CMSC 201 Homework 2 – Decisions

Due: Monday, September 21 at 11:59:59
Value: 50 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.

Make sure that you have a **complete file header comment at the top of <u>each</u> 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

Complete the following problems by writing Python programs. Each question is worth 8 points. You are permitted to use:

- Print and input built-in functions.
- String concatenation (using plus between strings).
- Casting quantities and variables to string, int, float.
- Modulus Division, as well as any other arithmetic and comparison operations.
- Logical operators (and, or, not)
- Conditional Logic (if, elif, else)

Sample output

Remember, the sample output is NOT COMPREHENSIVE. This means that you will need to test cases that aren't in the sample output to make sure that your code works for *all possible cases*.

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).

For Homework 2, you can store all five of the files you'll be creating in a single directory. First, navigate to the **`Homeworks`** directory inside the **`cmsc201`** directory (you can do this in a single "**`cd`**" command, as shown below). Next, create a directory to hold your Homework 2 files, and finally go into it.

```
linux3[1]% cd cmsc201/Homeworks
linux3[2]% mkdir hw2
linux3[3]% cd hw2
linux3[4]%
```

From here, you can use **`emacs`** to start creating and writing your different Homework 2 Python programs.

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 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

Coding standards can be found here.

Additional Specifications

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.

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'
```

Questions

Question 1 [10 points]

When baking zucchini bread you'll need:

1 cup of brown sugar
1 cup of white sugar
2 egg
1 cup of vegetable oil
3 cups flour
2 cups grated zucchini
(plus other ingredients)

Ask for ingredients in this order, and display the first ingredient we don't have enough of. For the zucchini, it's hard to know how many "cups" you have until you grate it, so instead, just ask if you have a big zucchini, if "yes", you're good, if "no" or any other user input, then you're not good.

Take in the number of cups/eggs of each ingredient and then display output like this (this does not display all of the options, but we will test them all):

```
linux3[104]% python3 hw2_part1.py
Welcome to the "Can we make zucchini bread today?"
calculator.
What number of cups of brown sugar do you have? 3
What number of cups of white sugar do you have? 2
How many eggs do you have? 5
How many cups of vegetable oil do you have? 1
How many cups of flour do you have? 4
Do you have a giant zucchini? yes
You are good to go, get baking!
```

```
linux3[105]% python3 hw2_part1.py
Welcome to the "Can we make zucchini bread today?"
calculator.
What number of cups of brown sugar do you have? 1
What number of cups of white sugar do you have? 1
How many eggs do you have? 2
How many cups of vegetable oil do you have? 0
How many cups of flour do you have? 1
Do you have a giant zucchini? yes
You do not have enough oil

linux3[106]% python3 hw2_part1.py
Welcome to the "Can we make zucchini bread today?"
calculator.
What number of cups of brown sugar do you have? 5
What number of cups of white sugar do you have? 5
How many eggs do you have? 5
How many cups of vegetable oil do you have? 5
How many cups of flour do you have? 5
Do you have a giant zucchini? no
You do not have a giant zucchini
```

Question 2 [10 points]

You are playing a role playing game (RPG) on your computer and find out that there's a puzzle to be solved. There is a door with two knobs and a switch.

There are two knobs with 12 positions each (1… 12) and a switch which can be toggled "up" or "down".

- In order to open the door, the first knob must be an even position, and the second knob must be in an odd position. The switch must be flipped to up. If this is all done, then print "The door opens, you get all the loot."
- If one of the knobs is flipped to its correct position (even or odd), or both knobs are correct but the switch is down, then "The door clanks but does not open, try again."
- If none of the knobs are correct, then you print "The handle doesn't budge."

Make sure you take in the knob positions first, and then the switch. Consider error conditions like the last sample output, where the user enters jibberish or invalid input.
(Keep in mind that all of the input will be of the correct type, that is to say, if you ask for an integer position, we may enter -17 but we will not enter "rabbit".)

