CMSC 201 Fall 2017

Homework 2 – Decisions

Assignment: Homework 2 – Decisions
Due Date: Friday, September 22nd, 2017 by 8:59:59 PM
Value: 40 points

Collaboration: For Homework 2, **collaboration is not allowed** – you must work individually. You may still come to office hours for help, but you may not work with any other CMSC 201 students.

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
```

Instructions

For each of the questions below, you are given a problem that you must solve or a task you must complete. For this exercise, you will need to use concepts previously practiced in Homework 1, as well as concepts covered in class during the last week.

You should already be familiar with variables, expressions, **`input()`**, casting to an integer, and **`print()`**. You will also need to use one-way and two-way decision structures, as well as nested decision structures. You may also need to use multi-way decision structures for this assignment.

Think carefully about what the overall goal of the algorithm is before you begin coding.

At the end, your Homework 2 files must run without any errors.

NOTE: Your filenames for this homework must match the given ones exactly.
And remember, filenames are case sensitive!

Additional Instructions – Creating the hw2 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

Just as you did for Homework 1, you should create a directory to store your Homework 2 files. We recommend calling it **`hw2`**, and creating it inside the **`Homeworks`** directory inside the **`201`** directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1. (You don't need to make a separate folder for each file. You should store all of the Homework 2 files in the same **`hw2`** folder.)

Coding Standards

Prior to this assignment, you should re-read the Coding Standards, available on Blackboard under "Assignments" and linked on the course website at the top of the "Assignments" page.

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

- Naming Conventions
- Use of Whitespace
- Comments (specifically, File Header Comments)
- Line Length

Additional Specifications

For this assignment, **you must use** `main()` as seen in your **`lab2.py`** file, and as discussed in class.

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), 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.

Here is what that 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'
```

Questions

Each question is worth the indicated number of points. Following the coding standards is worth 4 points. If you do not have complete file headers and correctly named files, you will lose points.

hw2_part1.py **(Worth 6 points)**

This program simulates a "procrastination reprimand" by asking the user how long they waited to start on the assignments for Project 1. The user was given 6 days until the design is due, and 13 days until the project is due. Ask the user:

- How many days were left when they started on the design?
- How many days were left when they started on the project?

Depending on their answer, print out one of three responses:

- If they had 0 days left when they started the design **or** they had 0 days left when they started the project
  - Print out "Why would you wait so long!? :("
- If they had 6 days left when they started the design **and** they had 10, 11, 12, or 13 days left when they started the project
  - Print out "Wow, you started really early! Great job! :)"
- If they enter anything else (including invalid numbers like -2 or 300):
  - Print out "Good luck on the project! You can do it!"

(See the next page for sample output.)

Here is some sample output for **hw2_part1.py**, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

```
bash-4.1$ python hw2_part1.py
Project 1 has just come out, and you have 6 days to
complete the design, and 13 days to complete the project.
Days left when you start the design: 0
Days left when you start the project: 13
Why would you wait so long!? :(

bash-4.1$ python hw2_part1.py
Project 1 has just come out, and you have 6 days to
complete the design, and 13 days to complete the project.
Days left when you start the design: 6
Days left when you start the project: 10
Wow, you started really early!  Great job! :)

bash-4.1$ python hw2_part1.py
Project 1 has just come out, and you have 6 days to
complete the design, and 13 days to complete the project.
Days left when you start the design: 6
Days left when you start the project: 13
Wow, you started really early!  Great job! :)

bash-4.1$ python hw2_part1.py
Project 1 has just come out, and you have 6 days to
complete the design, and 13 days to complete the project.
Days left when you start the design: 5
Days left when you start the project: 12
Good luck on the project!  You can do it!

bash-4.1$ python hw2_part1.py
Project 1 has just come out, and you have 6 days to
complete the design, and 13 days to complete the project.
Days left when you start the design: 6
Days left when you start the project: 0
Why would you wait so long!? :(
```

hw2_part2.py **(Worth 5 points)**

Ask the user for two inputs: a number, and whether they would like to round the number "up" or "down". Then, depending on their choice, your program must calculate the rounded number, and print out the result to the screen. The final **result must be in the form of an integer** (no decimal point!).

