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

Due Date: Monday, September 16, 2024 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 - Favorite Things	favorite_things.py
Problem 2 - Convert Temperature	kelvin.py
Problem 3 - MadPyLib	mad_py_lib.py
Problem 4 - Escape Velocity	escape_velocity.py
Problem 5 - Pupper Walks	pupper_walks.py

For example you would do:

```
linux1[4]% submit cmsc201 HW1 mad_py_lib.py
favorite_things.py escape_velocity.py kelvin.py
pupper_walks.py
Submitting mad_py_lib.py...OK
Submitting favorite_things.py...OK
Submitting escape_velocity.py...OK
Submitting kelvin.py...OK
Submitting pupper_walks.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.

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 - Favorite Things

Make sure you name your file favorite_things.py

Your program should do these things:

1. Output your favorite food.
2. Output your favorite movie.
3. Output your favorite music.

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

```
linux4[120]% python3 favorite_things.py
My favorite food is sushi.
My favorite movie is The Fifth Element.
My favorite music is mongolian throat singing.
```

Problem 2 - Fahrenheit to Celsius to Kelvin

Call your file kelvin.py.

For this problem, create a program that takes in a number of degrees in Fahrenheit and converts it to Celsius and to Kelvin.

$$^\circ C = (^\circ F - 32) \cdot 5/9$$

Be careful with order of operations. For the Celsius to Kelvin conversion:

$$K =^\circ C + 273.15$$

Remember that there's no such thing as a degree Kelvin.

Input a temperature in Fahrenheit and convert it to both Celsius and Kelvin and output the results.

If you like, you can use the round(x, 2) function to round to 2 decimal digits.

Sample Runs:

```
linux4[116]% python3 kelvin.py
Enter a temperature in Fahrenheit: 212
The temperature in celsius is 100.0.
The temperature in Kelvin is 373.15.

linux4[117]% python3 kelvin.py
Enter a temperature in Fahrenheit: -40
The temperature in celsius is -40.0.
The temperature in Kelvin is 233.15.

linux4[118]% python3 kelvin.py
Enter a temperature in Fahrenheit: 58.6
The temperature in celsius is 14.78.
The temperature in Kelvin is 287.93.

linux4[119]% python3 kelvin.py
Enter a temperature in Fahrenheit: 1200
The temperature in celsius is 648.89.
The temperature in Kelvin is 922.04.
```

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, do the following:

1. Ask the user their name.
2. Ask the user for a subject/noun.
3. Ask for an adjective.
4. Ask for a verb.
5. Ask for another noun.
6. Fill in the blanks with the words that you got from the user into the following mad-lib: (The words go in the same order as they are input.)

```
Hello _____, we are going to have an amazing semester
learning _______, it's going to be ________ so don't
worry if you need to ________ from a _______.
```

```
linux4[122]% python3 mad_py_lib.py
Tell me your name: Eric
Tell me a subject/thing (noun): Python
Tell me an adjective: fun
Tell me a verb: get help
Tell me a noun: TA
Hello Eric, we are going to have an amazing semester
learning Python, it's going to be fun so don't worry if
you need to get help from a TA.

linux4[123]% python3 mad_py_lib.py
Tell me your name: Olive
Tell me a subject/thing (noun): sushi
Tell me an adjective: aggressive
Tell me a verb: climb
Tell me a noun: tree
Hello Olive, we are going to have an amazing semester
learning sushi, it's going to be aggressive so don't
worry if you need to climb from a tree.
```

Problem 4 - Escape Velocity

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

For this problem, you will create a program which calculates the velocity required to escape the gravitational pull of an object, starting at a specific radius. This is called the escape velocity.

$$v_{esc} = \sqrt{\frac{2GM}{r}}$$

G = 6.67 x 10^{-11} (keep in mind the negative exponent)

M = the mass of the body for which you will need to ask the user.

r = the distance from the center of the body where you launch your object.

Keep in mind that M is in kg and r is in meters. With the constant G canceling out some of the units we end up with meters/sec as our final unit.

When we talk about scientific notation in this problem, we mean that when you write in proper scientific notation with one non-zero digit before the decimal point, that is the coefficient, and the exponent is the power we raise 10 to in order to get the correct value.

$$SCI = COEFF \cdot 10^{EXP}$$

Do not use the math library, use ** for exponentiation.

Remember that in python ** calculates powers. When in doubt, use parentheses to ensure order of operations.

Do **not** use ^, that is bitwise xor.

Your program should do these things:

1. Ask what body you are launching from.
2. Ask the user for the (scientific notation) coefficient of mass of the body (float).
3. Ask for the exponent of the mass of the body (int).
4. Ask for the radius/distance coefficient (float).
5. Ask for the radius/distance exponent (int).
6. Calculate the escape velocity.
7. Display the escape velocity.

You are permitted to use round(num, num_digits) for this problem if you wish. I rounded to 3 digits for readability. **This is not required.**

As a last note, if you look at the sample output below you'll notice that it takes 42 km/sec of velocity to escape from our solar system when launching from Earth. Notice that this is on the order of 4 times the escape velocity required to escape from Earth's influence.

Sample output:

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

```
linux4[116]% python3 escape_velocity.py
What body are we launching from? Earth
Enter the mass of the planet in scientific notation
with the floating number first: 5.972
What power of 10 is this? 24
Enter the coefficient of the scientific notation of the
radius from the center of Earth: 6.371
What power of 10 is this? 6
The escape velocity required for Earth is 11182.374 m/s

linux4[116]% python3 escape_velocity.py
What body are we launching from? The Moon
Enter the mass of the planet in scientific notation
with the floating number first: 7.3476
What power of 10 is this? 22
Enter the coefficient of the scientific notation of the
radius from the center of The Moon: 1.7381
What power of 10 is this? 6
The escape velocity required for The Moon is 2374.725
m/s

linux4[116]% python3 escape_velocity.py
What body are we launching from? The Sun (from Earth's
Orbit)
Enter the mass of the planet in scientific notation
with the floating number first: 1.9891
What power of 10 is this? 30
Enter the coefficient of the scientific notation of the
radius from the center of The Sun (from Earth's Orbit):
1.496
What power of 10 is this? 11
The escape velocity required for The Sun (from Earth's
Orbit) is 42115.351 m/s
```

Problem 5 - Pupper Walks

In this problem you will calculate how long you walk Pupper each year.

Make sure you name your file pupper_walks.py

Your program should do these things:

1. Ask for Pupper's real name.
2. Ask how many times they walk pupper per week?
3. Ask how long the walk is in miles.
4. Ask how long it takes for you to walk a mile.
5. Calculate the distance you have covered in a year. Print that to the console.
6. How many hours have you spent walking the dog in a year? Print that to the console.
7. During this program create two intermediate variables where you do the calculations for distance and time. (Don't just print out the formula.)

There are 52 weeks in a year.

Sample output:

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

```
linux4[120]% python3 pupper_walks.py
What is pupper's real name? Olive
How many times per week do you walk pupper? 4
How long is the walk in miles? 2
How many minutes does it take for you to walk a mile?
18
Your dog's name is Olive, and you have walked 416.0
miles this year, in 124.8 hours.
```

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 favorite_things.py escape_velocity.py
distance.py pupper_walks.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 distance.py
Submitting mad_py_lib.py...OK
Submitting distance.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 distance.py
It seems you have already submitted a file named
distance.py.
Do you wish to overwrite? (y/n):
y

linux1[6]%
```