HW3 - If statements, Loops, Logic

50 points

Homework Description

Write a Python program, which solves the problems from the book: 3.4, 3.9, 3.11, 3.12, 3.14 (Chapter 3, page 112).
Each problem is worth 10 points.

For each problem, write a comment just before the code specifying the problem number. Also add a comment before each problem explaining your coding solution. Remember, before every new piece of logic in your code, have a comment before clearly explaining what the code does.

Grading rubric

For full credit:
Complete the problem successfully, with good coding style and comments.

Ways to lose points:
-3 on each problem, if not completed, but attempted.
-1 on each problem, if bad style and/or comments.

Submission

On Cloud9: Create a folder “hw3”. Create one file “hw3.py” in folder “hw3”. Put all your code, for all problems in hw3.py, with good comments indicating which code solves which problem.
On Canvas: Submit your AWS Console Login url, and your Cloud9 IDE url. Spencer will click on your first url, then login. Then he’ll click on your second url and it should take him straight to your IDE where he can view and run your code.

Problems

The problems come from the book, I have also copy/pasted them below as images (I couldn’t copy/paste the text, the publisher wouldn't let me).

3.4 *(Fill in the Missing Code)* In the code below

```
for ***:
    for ***:
        print('@')
    print()
```

replace the `***` so that when you execute the code, it displays two rows, each containing seven `@` symbols, as in:

```
@@@@@@@
@@@@@@@
```

3.9 *(Separating the Digits in an Integer)* In Exercise 2.11, you wrote a script that separated a five-digit integer into its individual digits and displayed them. Reimplement your script to use a loop that in each iteration "picks off" one digit (left to right) using the `//` and `%` operators, then displays that digit.

Update: For 3.9, have the user enter an integer between 7 and 10 digits (not five-digit). Remember, you'll need to convert the inputted string to an integer. Here is an example output:

Enter a number 7 to 10 digits: 12345678
1
2
3
4
5
6
7
8

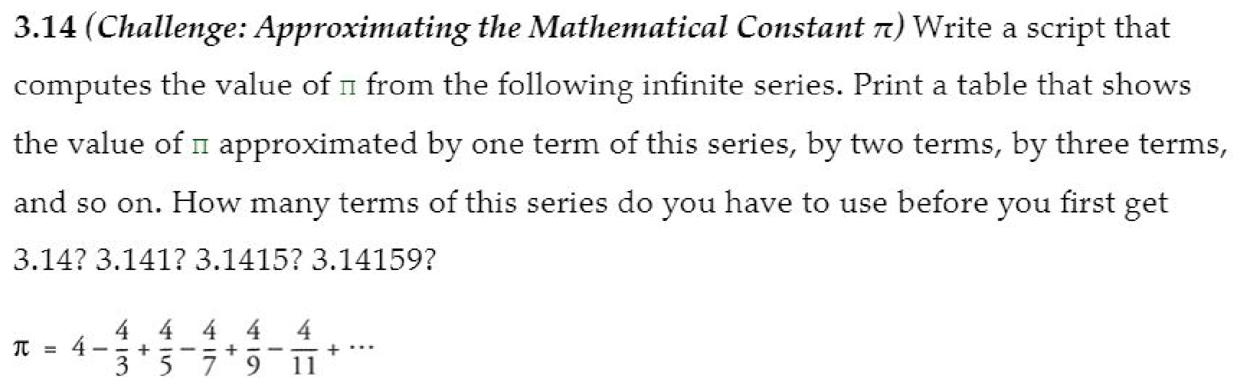
3.11 ***(Miles Per Gallon)*** Drivers are concerned with the mileage obtained by their automobiles. One driver has kept track of several tankfuls of gasoline by recording miles driven and gallons used for each tankful. Develop a sentinel-controlled-repetition script that prompts the user to input the miles driven and gallons used for each tankful. The script should calculate and display the miles per gallon obtained for each tankful. After processing all input information, the script should calculate and display the combined miles per gallon obtained for all tankfuls (that is, total miles driven divided by total gallons used).

```
Enter the gallons used (-1 to end): 12.8
Enter the miles driven: 287
The miles/gallon for this tank was 22.421875
Enter the gallons used (-1 to end): 10.3
Enter the miles driven: 200
The miles/gallon for this tank was 19.417475
Enter the gallons used (-1 to end): 5
Enter the miles driven: 120
The miles/gallon for this tank was 24.000000
Enter the gallons used (-1 to end): -1
The overall average miles/gallon was 21.601423
```

3.12 ***(Palindromes)*** A palindrome is a number, word or text phrase that reads the same backwards or forwards. For example, each of the following five-digit integers is a palindrome: 12321, 55555, 45554 and 11611. Write a script that reads in a five-digit integer and determines whether it's a palindrome. [*Hint*: Use the `//` and `%` operators to separate the number into its digits.]

Update on question 3.12:
You can input the number as a string, and then read it backwards to determine if it's a palindrome. This is a classic interview problem, so I'm ok if you do it using a string. (although to me a 5 digit integer is easier).

However, do not test it like this, using [::-1] this is not allowed (because we haven't covered it yet):
If 'bob' == 'bob'[::-1]: print(“palindrome”)



3.14 *(Challenge: Approximating the Mathematical Constant π)* Write a script that computes the value of π from the following infinite series. Print a table that shows the value of π approximated by one term of this series, by two terms, by three terms, and so on. How many terms of this series do you have to use before you first get 3.14? 3.141? 3.1415? 3.14159?

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \cdots$$

After you have written your program to approximate pi, answer the last question (How many terms of this series do you have to use before you first get 3.14 and 3.141?...) in the comments just above your code.

Update on question 3.14: You do not need to approximate to 3.1415 and 3.14159. You can just approximate to 3.14 and 3.141, but please tell me at what iteration of the loop you see 3.14 twice in a row, not just once. Same for 3.141, tell me at which iteration in the loop you 3.141 twice in a row, not just once.
Hint: your for loop shouldn’t have to go more than 3000 iterations.
Another hint: The reason I updated this requirement is because people’s AWS has been crashing when you run a for loop above 10000. Setting it to 3000, should not crash the server, and be enough to allow you to approximate to 3.14 and 3.141.