```
linux3[6]% python3 hw2_part2.py
What is the position of the first knob? (Enter 1-12) 3
What is the position of the second knob? (Enter 1-12) 5
What is the position of the switch? (Enter up or down)
down
The door clanks but does not open, try again.

linux3[7]% python3 hw2_part2.py
What is the position of the first knob? (Enter 1-12) 4
What is the position of the second knob? (Enter 1-12) 7
What is the position of the switch? (Enter up or down) up
The door opens, you get all the loot.

linux3[8]% python3 hw2_part2.py
What is the position of the first knob? (Enter 1-12) -2
What is the position of the second knob? (Enter 1-12) 3
What is the position of the switch? (Enter up or down) up
Knob 1 needs to be set to 1 - 12

linux3[9]% python3 hw2_part2.py
What is the position of the first knob? (Enter 1-12) 5
What is the position of the second knob? (Enter 1-12) 2
What is the position of the switch? (Enter up or down)
down
The handle doesn't budge.
```

Question 3 [10 points]

We're going to make a simplified game of 20 questions, where we're going to guess original Star Trek characters. Mimic the following structure of questions and responses. Accept "yes" and "no" answers only.

- Is the character human (at least partially)?
    - If yes, ask whether they're bald.
        - If Yes, they're Captain Picard
    - Now ask if they are an empath.
        - If yes, they're Counselor Troi
    - Are they blind?
        - If yes, then they're Geordi.
    - Are they number 1?
        - Then they're Riker.
    - If none of these work, then print out "Shut up Wesley."
- Ask if they're Klingon?
    - If yes, then Worf.
- Ask if they're an Android?
    - If yes, then Data.
- If none of it is right, then just guess "Mot the Barber."

What about Case? All input will match the case used in the sample output, but if you want you are allowed to convert everything to upper or lowercase using the .lower() or .upper() method to avoid problems with case.

```
linux3[9]% python3 hw2_part3.py
Is the person human (at least half)? no
Is the person Klingon? yes
You are thinking about Worf.

linux3[10]% python3 hw2_part3.py
Is the person human (at least half)? yes
Is the person bald? no
Is the person empathic? yes
You are thinking about Troi

linux3[11]% python3 hw2_part3.py
Is the person human (at least half)? no
Is the person Klingon? no
Are they an android? no
I don't know, Mot the Barber maybe?


linux3[12]% python3 hw2_part3.py
Is the person human (at least half)? yes
Is the person bald? no
Is the person empathic? no
Is the person blind? no
Is the person number 1? yes
You are thinking about Riker

linux3[12]% python3 hw2_part3.py
Is the person human (at least half)? yes
Is the person bald? no
Is the person empathic? no
Is the person blind? no
Is the person number 1? no
Shut up Wesley
```

Question 4 [10 points]

Leap years work this way:

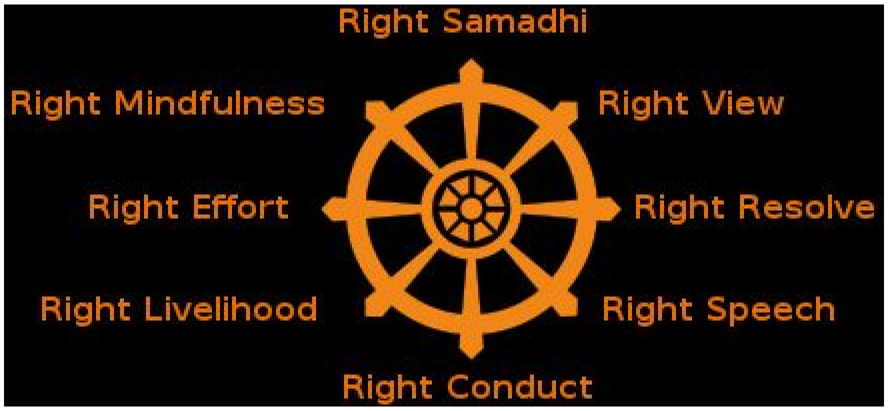
- If the year is divisible by 4, then it is a leap year, unless:
- If the year is divisible by 100, then it is not a leap year, even though it will be divisible by 4.
- If the year is divisible by 400, then it is again a leap year. So 2000 was a leap year but 2100 will not be for example.

Input a year, and determine whether the year is a leap year. Print output containing "leap year" if it is a leap year, and containing "not a leap year" if it is not a leap year.