Your program should be able to handle both decimals (floats) and whole numbers (integers) as input. It should also correctly handle negative numbers. Finally, it should print out the action it is taking, choosing from:

- Rounding down
- Rounding up
- (No action taken because it is already a whole number)
- Invalid command

HINT: We highly recommend that you test out integer division, casting, and any other ideas you have for this problem in the Python interpreter. Integer division and floating point numbers interact in ways you might not expect.

(You may not import the math library, and you may not use the built-in `round()` function in this assignment. Doing so will earn you 0 points.)

(See the next page for sample output.)

Here is some sample output for **hw2_part2.py**, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

The action taken is shown in **orange** (note the lack of action in the first example, where the input is the whole number 6.)
(The sample output has been updated.)

```
bash-4.1$ python hw2_part2.py
Input the number you are rounding: 6
Do you want to round up or down? down
Original value: 6.0
Rounded value: 6

bash-4.1$ python hw2_part2.py
Input the number you are rounding: -7.2
Do you want to round up or down? up
Original value: -7.2
Rounding up...
Rounded value: -7

bash-4.1$ python hw2_part2.py
Input the number you are rounding: 8.5
Do you want to round up or down? down
Original value: 8.5
Rounding down...
Rounded value: 8

bash-4.1$ python hw2_part2.py
Input the number you are rounding: 8.5
Do you want to round up or down? up
Original value: 8.5
Rounding up...
Rounded value: 9

bash-4.1$ python hw2_part2.py
Input the number you are rounding: 8.5
Do you want to round up or down? dogs
Original value: 8.5
Invalid command...
Rounded value: 8.5
```

hw2_part3.py **(Worth 3 points)**

This program simulates a simple deli ordering counter, asking the user what they would like on their sandwich, as a side, and to drink. After they are done ordering, the program informs them of what they ordered. (If they enter an invalid choice, they are informed of this at the end.)

The user's options are:

- Sandwich: ham, cheese, or veggie
- Side dish: cookies, chips, or pickle
- Drink: water, lemonade, or soda

Here is some sample output, with the user input in **blue**.
(Yours does not have to match this word for word, but it should be similar.)

```
bash-4.1$ python hw2_part3.py
Welcome to the CMSC 201 deli
Choose 'ham', 'cheese', or 'veggie' for your sandwich
Please choose a sandwich: CHEESE
Choose 'cookies', 'chips', or 'pickle' for your side
Please choose a side: cupcake
Choose 'water', 'lemonade', or 'soda' for your drink
Please choose a drink: water

A CHEESE is not a proper choice for a sandwich.
A cupcake is not a proper choice for a side.
You choose a water for your drink.
Enjoy your meal!

bash-4.1$ python hw2_part3.py
Welcome to the CMSC 201 deli
Choose 'ham', 'cheese', or 'veggie' for your sandwich
Please choose a sandwich: ham
Choose 'cookies', 'chips', or 'pickle' for your side
Please choose a side: pickle
Choose 'water', 'lemonade', or 'soda' for your drink
Please choose a drink: soda

You choose a ham sandwich.
You choose pickle for your side.
You choose a soda for your drink.
Enjoy your meal!
```

hw2_part4.py **(Worth 4 points)**

This program simulates a simple calculator, capable of performing the seven basic arithmetic operations using two integers. The user first selects which operation they would like to do, and then enters the two numbers to use. The options are add, subtract, multiply, divide, int divide, mod, and exponents.

The program then prints out the mathematical equation, along with the answer. For example, if the user chooses "subtract" and then enters the numbers 100 and 13, the program should print out `100 - 13 = 87`

If the user selects an invalid choice of operation, they receive an error.

