CMSC 201 Section 40

Homework 1
Print, Input, Variables, etc

Due Date: Monday, February 21st,, 2021 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 - PyCoin	py_coin.py
Problem 2 - Distance	distance.py
Problem 3 - MadPyLib	mad_py_lib.py
Problem 4 - The Expanse	the_expanse.py
Problem 5 - Interest	interest.py

For example you would do:

```
linux1[4]% submit cmsc201 HW1 mad_py_lib.py py_coin.py
the_expanse.py distance.py interest.py
Submitting py_coin.py...OK
Submitting favorite_things.py...OK
Submitting escape_velocity.py...OK
Submitting distance.py...OK
Submitting interest.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 - PyCoin

Make sure you name your file py_coin.py

Given the vast explosion of cryptocurrencies, I've decided to get into the game.

You should convert an item's value into py-coin.

1) First ask how much a py-coin is worth.
2) Ask the name of the item
3) Ask the value of the item.
4) Print out the value in Py-Coins.

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

```
linux4[120]% python3 py_coin.py
How many dollars is a py-coin now? 8.5
What item do you want to convert to py-coins? tesla
model 3
How many dollars is the item you want to buy? 45000
The value of a tesla model 3 in pycoins is
5294.117647058823

linux4[121]% python3 py_coin.py
How many dollars is a py-coin now? 10
What item do you want to convert to py-coins? ham
sandwich
How many dollars is the item you want to buy? 12
The value of a ham sandwich in pycoins is 1.2
```

Problem 2 - Distance

For this problem you'll determine the distance between two points. Make sure you name the file distance.py (all lower case).

The formula for Euclidean distance from (x_1, y_1) to (x_2, y_2) is:

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

Your program should do these things:

1. Ask the user for floating point numbers x_1, y_1, x_2, y_2 in that order.
2. Calculate the distance using the distance formula.
   a. Don't use math.sqrt()
3. **Do not round.**
4. Display the string:

   ```
   'The distance from (3.2, 5.0) to (5.7, 8.1) is '
   ```

   With the numbers replaced in the proper spots in the formula.

Sample Runs:

```
linux4[116]% python3 distance.py
Enter x_1: 3.2
Enter y_1: 5.0
Enter x_2: 5.7
Enter y_2: 8.1
The distance between (3.2, 5.0) and (5.7, 8.1) is
3.982461550347975

linux4[117]% python3 distance.py
Enter x_1: 0
Enter y_1: 0
Enter x_2: 1
Enter y_2: 1
The distance between (0.0, 0.0) and (1.0, 1.0) is
1.4142135623730951


linux4[118]% python3 distance.py
Enter x_1: 52
Enter y_1: -3
Enter x_2: 5
Enter y_2: -81
The distance between (52.0, -3.0) and (5.0, -81.0) is
91.0659101969557
```

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 - The Expanse

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

In the TV show 'The Expanse' ships would burn and constantly accelerate to simulate gravity, and then half way through the trip they would flip the ship and burn in the opposite direction in order to decelerate so that they could meet their target instead of flying past it.

Given a distance to travel d, and a factor of Earth gravity α, the amount of time it will take is:

$$t = 2 \cdot \sqrt{\frac{d}{\alpha g}}$$

For g, use 9.8 m/sec^2. What you should do is input the distance you want to travel, and then the factor of g that you want to accelerate.

Then calculate the time, and tell me the number of seconds, and then the number of hours (where you divide the result by 3600), and the number of days (dividing the result by 86,400).

Remember that powers are written as ** in Python. Do **NOT** import math to use the pow function. Remember to use parentheses in order to get order of operations correct.

In order to get the sample output, I used tabs '\t' in order to format the text a little.

Sample output:

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

```
linux4[116]% python3 the_expanse.py
What distance do you want to travel? 314885760000
What factor of g do you want to travel? .333
The amount of time a trip of 314885760000.0 meters will
take at an acceleration of 0.333*g is:
    621257.0807325743 seconds
    172.57141131460398 hours
    7.190475471441832 days

linux4[117]% python3 the_expanse.py
What distance do you want to travel? 885449678000
What factor of g do you want to travel? 1.5
The amount of time a trip of 885449678000.0 meters will
take at an acceleration of 1.5*g is:
    490855.05792561255 seconds
    136.34862720155905 hours
    5.68119280006496 days

linux4[118]% python3 the_expanse.py
What distance do you want to travel? 5289005767000
What factor of g do you want to travel? 1
The amount of time a trip of 5289005767000.0 meters
will take at an acceleration of 1.0*g is:
    1469278.0077581073 seconds
    408.1327799328076 hours
    17.00553249720317 days
```

Problem 5 - Interest

The cost of a loan with principal P, interest rate r, and number of payment periods n will be:

$$cost = \frac{Pr(1+r)^n}{(1+r)^n - 1}$$

Translate this pseudocode into a Python program.