```
linux3[12]% python3 hw2_part4.py
Which year do you want to know if it is a leap year? 2100
This is not a leap year.

linux3[13]% python3 hw2_part4.py
Which year do you want to know if it is a leap year? 2400
This is a leap year.

linux3[14]% python3 hw2_part4.py
Which year do you want to know if it is a leap year? 1704
This is a leap year.

linux3[15]% python3 hw2_part4.py
Which year do you want to know if it is a leap year? 1833
The year is not a leap year.

linux3[16]% python3 hw2_part4.py
Which year do you want to know if it is a leap year? -17
The year is less than 1 AD/CE, we aren't going back in
time.
```

Question 5 [10 points]

The eightfold path in Buddhism is drawn as the following object:



Note that each point on the wheel represents an angle of 45 degrees apart from the previous angle.

We'll start at zero, the x-axis as usual with Right Resolve, to Right View at 45 degrees, Right Samadhi at 90, Right Mindfulness at 135, etc, and then move counterclockwise as angles go until we reach Right Speech at 315, and finally back to Right Resolve at 360 degrees. It doesn't matter how many times you wind around the wheel, so 360, 720, etc will all lead back to Right Resolve.

Your job is to calculate the part of the eightfold path at a particular angle. Use integers, and if you get an integer which is not on the path, you should print "You have not reached enlightenment yet, try an angle divisible by 45."

Do not use lists, but you are encouraged to use the copy-paste feature to complete most of the cases.

Sample output on the next page:

```
linux3[14]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 121
You have not reached enlightenment yet, try an angle divisible by
45.

linux3[15]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 45
You have selected Right View

linux3[16]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 225
You have selected Right Livelihood

linux3[17]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 0
You have selected Right Resolve

linux3[17]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 720
You have selected Right Resolve

linux3[17]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 405
You have selected Right View
```

Submitting

NOTE: How to submit is covered in detail in Homework 0. If you have not read those assignments yet, you should do so before completing this part of the homework.

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

```
linux1[101]% ls
hw2_part1.py  hw2_part3.py  hw2_part5.py
hw2_part2.py  hw2_part4.py
linux1[102]% 
```

To submit your homework Python files, we use the **`submit`** command, where the class is **`cmsc201`**, and the assignment is **`HW2`**. Type in (all on one line) **`submit cmsc201 HW2 hw2_part1.py hw2_part2.py hw2_part3.py hw2_part4.py hw2_part5.py`** and press enter.

```
linux1[4]% submit cmsc201 HW2 hw2_part1.py hw2_part2.py
hw2_part3.py hw2_part4.py hw2_part5.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
linux1[5]% 
```

(continued on next page)

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 see what files were successfully submitted by running:
`submitls cmsc201 HW2`

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 (a good idea to do as you complete each part, *hint! hint!*) simply by modifying the command. For example, if you wanted to turn in just parts 2 and 3, type in
`submit cmsc201 HW2 hw2_part2.py hw2_part3.py` and press enter.

```
linux1[4]% submit cmsc201 HW2 hw2_part2.py hw2_part3.py
Submitting hw2_part2.py...OK
Submitting hw2_part3.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 HW2 hw2_part3.py
It seems you have already submitted a file named
hw2_part3.py.
Do you wish to overwrite? (y/n):
y

linux1[6]% 
```