Here is some sample output, with the user input in **blue**.
(Yours does not have to match this word for word, but it should be similar.)

(The intro text is removed from all but the first example below.)

```
bash-4.1$ python hw2_part4.py
You can do many operations with this program:
add, subtract, multiply, divide,
int divide, mod, and exponents
Which operation do you want to perform? dogs
Enter the first number of the operation: 5
Enter the second number of the operation: 8
Invalid Operation

bash-4.1$ python hw2_part4.py
Which operation do you want to perform? mod
Enter the first number of the operation: 77
Enter the second number of the operation: 6
77 % 6 = 5

bash-4.1$ python hw2_part4.py
Which operation do you want to perform? divide
Enter the first number of the operation: -563
Enter the second number of the operation: 72
-563 / 72 = -7.819444444444445

bash-4.1$ python hw2_part4.py
Which operation do you want to perform? int divide
Enter the first number of the operation: 986
Enter the second number of the operation: 25
986 // 25 = 39
```

hw2_part5.py **(Worth 9 points)**

This program plays a simple game, where it asks the player about their dog, and attempts to guess the dog's breed based on their answers. For simplicity's sake, there are only five possible dog breeds.

*(**WARNING**: This part of the homework is the most challenging, so budget plenty of time and brain power. And read the instructions carefully!)*

The program can ask the player about four characteristics. It should ask the ***minimum*** number of questions needed to guess the dog's breed.
(HINT: It should need to ask no less than two questions and no more than three questions to find the right breed.)

- Do the dog's ears stick up?
- Does the dog come in multiple colors?
- Can the dog bark?
- Does the dog have a curly tail?

For these inputs, the program can assume the following:

- The user will only ever enter either lowercase `yes` (for "yes") or lowercase `no` (for "no")

Based on the user's responses, the program must select the correct breed and print it to the screen. Here are the possibilities for the dog breeds:

- Dog has a curly tail and can bark: Eurasier
- Dog has a curly tail and *cannot* bark: Basenji
- Dog does *not* have a curly tail and its ears stick up: Pharaoh Hound
- Dog does *not* have a curly tail, its ears do *not* stick up, and it does come in multiple colors: Chesapeake Bay Retriever
- Dog does *not* have a curly tail, its ears do *not* stick up, and it does *not* come in multiple colors: Maremma Sheepdog

*(HINT: Do **not** start coding this part without having a plan! If you are stuck, try drawing a flowchart of the different options, or come to office hours for help.)*

(See the next page for sample output.)

Here is some sample output for **hw2_part5.py**, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

Make sure you spell all of the dog breeds correctly!

```
bash-4.1$ python hw2_part5.py
Please enter 'yes' or 'no' to these questions.
Does your dog have a curly tail? yes
Can your dog bark? yes
Your dog is a Eurasier!

bash-4.1$ python hw2_part5.py
Please enter 'yes' or 'no' to these questions.
Does your dog have a curly tail? yes
Can your dog bark? no
Your dog is a Basenji!

bash-4.1$ python hw2_part5.py
Please enter 'yes' or 'no' to these questions.
Does your dog have a curly tail? no
Do your dog's ear stick up? yes
Your dog is a Pharaoh Hound!

bash-4.1$ python hw2_part5.py
Please enter 'yes' or 'no' to these questions.
Does your dog have a curly tail? no
Do your dog's ear stick up? no
Does your dog come in multiple colors? yes
Your dog is a Chesapeake Bay Retriever!

bash-4.1$ python hw2_part5.py
Please enter 'yes' or 'no' to these questions.
Does your dog have a curly tail? no
Do your dog's ear stick up? no
Does your dog come in multiple colors? no
Your dog is a Maremma Sheepdog!
```

hw2_part6.py **(Worth 5 points)**

For this program, create a (very simplified) day of the week calculator. Ask the user to enter the day of the month, and respond with the correct day of the week.

