Homework 1
Print, Input, Variables, etc
CMSC 201 Section 60

Due Date: Monday, September 18th, 2023 by 11:59:59 PM
Value: 40 points

This assignment falls under the standard cmsc201 academic integrity policy. This means you should not discuss/show/copy/distribute your solutions, your code or your main ideas for the solutions to any other student. Also, you should not post these problems on any forum, internet solutions website, etc.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
"""
File:     FILENAME.py
Author:   YOUR NAME
Date:     THE DATE
Section:  YOUR DISCUSSION SECTION NUMBER
E-mail:   YOUR_EMAIL@umbc.edu
Description:
  DESCRIPTION OF WHAT THE PROGRAM DOES
"""
```

Submission Details

Submit the files under the following titles:
(These are case sensitive as usual.)
submit cmsc201 HW1 {files go here}

Problem 1 - Cookies	cookies.py
Problem 2 - Best Things	best_things.py
Problem 3 - Distance Formula	distance.py
Problem 4 - MadPyLib	mad_py_lib.py
Problem 5 - Cannon Ball	cannon_ball.py

For example you would do:

```
linux1[4]% submit cmsc201 HW1 py_coin.py geom_average.py
mad_py_lib.py gravitation.py
Submitting.py...OK
Submitting best_things.py...OK
Submitting distance.py...OK
Submitting mad_py_lib.py...OK
Submitting cannon_ball.py...OK
linux1[5]%
```

Creating Your HW1 Directory

```
linux3[1]% cd cmsc201/Homeworks
linux3[2]% mkdir hw1
linux3[3]% cd hw1
linux3[4]%
```

From here, you can use **`emacs`** to start creating and writing your different Homework 1 Python programs.

You don't need to and should not make a separate folder for each file. You should store all of the Homework 1 files in the same **`hw1`** folder.

There is a common mistake where people create a py_lib.py directory and then put a py_lib.py file inside of it. Make sure you don't do that because if you submit a directory, it copies as an empty file (it won't copy the entire directory).

Coding Standards

Coding standards for CMSC 201 can be found here.

For now, you should pay special attention to the sections about:

- Naming Conventions
- Use of Whitespace
- Comments (specifically, File Header Comments)
- Variable names. Hint, Hint

We will not grade "harshly" on Coding Standards this early in the semester, but you should start forming good habits now. Make sure to pay attention to your TA's feedback when you receive your Homework 1 grade back.

Input Validation

For this assignment, you do **not** need to worry about any "input validation."

If the user enters a different type of data than what you asked for, your program may crash. This is acceptable.
If the user enters "bogus" data (for example: a negative value when asked for a positive number), this is acceptable. (Your program does not need to worry about correcting the value or fixing it in any way.)

For example, if your program asks the user to enter a whole number, it is acceptable if your program crashes if they enter something else like "dog" or "twenty" or "88.2" instead.

For this assignment you would need if statements, which are banned, in order to detect a lot of these problems.

Here is what that error might look like:

```
Please enter a number: twenty
Traceback (most recent call last):
  File "test_file.py", line 10, in <module>
    num = int(input("Please enter a number: "))
ValueError: invalid literal for int() with base 10: 'twenty'
```

Allowed Built-ins/Methods/etc

- Declaring and assigning variables, ints, floats, bools, strings.
- Casting int(x), str(x), float(x), (technically bool(x))
- Using +, -, *, /, //, %, **; +=, -=, *=, /=, //=, %=, **= where appropriate
- Print, with string formatting, with end= or sep=:
    - '{}'.format(var), '%d' % some_int (printf formatting), f-strings
    - Really the point is that we don't care how you format strings in Python
    - Ord, chr, but you won't need them this time.
- Input, again with string formatting in the prompt, casting the returned value.
- Using the functions provided to you in the starter code.
- String operation: concatenation +, +=,
- Using import with libraries and specific functions **as allowed** by the project/homework.

You may think: "Wow... we aren't allowed to use anything!"

But never fear, as the semester progresses, a great deal of the below forbidden list will be moved into the above allowed list. But this is Homework 1, and the only things you need to know are on the allowed list. Anything else is forbidden not to prevent you from coming up with an easier solution, but mainly to prevent you from going far beyond what these problems require.

Forbidden Built-ins/Methods/etc

This is not a complete listing, but it includes:

- Comparisons ==, <=, >=, >, <, !=, in
- Logical and, or, not
- if/elif/else, nested if statements
- For loops, both *for i* and *for each* type.
- While loops
    - sentinel values, boolean flags to terminate while loops
- Lists, list(), indexing, i.e. my_list[i] or my_list[3]
    - 2d-lists if you want them/need them my_2d[i][j]
    - Append, remove
    - **list slicing**
- If you have read this section, then you know the secret word is: adventurous.
- String operations, split(), strip(), join(), upper(), lower(), isupper(), islower()
    - **string slicing**
- **Dictionaries**
    - creation using dict(), or {}, copying using dict(other_dict)
    - .get(value, not_found_value) method
    - accessing, inserting elements, removing elements.
- break, continue
- methods outside those permitted within allowed types
    - for instance str.endswith
    - list.index, list.count, etc.
- Keywords you definitely don't need: await, as, assert, async, class, except, finally, global, lambda, nonlocal, raise, try, yield
- The *is* keyword is forbidden, not because it's necessarily bad, but because it doesn't behave as you might expect (it's not the same as ==).
- built in functions: any, all, breakpoint, callable, classmethod, compile, exec, delattr, divmod, enumerate, filter, map, max,

min, isinstance, issubclass, iter, locals, oct, next, memoryview, property, repr, reversed, round, set, setattr, sorted, staticmethod, sum, super, type, vars, zip

- If you have read this section, then you know the secret word is: argumentative.
- exit() or quit()
- If something is not on the allowed list, not on this list, then it is probably forbidden.
- The forbidden list can always be overridden by a particular problem, so if a problem allows something on this list, then it is allowed for that problem.

Problem 1 - Cookies

Make sure you name your file cookies.py

Your task is to input the number of batches of cookies. A single batch of cookies requires:

2 ¼ cups (that is 2.25 cups) of flour.
2 sticks of butter
¾ cups of granulated sugar
¾ cups of brown sugar
1 teaspoon of vanilla extract
1 teaspoon of baking soda
1 teaspoon of salt
2 cups of chocolate chips

Your task is to ask (as a float) how many batches of cookies the user wants to make, then tell them the amount of ingredients that they require based on their input.

User input is generally colored **blue** to help distinguish it from the rest of the text.

```
linux4[120]% python3 cookies.py
How many batches of cookies do you want to make? 3
You Need:
     6.75 cups of flour
     6.0 sticks of butter
     2.25 cups of granulated sugar
     2.25 cups of brown sugar
     3.0 teaspoons of vanilla extract
     3.0 teaspoons of baking soda
     3.0 teaspoons of salt
     6.0 cups of chocolate chips
```

```
linux4[121]% python3 cookies.py
How many batches of cookies do you want to make? 0.5
You Need:
     1.125 cups of flour
     1.0 sticks of butter
     0.375 cups of granulated sugar
     0.375 cups of brown sugar
     0.5 teaspoons of vanilla extract
     0.5 teaspoons of baking soda
     0.5 teaspoons of salt
     1.0 cups of chocolate chips

linux4[121]% python3 cookies.py
How many batches of cookies do you want to make? 200
You Need:
     450.0 cups of flour
     400.0 sticks of butter
     150.0 cups of granulated sugar
     150.0 cups of brown sugar
     200.0 teaspoons of vanilla extract
     200.0 teaspoons of baking soda
     200.0 teaspoons of salt
     400.0 cups of chocolate chips
```

Problem 2 - Best Things

We are interested in your favorite things here. So we're going to ask the user four questions:

1) What is your favorite movie?
2) What is your favorite song?
3) What is your favorite game?
4) What is your favorite food?

Then we should output something about each one. Tell us that the movie is a great film, that you love to eat the favorite food, that you've heard great things about the game, but never heard of the song. Include the input in your output.

```
linux4[120]% python3 best_things.py
Tell me your favorite movie: The Fifth Element
Tell me your favorite song: Don't fear the Reaper
Tell me your favorite game: Homeworld
What is your favorite food: Vindaloo
The Fifth Element is a great film!
I love to eat Vindaloo!
I heard great things about Homeworld
Never heard of Don't fear the Reaper though.

linux4[121]% python3 best_things.py
Tell me your favorite movie: Blade Runner
Tell me your favorite song: Pachabel's Canon in D
Tell me your favorite game: Sim City 2000
What is your favorite food: Sushi
Blade Runner is a great film!
I love to eat Sushi!
I heard great things about Sim City 2000
Never heard of Pachabel's Canon in D though.
```

Problem 3 - Distance Formula

Ask the user for two points (in two dimensions). Then calculate the distance between those points (x_1, y_1) and (x_2, y_2) given the formula here:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Remember that the exponent operator in python is **

Enter the values as floats. You don't need to round.