```
Ask the user the amount of the loan.
Ask the user over how many years the loan will be paid back.
Ask the user the interest rate.

Calculate the total payment per month. Since we'll use monthly compounding you must use n = years * 12 and the interest rate must be divided by 12 before calculating.
```

For this program, not all of the names of the variables are not given to you. You should choose meaningful variable names. Do not just use P, r, n from the formula. Keep in mind that the interest rate should be given as a decimal, so 0.03 = 3%.

Sample output:

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

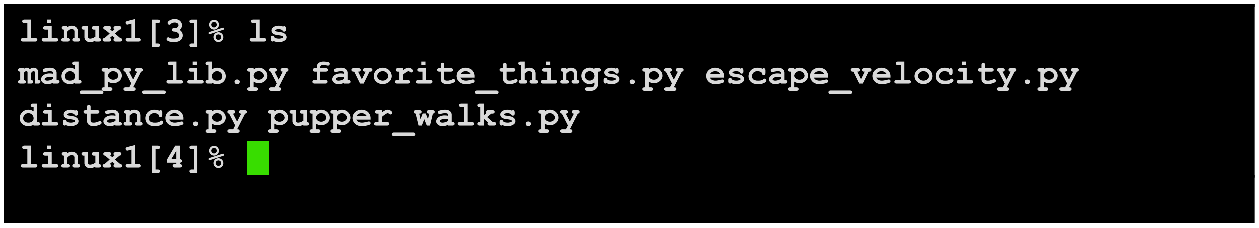
```
linux4[122]% python3 hw1_part3.py
What is the principal of the loan? 10000
What is the interest rate? 0.03
What is the length of the loan in years? 6
The monthly cost of the loan is 151.93675819439036

linux4[123]% python3 hw1_part3.py
What is the principal of the loan? 500000
What is the interest rate? 0.05
What is the length of the loan in years? 30
The monthly cost of the loan is 2684.108115060699

linux4[125]% python3 hw1_part3.py
What is the principal of the loan? 120000
What is the interest rate? 0.0667
What is the length of the loan in years? 10
The monthly cost of the loan is 1372.978247728574
```

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**.

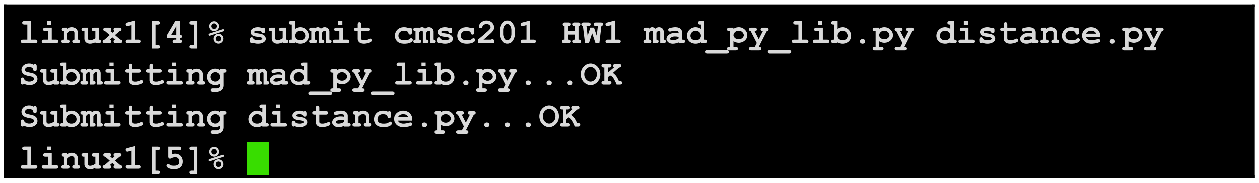


```
linux1[3]% ls
mad_py_lib.py favorite_things.py escape_velocity.py
distance.py pupper_walks.py
linux1[4]%
```

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

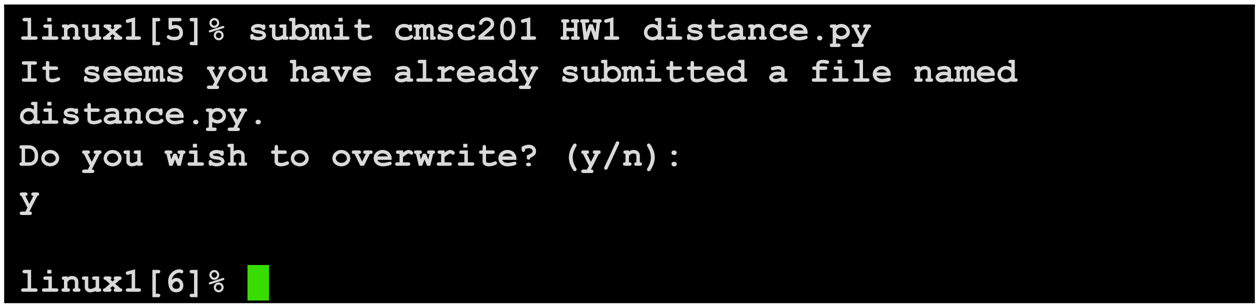
For more information on how to read the output from **`submitls`**, consult with Homework 0. Double-check that you submitted your homework correctly, since **empty files will result in a grade of zero.**

<u>Advice for Submitting Early and Often:</u>
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]%
```