The program will assume that the month starts on Friday and has 30 days (just like the month of September in 2017). The program

- Can assume that the number entered will be an integer
- Cannot assume that the number entered will be valid!

If the day of the month the user entered is not a valid day of the month (less than 1 or greater than 30), simply print a short error message to the user. Otherwise, print the day of the week that day falls on. For instance, the 2nd would be a Saturday, the 10th would be a Sunday, etc.

IMPORTANT: Do not write a case for each day of the month. If your program uses dozens of individual `if`, `elif`, or `else` statements, you will lose significant points.

(HINT: There is a mathematical operator in Python that will allow you to write this program without needing to have dozens of individual decision statements. Review Lecture 03 (Operators) to see it in action.)

(See the next page for sample output.)

Here is some sample output for **hw2_part6.py**, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

```
bash-4.1$ python hw2_part6.py
Please enter the day of the month: 17
Today is a Sunday!

bash-4.1$ python hw2_part6.py
Please enter the day of the month: 31
The date 31 is an invalid day.

bash-4.1$ python hw2_part6.py
Please enter the day of the month: 7
Today is a Thursday!

bash-4.1$ python hw2_part6.py
Please enter the day of the month: 0
The date 0 is an invalid day.

bash-4.1$ python hw2_part6.py
Please enter the day of the month: 1
Today is a Friday!
```

hw2_part7.py **(Worth 4 points)**

For this last program, you are going to ask the user about the state of two switches, and then use this information to determine the state of a generator.

For the input to these two questions, you can assume the following:

- The user will only ever enter either lowercase **`y`** (for "yes") or lowercase **`n`** (for "no")

If the user enters that both switches are in the same state (both on or both off), you should print "The generator is off."
If the user has answered that one switch is on and one switch is off, you should print "The generator is on."

IMPORTANT: You can only use a single **`if-else`** statement within this program! Think carefully about how you can accomplish this.
(The **`if`** statement can have a combination of multiple **`and`**s and **`or`**s, if you think that is needed to solve the problem.)

Here is some sample output, with the user input in **blue**.
(Yours does not have to match this word for word, but it should be similar.)

```
bash-4.1$ python hw2_part7.py
Please enter 'y' for yes and 'n' for no.
Is the first switch on? (y/n) y
Is the second switch on? (y/n) y
The generator is off.

bash-4.1$ python hw2_part7.py
Please enter 'y' for yes and 'n' for no.
Is the first switch on? (y/n) n
Is the second switch on? (y/n) n
The generator is off.

bash-4.1$ python hw2_part7.py
Please enter 'y' for yes and 'n' for no.
Is the first switch on? (y/n) y
Is the second switch on? (y/n) n
The generator is on.
```

Submitting

Once your **hw2_part1.py**, **hw2_part2.py**, **hw2_part3.py**, **hw2_part4.py**, **hw2_part5.py**, **hw2_part6.py**, and **hw2_part7.py** 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 2 Python files. To double-check you are in the directory with the correct files, you can type **ls**.

```
linux1[3]% ls
hw2_part1.py  hw2_part3.py  hw2_part5.py  hw2_part7.py
hw2_part2.py  hw2_part4.py  hw2_part6.py
linux1[4]%
```

To submit your Homework 2 Python files, we use the **submit** command, where the class is **cs201**, and the assignment is **HW2**. Type in (all on one line) **submit cs201 HW2 hw2_part1.py hw2_part2.py hw2_part3.py hw2_part4.py hw2_part5.py hw2_part6.py hw2_part7.py** and press enter.

```
linux1[4]% submit cs201 HW2 hw2_part1.py hw2_part2.py
hw2_part3.py hw2_part4.py hw2_part5.py hw2_part6.py
hw2_part7.py
Submitting hw2_part1.py...OK
Submitting hw2_part2.py...OK
Submitting hw2_part3.py...OK
Submitting hw2_part4.py...OK
Submitting hw2_part5.py...OK
Submitting hw2_part6.py...OK
Submitting hw2_part7.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**