recorded as present/late.
    - If you miss the lab meeting, contact your instructor via email right away to request a make-up session.
- You may collaborate with at most one lab partner; Feel free to switch partners next week if you wish
- Each student must submit their own work to Gradescope for credit, even if you worked with a partner
- At most 100 points are possible. To get a combined score, submit all of your solutions at once.
- Please only use features of the language that have been covered in reading and lectures
    - Solutions which make use of language features not yet covered will not receive credit.
    - Students whose submissions repeatedly make use of additional skills not covered in the course so far may be suspected of violating academic integrity.
    - *Even if you know the language already, please work within the subset of the language covered up to this point in the course.*
- You may not use artificial intelligence tools to solve these problems.
- Your lab work is due at the end of the lab period. Late work will be accepted with a 30 point penalty if it is submitted within 12 hours of the end of your lab.

Helpful to Know:

- You may work these problems in any order
- Lab work will be graded by the auto-grader
- The last score earned will be recorded for your lab score
- To get a combined lab score, you must submit multiple files at the same time
- Lab assistants are here for your support. Ask them questions if you are stuck!

Important Information About The Auto-Grader

If a problem in this packet calls for you to write a function that the auto-grader checks for correctness, it is important that any code that you have written other than the function be written beneath an `if __name__ == '__main__':` block and indented.

Below is one example that *will* work and one example that *will not* work.

Correct:

```
def addTwoNumbers(num1, num2):
    sum = num1 + num2
    return sum

if __name__ == '__main__':
    a = float(input("Enter number 1: "))
    b = float(input("Enter Number 2: "))
    c = addTwoNumbers(a, b)
    print(a, "+", b, "=", c)
```

Incorrect:

```
def addTwoNumbers(num1, num2):
    sum = num1 + num2
    return sum

a = float(input("Enter number 1: "))
b = float(input("Enter Number 2: "))
c = addTwoNumbers(a, b)
print(a, "+", b, "=", c)
```

Both of these will work when you try them on your own computer, but the incorrect example will fail when the auto-grader attempts to run it.

In this packet, the information is relevant to problems B, C, and D.

Problem A: Total Seconds

Submission File: `lab02a.py`

Write a program that reads in seconds, minutes, and hours as integer inputs, and outputs the time in seconds only.

Important: The program should not display any message(s) when reading in values from the keyboard. Just use `input()` alone.

Ex: If the input is:

```
40
6
1
```

where 40 is the number of seconds, 6 is the number of minutes, and 1 is the number of hours, the output is:

```
4000 seconds
```

The auto-grader will give you feedback using the following symbols

Symbol	Meaning
-	The text to the right of the - is missing from your output
+	The text to the right of the + is present in our output, but it shouldn't be

Revise your solution for this problem until it earns all the points from the auto-grader.

Problem B: Yarn Skein Calculator

Submission File: `lab02b.py`

In this problem you will write a function `yarnRequired(...)` that determines the number of skeins of yarn required to complete a crochet or knitting project. A skein is the unit yarn is sold in.

In crochet and knitting, different people use various hooks/needles, tensions, and stitch types in their work, making it difficult to determine exactly how much yarn is needed to complete a given project. To estimate the yarn required, crafters often create a swatch or sample using their chosen yarn, stitch, and hook/needle. By measuring and weighing this swatch, they can estimate how much yarn is needed to produce a given quantity of finished fabric.

Crafters typically over-estimate yarn requirements by a small margin to ensure they have enough yarn to complete their project.

Yarn Skein and Fabric Swatch photo by litlnemo

Write a function `yarnRequired(...)` that receives the following floating-point values, in order:

- `skeinWeight`: The weight (mass) of yarn in one skein (in grams).
- `swatchLength`: The length of the swatch (in centimeters).
- `swatchWidth`: The width of the swatch (in centimeters).
- `swatchWeight`: The weight (mass) of yarn used to create the swatch (in grams).
- `projectLength`: The length of the project (in centimeters).
- `projectWidth`: The width of the project (in centimeters).

