Chapter 3 assignment

3.1 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 Python program that prompts the user to input the miles driven and gallons used for each tank- ful. The program should calculate and display the miles per gallon obtained for each tankful. After processing all input information, the program should calculate and print the combined miles per gal- lon obtained for all tankful (= 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

3.2 A palindrome is a number or a 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 program that reads in a five-digit integer and determines whether it is a palindrome. (*Hint*: Use the division and modulus operators to separate the number into its individual digits.)

3.3 Input an integer containing 0s and 1s (i.e., a "binary" integer) and print its decimal equiva- lent. Appendix C, Number Systems, discusses the binary number system. (*Hint*: Use the modulus and division operators to pick off the "binary" number's digits one at a time from right to left. Just as in the decimal number system, where the rightmost digit has the positional value 1 and the next digit leftward has the positional value 10, then 100, then 1000, etc., in the binary number system, the right- most digit has a positional value 1, the next digit leftward has the positional value 2, then 4, then 8, etc. Thus, the decimal number 234 can be interpreted as 2 * 100 + 3 * 10 + 4 * 1. The decimal equiv- alent of binary 1101 is 1 * 8 + 1 * 4 + 0 * 2 + 1 * 1.)

3.4 The factorial of a nonnegative integer n is written $n!$ (pronounced "n factorial") and is defined as follows:

$n!=n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot1$ (forvaluesof$n$greaterthanorequalto1) and
$n!=1$ (for$n=0$).

For example, 5! = 5 · 4 · 3 · 2 · 1, which is 120. Factorials increase in size very rapidly. What is the largest factorial that your program can calculate before leading to an overflow error?

a) Write a program that reads a nonnegative integer and computes and prints its factorial.

b) Write a program that estimates the value of the mathematical constant *e* by using the formula

e = 1 + 1/1!+ 1/2!+ 1/3!+....

c) Write a program that computes the value of *ex* by using the formula [*Note*: Your program can stop after summing 10 terms.]

e^x =1+ x/1! +x^2/2! + x^3/3! +...

3.5 Write a program that prints the following patterns separately, one below the other each pat- tern separated from the next by one blank line. Use **for** loops to generate the patterns. All asterisks (*****) should be printed by a single statement of the form

print '*',

(which causes the asterisks to print side by side separated by a space). (*Hint*: The last two patterns require that each line begin with an appropriate number of blanks.) Extra credit: Combine your code from the four separate problems into a single program that prints all four patterns side by side by making clever use of nested **for** loops. For all parts of this program—minimize the numbers of asterisks and spaces and the number of statements that print these characters.

a) Half pyramid

b)Left Half Pyramid

c) Inverted half pyramid

d)Full inverted Pyramid

3.6 *(Pythagorean Triples)* A right triangle can have sides that are all integers. The set of three integer values for the sides of a right triangle is called a Pythagorean triple. These three sides must satisfy the relationship that the sum of the squares of two of the sides is equal to the square of the hypotenuse. Find all Pythagorean triples for **side1**, **side2** and **hypotenuse** all no larger than 20. Use a triple-nested **for**-loop that tries all possibilities. This is an example of "brute force" comput- ing. You will learn in more advanced computer science courses that there are many interesting prob- lems for which there is no known algorithmic approach other than sheer brute force.

3.7 Write a Python program which takes two digits m (row) and n (column) as input and generates a two-dimensional array. The element value in the i-th row and j-th column of the array should be i*j.

Note :

i = 0,1.., m-1

j = 0,1, n-1.

Test Data : Rows = 3, Columns = 4

Expected Result : [[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6]]

3.8 Suppose you give a dinner party for six guests, but your table seats only four. In how many ways can four of the six guests arrange themselves at the table? Any of the six guests can sit in the first chair. Any of the remaining five can sit in the second chair. Any of the remaining four can sit in the third chair, and any of the remaining three can sit in the fourth chair. (The last two will have to stand.) So the number of possible arrangements of six guests in four chairs is 6*5*4*3, which is 360. Write a program that calcu- lates the number of possible arrangements for any number of guests and any number of chairs. (Assume there will never be fewer guests than chairs.) Don't let this get too com- plicated. A simple for loop should do it.

3.9 Write a program that calculates how much money you'll end up with the invested amount of money at a fixed interest rate, compounded yearly. Have the user furnish the initial amount, the number of years, and the yearly interest rate in percent. Some interaction with the program might look like this:

Enter initial amount: 3000

Enter number of years: 10

Enter interest rate (percent per year): 5.5

At the end of 10 years, you will have 5124.43 dollars.

At the end of the first year you have 3000 + (3000 * 0.055), which is 3165. At the end of the second year you have 3165 + (3165 * 0.055), which is 3339.08. Do this as many times as there are years. A for loop makes the calculation easy.