Homework 1
Print, Input, Variables, etc

Due Date: Monday, September 19th, 2022 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 - Attendance	attendance.py
Problem 2 - Rate of Return	rate_of_return.py
Problem 3 - MadPyLib	mad_py_lib.py
Problem 4 - Standard Deviation	std_dev.py

For example you would do:

```
linux1[4]% submit cmsc201 HW1 mad_py_lib.py std_dev.py
attendance.py rate_of_return.py
Submitting mad_py_lib.py...OK
Submitting std_dev.py...OK
Submitting attendance.py...OK
Submitting rate_of_return.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, 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 - Attendance

Make sure you name your file attendance.py

Studies show that final grades in a course have a direct correlation on how often a student shows up for class (attendance). Hint, hint. The program will calculate a student's time in class and convert it from hours to minutes.

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

Your program should do these things:

1. Ask the user a student's name, and how long in hours (float) was the student in class, in that order.
2. Calculate the number of minutes the student was in class.
3. **Do not round.**
4. Display the string:

   `'NAME was in class for a total of MINS minutes.'`

   With the numbers replaced name and minutes in the proper spots in the display.

```
linux4[120]% python3 attendance.py
Attendance Checker
Please enter the student's name: Shawn
How long (in hours) was the student in class? 1.0
Shawn was in class for a total of 60.0 minutes.

linux4[121]% python3 attendance.py
Attendance Checker
Please enter the student's name: Bowen
How long (in hours) was the student in class? 1.5
Bowen was in class for a total of 90.0 minutes.
```

Problem 2 - Rate of Return

For this problem you'll determine the rate of return on an investment between when bought and then sold. For now, we are ignoring how long in between bought/sold. Make sure you name the file rate_of_return.py (all lower case). The formula for rate of return is:

R = √(V2 / V0) – 1

where:

V0 is the value when you **bought** the investment
V2 is the value when you **sold** the investment
R is the return on the investment

Your program should do these things:

1. Ask the user for floating point numbers bought and sold, in that order.
2. Calculate the return on investment.
    a. **Don't use math.sqrt()**
3. **Do not round.**
4. Display the result, but in percent form.

```
linux4[116]% python3 rate_of_return.py
Welcome, let's check your rate of return on your investment.
What investment opportunity did you buy? house
How much was the investment when you bought it? 250000
How much was the investment when you sold it? 302500
The house you bought gave you a return on investment of
10.000000000000009 %

linux4[117]% python3 rate_of_return.py
Welcome, let's check your rate of return on your investment.
What investment opportunity did you buy? car
How much was the investment when you bought it? 10200
How much was the investment when you sold it? 6800
The car you bought gave you a return on investment of
-18.350341907227396 %
```

Problem 3 - MadPyLib

In this problem you'll write a program that makes a mad-lib. A mad-lib is a "phrasal template word game" so says wikipedia. Basically there are blanks, we ask you for words, and you put them into the blanks.
Make sure to name your program mad_py_lib.py

For this program, it will ask for various nouns, adjectives, adverbs, etc... in order to complete the adlib below.

```
Today I went to the zoo. I saw a(n)__________(adjective)
_____________(Noun) jumping up and down in its tree.
He _____________(verb, past tense) __________(adverb)
through the large tunnel that led to its _____(adjective)
__________(noun). I got some peanuts and passed them
through the cage to a gigantic gray _______(noun)
towering above my head. Feeding that animal made me
hungry. I went to get a __________(adjective) scoop
of ice cream. It filled my stomach. Afterwards I had to
__________(verb) __________ (adverb) to catch our bus.
When I got home I __________(verb, past tense) my
mom for a __________(adjective) day at the zoo.
```

(running example on next page)

```
linux4[122]% python3 mad_py_lib.py
Give me an adjective: happy
Give me an animal (noun): monkey
Give me a verb, past tense: ran
Give me an adverb: noisily
Give me another adjective: large
Give me a noun: cage
Give me another noun: tray
Give me another adjective: large
Give me a verb: run
Give me another adverb: quickly
Give me another verb, past tense: thanked
Give me another adjective: wonderful

Today I went to the zoo. I saw a(n)happy monkey jumping
up and down in its tree.He ran noisily through the
large tunnel that led to its large cage. I got some
peanuts and passed them through the cage to a gigantic
gray tray towering above my head. Feeding that animal
made me hungry. I went to get a large scoop of ice
cream. It filled my stomach. Afterwards I had to run
quickly to catch our bus. When I got home I thanked my
mom for a wonderful day at the zoo.
```

Problem 4 - Standard Deviation

Create a file called std_dev.py in your HW1 directory.

An occurring theme throughout your HWs will be about attendance. Your course coordinator is developing a project for a card swipe system for large courses. In this application, it will measure 3 weeks of attendance data for an example student. The class meets two days a week, so there will be only 6 samples. The samples are when the student arrives for class. If the sample is positive, the student entered class before the official start. This application will calculate the standard deviation of these samples.

Calculating the standard deviation uses the equation below:

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

(for help on calculating the standard deviation)

Sample output:

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

(running example on next page)

```
linux4[116]% python3 std_dev.py
Please enter sample 1 (in minutes): 82
Please enter sample 2 (in minutes): 93
Please enter sample 3 (in minutes): 98
Please enter sample 4 (in minutes): 89
Please enter sample 5 (in minutes): 88
Please enter sample 6 (in minutes): 99
The standard deviation of these times was:
6.473020933072903
```

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
mad_py_lib.py attendance.py std_dev.py rate_of_return.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 attendance.py
Submitting mad_py_lib.py...OK
Submitting attendance.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 attendance.py
It seems you have already submitted a file named
attendance.py.
Do you wish to overwrite? (y/n):
y

linux1[6]%
```