CMSC 201, Section 40
Spring 2020

Homework 5 – While Loops (con't)

Assignment: Homework 5 – While Loops (con't)
Due Date: Monday, March 23rd before Midnight
Value: 40 points

Collaboration: For Homework 5, collaboration is **NOT** allowed.

Make sure that you have a **complete file header comment at the top of <u>each</u> file**, and that all the information is correctly filled in.

```
"""
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
"""
```

<u>Instructions</u>

For each of the questions below, you are given a problem that you must solve or a task you must complete. This homework will deal with the concepts of **`while`** loops, which were covered over last week two weeks.

At the end, your Homework 5 files must run without any errors.

Additional Instructions – Creating the hw5 Directory

Just as you did for your previous homeworks, you should create a directory to store your Homework 5 files. We recommend calling it **`hw5`** and creating it inside the **`Homeworks`** directory inside your **`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 Homework 5 files in the same **`hw5`** folder.)

Coding Standards

Prior to this assignment, you should re-read the Coding Standards, available 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
- Constants
- In-line Comments
- Line Length

Additional Specifications

As usual, **you must use** **`if __name__ == '__main__':`** as discussed in class.

Many of the parts of this assignment center on validating input from the user. For example, the user may enter a negative value, but your program may require a positive value. **Make sure to follow each part's instructions about input validation.**

If the user enters a different type of data than you asked for, your program may crash. This is acceptable.

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 header comments and correctly named files, you will lose points.

hw5_part1.py – Guessing a Number (Worth 8 points)

Write a program that asks the user to guess a number that has the attributes described below. If the number fulfills all the criteria, then a message is printed saying that the user guessed correctly. Otherwise, print a message saying what criteria were not fulfilled by the input. You are required to use a Boolean flag-controlled while loop for this problem.

The criteria for the number are:

- It is positive (> 0).
- It is less than 100.
- It is evenly divisible by 5, 3, and 9.
- The remainder of the number divided by 6 is 3.
- The product of 2 and the number plus 10 is 100.

Make good use of constants!

Some sample output is shown on the next page, with the user input in blue. (Yours does not have to match this word for word, but it should be similar.)

```
linux4[144]% python3 hw5_part1.py
Please enter a guess: 0
0 is not positive.
The remainder of 0 divided by 6 is not 3.
The product of 2 and 0 plus 10 is not 100.
Please enter a guess: 100
100 is not less than 100.
100 is not evenly divisible by 9.
100 is not evenly divisible by 3.
The remainder of 100 divided by 6 is not 3.
The product of 2 and 100 plus 10 is not 100.
Please enter a guess: 46
46 is not evenly divisible by 5.
46 is not evenly divisible by 9.
46 is not evenly divisible by 3.
The remainder of 46 divided by 6 is not 3.
The product of 2 and 46 plus 10 is not 100.
Please enter a guess: 45
45 is the secret number!
```

hw5_part2.py – Seasons **(Worth 7 points)**

This program allows people to enter their favorite month and then determines the total votes cast for each of the four seasons in which the favorite months fall.

The program should display a menu of choices for the 12 months. Then, the user enters votes for their favorite month. When the votes are done being cast, the program displays the total votes by season.

You must use a list to hold the 12 menu items (e.g. "1 – January")

The user can continue entering favorite months indefinitely, stopping only when they enter the sentinel value **"-1"**. Each time a vote is cast, it should print out the name of the month that was voted for. After they stop entering in votes, the program should print out how many votes were made in each season.

After their list is complete, the votes cast by season should be printed back out to them. Here is the breakdown for months by season:

- **Fall**: September, October, November
- **Winter**: December, January, February
- **Spring**: March, April, May
- **Summer**: June, July, August

You can assume that the user will enter a valid month number (1 – 12, inclusive).

Some sample output is shown on the next page, with the user input in blue. (Yours does not have to match this word for word, but it should be similar.)

```
linux4[263]% python3 hw5_part2.py
 1 - January
 2 - February
 3 - March
 4 - April
 5 - May
 6 - June
 7 - July
 8 - August
 9 - September
10 - October
11 - November
12 - December
What is your favorite month? (-1 to quit): -1
0 votes for winter
0 votes for spring
0 votes for summer
0 votes for fall

linux4[264]% python3 hw5_part2.py
[[ MENU OMITTED TO FIT PAGE ]]
What is your favorite month? (-1 to quit): 1
What is your favorite month? (-1 to quit): 1
What is your favorite month? (-1 to quit): 1
What is your favorite month? (-1 to quit): 1
What is your favorite month? (-1 to quit): 5
What is your favorite month? (-1 to quit): 4
What is your favorite month? (-1 to quit): 6
What is your favorite month? (-1 to quit): 5
What is your favorite month? (-1 to quit): 9
What is your favorite month? (-1 to quit): -1
4 votes for winter
3 votes for spring
1 votes for summer
1 votes for fall
```

hw5_part3.py – Special Day **(Worth 8 points)**

Write a program that determines if a particular day in February is a "special day." A special day is defined as:

- A day of the month that ends in a 9. For a leap year, that includes the 29th; for a non-leap year, it does not.
- The day of the week is a Tuesday.