The function should determine the amount of yarn required to complete the project, with a 10% increase to ensure there is enough yarn, and return an integer representing the number of skeins to buy.

Hint: Python's math module includes a function, `ceil(...)` which rounds a number up to the nearest integer, if necessary, and returns the result. Calling `math.ceil(10.001)` returns 11.

A crafter who prepares a 10cm x 10cm swatch which weighs 4 grams decides to use that yarn to make a 150cm x 125cm blanket. The chosen yarn comes in skeins which weigh 170 grams each, so the crafter should buy 5 skeins to be on the safe side.

Calling `yarnRequired(170, 10, 10, 4, 150, 125)` should return a value of 5.

Problem C: Count Odd Numbers

Submission File: `lab02c.py`

Write a function countOdds() that has five integer parameters, and returns the count of parameters where the value is an odd number (i.e. evenly divisible by 2).

Ex: If the five arguments are:
1, 22, 11, 40, 37

then the returned count will be:
3

Hint: Use the modulo operator % to determine if each number is even or odd by finding the remainder when the number is divided by 2.

Caution: Please only use the language features that you have been taught up to this point. This problem can be solved without using any conditionals, lists, or loops.

Your program must define the function:
`countOdds(num1, num2, num3, num4, num5)`

The auto-grader will be checking your work by calling the `countOdds()` function, which should **return** the value, rather than printing it out.

You may find the starter code below helpful:

```
# Define your function here

if __name__ == '__main__':
    num1 = int(input())
    num2 = int(input())
    num3 = int(input())
    num4 = int(input())
    num5 = int(input())

    result = countOdds(num1, num2, num3, num4, num5)
    print(f'Total odds: { result }')
```

Problem D: Pizza Party

Submission File: `lab02d.py`

Mario and Luigi's Pizzeria needs a program to calculate:

- the number of slices a pizza of any size can be divided into, and
- the number of pizzas a customer should order so there is enough pizza for their party.

Write a program that:

- Asks the user to enter the diameter of the pizzas to be ordered (in inches)
- Asks the user to enter the number of people in the customer's party.
- Calculates the number of slices that may be taken from a pizza of the given size.
- Calculates the number of pizzas that need to be ordered, assuming that each person in the customer's party will eat an average of 3 slices.
- Display the results, that is, the number of slices each pizza yields and how many pizzas should be ordered.

To calculate the number of slices that may be taken from a pizza, you must know the following facts:

- Each slice should have an area of 14.125 inches
- The area of the pizza is calculated with the same formula you use to calculate the area of a circle

You must determine the number of slices by writing a function `slicesPerPizza(diameter)`. Given the diameter of the pizza, it will return the number of slices the pizza will have as an integer.

Note: You will need to round the results. In order to do that, you must import the math module (*hint:* you will need to use ceil() and floor()).

Here are a few sample runs:

```
Welcome to Mario and Luigi's Pizzeria

Enter the diameter of the pizzas you want to order (in inches): 14
Enter the number of people in your party: 3
For a party of 3 people you need to order 1 pizza(s).
A 14 inch pizza will yield 10 slices.
```

```
Welcome to Mario and Luigi's Pizzeria

Enter the diameter of the pizzas you want to order (in inches): 16
Enter the number of people in your party: 20
For a party of 3 people you need to order 5 pizza(s).
A 16 inch pizza will yield 14 slices.
```

Below is a template that you may use to begin work for this problem.

```
# import necessary modules
# define function

if __name__=="__main__":
    # get data from the user
    # declare other values needed for the calculations
    # calculations
    # display results
```

In order for the autograder to correctly test your function, the function must be defined above `if __name__=="__main__":` and any input and output must be done in the area below `if __name__=="__main__":`.

Don't hesitate to seek help from a lab assistant if you are having troubles with the auto-grader.