```
linux4[116]% python3 distance.py
Enter x1: 0
Enter y1: 0
Enter x2: 0
Enter y2: 5
The distance between (0.0,0.0) and (0.0, 5.0) is 5.0

linux4[117]% python3 distance.py
Enter x1: 2
Enter y1: 3
Enter x2: 7
Enter y2: 4.9
The distance between (2.0,3.0) and (7.0, 4.9) is
5.348831648126533

linux4[118]% python3 distance.py
Enter x1: -2.5
Enter y1: 4.5
Enter x2: 3.1
Enter y2: 8.9
The distance between (-2.5,4.5) and (3.1, 8.9) is
7.121797525905942
```

Problem 4 - MadPyLib

We're going to do a MadLib in using this poem here:

There once was a man from Nantucket
Who kept all his cash in a bucket.
  But his daughter, named Nan,
  Ran away with a man
And as for the bucket, Nantucket.

We're going to replace the words like this:

There once was a man from _____
Who kept all his ____ in a _____.
  But his _____, named ____,
  Ran away with a ____
And as for the _____, _____.

However we're going to reuse the first two inputs again for the last line.

We'll ask for the following things:

1) A place name
2) A noun
3) Another noun
4) A familial relation (brother, sister, father, mother, etc)
5) A person's name
6) A last noun

Then fill the mad-lib poem

```
linux4[122]% python3 mad_py_lib.py
What is a place name? Nantucket
Give an example of a noun. cash
Give an example of another noun. bucket
Give an example of a familial relation. daughter
Give an example of a person's name. Nan
Give yet another noun. man

There once was a man from Nantucket
Who kept all his cash in a bucket.
     But his daughter, named Nan,
     Ran away with a man
And as for the bucket, Nantucket.

linux4[123]% python3 mad_py_lib.py
What is a place name? Alaska
Give an example of a noun. fish
Give an example of another noun. bike
Give an example of a familial relation. brother
Give an example of a person's name. Archibald
Give yet another noun. ground

There once was a man from Alaska
Who kept all his fish in a bike.
     But his brother, named Archibald,
     Ran away with a ground
And as for the bike, Alaska.
```

Problem 5 - Cannon Ball

In this program we'll calculate the distance that a cannon ball travels if shot at an angle theta and an initial velocity v_0.

The formula you need is:

$$d = \frac{v_0^2 sin(2\theta)}{g}$$

For our purposes we can use the math library, so we can for this problem only put the line at the top of our code, under our comment with our name and section information:

```
import math
```

After this you can use math.sin(x) to take the sine of a number, in radians, so to convert between degrees and radians you will need to do this:

$$d = \frac{v_0^2 sin(2\theta \cdot \pi / 180)}{g}$$

You can use the constant g = 9.8 m/s^2.

Here is one example between the Earth and the Sun

```
linux4[116]% python3 cannon_ball.py
Enter the initial velocity (V0): 50
Enter the angle in degrees that you will fire the cannon: 45
The distance the cannon ball will travel is 255.1020408163265
meters

linux4[117]% python3 cannon_ball.py
Enter the initial velocity (V0): 250
Enter the angle in degrees that you will fire the cannon: 35
The distance the cannon ball will travel is 5992.93763256319
meters

linux4[118]% python3 cannon_ball.py
Enter the initial velocity (V0): 35
Enter the angle in degrees that you will fire the cannon: 85
The distance the cannon ball will travel is 21.70602220836628
meters
```

More Submission Details

How to submit is covered in detail in Homework 0.

Once each or all of your files are complete, it is time to turn them in with the **submit** command. (You may also turn in individual files as you complete them. To do so, only **submit** those files that are complete.)

You must be logged into your account on GL, and you must be in the same directory as your Homework 1 Python files. To double-check you are in the directory with the correct files, you can type **ls**.

```
linux1[3]% ls
cookies.py best_things.py distance.py mad_py_lib.py
cannon_ball.py
linux1[4]%
```

You can see what files were successfully submitted by running:
`submitls cmsc201 HW1`

For more information on how to read the output from **`submitls`**, consult with Homework 0. Double-check that you submitted your homework correctly, since **empty files will result in a grade of zero.**

Advice for Submitting Early and Often:

You can also submit one or two files at a time, which means you can submit a file without submitting the rest of the assignment. It's better to miss one part than an entire assignment. Do not wait to submit everything until five minutes before the due date.

```
linux1[4]% submit cmsc201 HW1 mad_py_lib.py cannon_ball.py
Submitting mad_py_lib.py...OK
Submitting cannon_ball.py...OK
linux1[5]%
```

If you're re-submitting, the system will ask that you confirm you want to overwrite each file; make sure that you confirm by typing "y" and hitting enter if you want to do so.

```
linux1[5]% submit cmsc201 HW1 py_mad_lib.py
It seems you have already submitted a file named
py_mad_lib.py.
Do you wish to overwrite? (y/n):
y

linux1[6]%
```