UMBC
AN HONORS UNIVERSITY IN MARYLAND

CMSC 201 Spring 2018
Homework 2 – Decisions

Assignment: Homework 2 – Decisions
Due Date: Friday, February 23rd, 2018 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, 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 color mixer by asking the user to provide two colors, and predicting what the resulting color will be.

Depending on the colors they provide, print out one of three responses:

- If they entered the same color twice:
    - Print out That's so boring, you can't mix <COLOR> with itself! *(where <COLOR> is the color they entered)*

- If they entered two primary colors (blue, red, or yellow):
    - Print out Mixing <COLOR1> and <COLOR2> makes <MIXED> *(where <MIXED> is the secondary color created)*
        - Red and blue creates purple
        - Red and yellow creates orange
        - Blue and yellow creates green

- If they enter anything else:
    - Print out Mixing <COLOR1> and <COLOR2> makes brown

Your program only needs to work with lowercase letters; you do not need to worry about handling capitalization.

*(HINT: Do **not** start coding this part without having a plan! The order the colors are entered in should not make a difference, so think carefully about what your conditionals should be and how they should be nested.)*

(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
Please enter the first  color: blue
Please enter the second color: green
Mixing blue and green makes brown

bash-4.1$ python hw2_part1.py
Please enter the first  color: orange
Please enter the second color: orange
That's so boring, you can't mix orange with itself!

bash-4.1$ python hw2_part1.py
Please enter the first  color: red
Please enter the second color: blue
Mixing red and blue makes purple

bash-4.1$ python hw2_part1.py
Please enter the first  color: blue
Please enter the second color: yellow
Mixing blue and yellow makes green

bash-4.1$ python hw2_part1.py
Please enter the first  color: yellow
Please enter the second color: blue
Mixing yellow and blue makes green

bash-4.1$ python hw2_part1.py
Please enter the first  color: yellow
Please enter the second color: red
Mixing yellow and red makes orange

bash-4.1$ python hw2_part1.py
Please enter the first  color: red
Please enter the second color: yellow
Mixing red and yellow makes orange
```

hw2_part2.py **(Worth 8 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 points!).

Your program should be able to handle both decimals (floats) and whole numbers (integers) as input. It should also correctly handle negative numbers.

It should print out both the original number, as well as the rounded number that resulted from the operation.

Additionally, it should print out the action it is taking, choosing from:

- Rounding down
- Rounding up
- No rounding needed (already a whole number)
- Invalid command
  - (In this case, do not print out the “final” number result)

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 may interact in ways you don’t 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**, so that you can see it more clearly.

```
bash-4.1$ python hw2_part2.py
Input the number you are rounding: 3.7
Do you want to round up or down? dogs
Original number: 3.7
Invalid choice...

bash-4.1$ python hw2_part2.py
Input the number you are rounding: 6
Do you want to round up or down? down
Original number: 6.0
No rounding needed...
Rounded number: 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 number: -7.2
Rounding up...
Rounded number: -7

bash-4.1$ python hw2_part2.py
Input the number you are rounding: 18.5
Do you want to round up or down? down
Original number: 18.5
Rounding down...
Rounded number: 18

bash-4.1$ python hw2_part2.py
Input the number you are rounding: 18.5
Do you want to round up or down? up
Original number: 18.5
Rounding up...
Rounded number: 19
```

hw2_part3.py **(Worth 4 points)**

This program simulates a <u>very</u> simple calculator, which only performs division and subtraction.

The user first enters the two integers to use, then selects which operation they would like to do. The options are “subtract” and “divide”.

When done, the program 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`

However, regardless of the order the numbers are entered in, the program always:

- Subtracts the smaller number from the larger number
    - Creating a positive result
- Divides the smaller number by the larger number
    - Creating a result between 0 and 1

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

(See the next page for sample output.)

Here is some sample output for **hw2_part3.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_part3.py
Please enter the first  integer: 15
Please enter the second integer: 80
The options are subtract and divide.
What operation do you want to perform? subtract
80 - 15 = 65

bash-4.1$ python hw2_part3.py
Please enter the first  integer: 80
Please enter the second integer: 15
The options are subtract and divide.
What operation do you want to perform? subtract
80 - 15 = 65

bash-4.1$ python hw2_part3.py
Please enter the first  integer: 7
Please enter the second integer: 30
The options are subtract and divide.
What operation do you want to perform? divide
7 / 30 = 0.23333333333333334

bash-4.1$ python hw2_part3.py
Please enter the first  integer: 30
Please enter the second integer: 7
The options are subtract and divide.
What operation do you want to perform? divide
7 / 30 = 0.23333333333333334

bash-4.1$ python hw2_part3.py
Please enter the first  integer: 201
Please enter the second integer: 68
The options are subtract and divide.
What operation do you want to perform? dogs
Invalid operation.
```

hw2_part4.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 hang down?
- Does the dog have a long coat?
- Is the dog a herding breed?
- Does the dog only come in white?

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 is a herding breed and has a long coat: Belgian Shepherd
- Dog is a herding breed and *doesn't* have a long coat: Australian Kelpie
- Dog is *not* a herding breed and only comes in white: Dogo Argentino
- Dog is *not* a herding breed, *does not* only come in white, and its ears hang down: German Spaniel
- Dog is *not* a herding breed, *does not* only come in white, and its ears *do not* hang down: Canadian Eskimo Dog

*(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_part4.py**, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

Make sure to spell all of the dog breeds correctly, or you'll lose points!

```
bash-4.1$ python hw2_part4.py
Please enter 'yes' or 'no' to these questions.
Is your dog a herding breed? yes
Does your dog have a long coat? yes
Your dog is a Belgian Shepherd!

bash-4.1$ python hw2_part4.py
Please enter 'yes' or 'no' to these questions.
Is your dog a herding breed? yes
Does your dog have a long coat? no
Your dog is an Australian Kelpie!

bash-4.1$ python hw2_part4.py
Please enter 'yes' or 'no' to these questions.
Is your dog a herding breed? no
Does your dog only come in white? yes
Your dog is a Dogo Argentino!

bash-4.1$ python hw2_part4.py
Please enter 'yes' or 'no' to these questions.
Is your dog a herding breed? no
Does your dog only come in white? no
Do your dog's ears hang down? yes
Your dog is a German Spaniel!

bash-4.1$ python hw2_part4.py
Please enter 'yes' or 'no' to these questions.
Is your dog a herding breed? no
Does your dog only come in white? no
Do your dog's ears hang down? no
Your dog is a Canadian Eskimo Dog!
```

hw2_part5.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 Thursday and has 28 days (just like the month of February in 2018). 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 28), 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 Friday, the 10th would be a Saturday, 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_part5.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_part5.py
Please enter the day of the month: 17
Today is a Saturday!

bash-4.1$ python hw2_part5.py
Please enter the day of the month: 29
The date 29 is an invalid day.

bash-4.1$ python hw2_part5.py
Please enter the day of the month: 7
Today is a Wednesday!

bash-4.1$ python hw2_part5.py
Please enter the day of the month: 0
The date 0 is an invalid day.

bash-4.1$ python hw2_part5.py
Please enter the day of the month: 1
Today is a Thursday!

bash-4.1$ python hw2_part5.py
Please enter the day of the month: 28
Today is a Wednesday!
```

hw2_part6.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`** and **`or`** statements, 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_part6.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_part6.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_part6.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`**, and **`hw2_part6.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_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`** 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
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
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.**