You are required to use a Boolean flag-controlled while loop for this problem.

You may assume that the user will enter a correct day of the week (e.g. Sunday, Monday, . . ., Saturday). You may also assume that they will enter an integer for the day of the month.

Some sample output is shown on the next page, with the user input in blue. (Yours does not have to match this word for word, but it should be similar.)

```
linux4[170]% python3 hw5_part3.py
Is it a leap year (y/n)? y
What day of the month is it? 28
What day of the week is it? Tuesday
February 28 is not a special day
Is it a leap year (y/n)? n
What day of the month is it? 19
What day of the week is it? Monday
February 19 is not a special day
Is it a leap year (y/n)? y
What day of the month is it? 29
What day of the week is it? Tuesday
February 29 is a special day!
linux4[171]% python3 hw5_part3.py
Is it a leap year (y/n)? n
What day of the month is it? 19
What day of the week is it? Tuesday
February 19 is a special day!
```

hw5_part4.py – Integer Division (Worth 5 points)

Write a program that can calculate the answer to an integer division problem without using the division, integer division, mod, or multiplication operators.

The program should ask the user for two integers and then compute the answer to first_num // second_num. The program should then print the full equation and the answer.

For the inputs, you can assume:

- The first number may be any positive integer or zero.
- The second number may be any positive integer greater than zero.

Hint: Think of using subtraction in your while loop.

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.)

```
linux4[230]% python3 hw5_part4.py
Please enter the first number: 0
Please enter the second number: 15
0 // 15 = 0

linux4[231]% python3 hw5_part4.py
Please enter the first number: 15
Please enter the second number: 7
15 // 7 = 2

linux4[232]% python3 hw5_part4.py
Please enter the first number: 7359
Please enter the second number: 9
7359 // 9 = 817

linux4[235]% python3 hw5_part4.py
Please enter the first number: 201
Please enter the second number: 42
201 // 42 = 4
```

hw5_part5.py – String Lengths (Worth 8 points)

Write a program that determines if all the strings that a user enters contain less than or equal to 3 characters, if exactly one of the strings contains greater than 3 characters, or if more than one string contains 3 characters.

The user will be able to enter in as many strings as they want until they input "QUIT".

Your program should have two constants.

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.)

```
linux4[224]% python3 hw5_part5.py
Please enter a string (or QUIT to exit): QUIT
All strings entered have less than or equal to 3
characters in them.

linux4[225]% python3 hw5_part5.py
Please enter a string (or QUIT to exit): abc
Please enter a string (or QUIT to exit): QUIT
All strings entered have less than or equal to 3
characters in them.

linux4[226]% python3 hw5_part5.py
Please enter a string (or QUIT to exit): a
Please enter a string (or QUIT to exit): abcd
Please enter a string (or QUIT to exit): d
Please enter a string (or QUIT to exit): QUIT
There is one string that has greater than 3 characters in
it.

linux4[227]% python3 hw5_part5.py
Please enter a string (or QUIT to exit): abcd
Please enter a string (or QUIT to exit): abcde
Please enter a string (or QUIT to exit): e
Please enter a string (or QUIT to exit): QUIT
There are 2 strings that have greater than 3 characters in
them.
```

Submitting

Once your **hw5_part1.py**, **hw5_part2.py**, **hw5_part3.py**, **hw4_part5.py**, and **hw5_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 may resubmit any or all your files as many times as you like up until the due date/time. Be aware that when you resubmit a file, you are overwriting the last version that you submitted!

You must be logged into your account on GL, and you must be in the same directory as your Homework 5 Python files. To double-check that you are in the directory with the correct files, you can type **ls**.

```
linux1[3]% ls
hw5_part1.py  hw5_part3.py  hw5_part5.py
hw5_part2.py  hw5_part4.py
linux1[4]%
```

To submit your Homework 5 Python files, we use the **submit** command, where the class is **cs201s**, and the assignment is **HW5**. Type in (all on one line, even if it wraps around the screen) **submit cs201s HW5 hw5_part1.py hw5_part2.py hw5_part3.py hw5_part4.py hw5_part5.py** and press Enter.

```
linux1[4]% submit cs201s HW5 hw5_part1.py hw5_part2.py
hw5_part3.py hw5_part4.py hw5_part5.py
Submitting hw5_part1.py...OK
Submitting hw5_part2.py...OK
Submitting hw5_part3.py...OK
Submitting hw5_part4.py...OK
Submitting hw5_part5.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.**