CS1026: Assignment 1

Weight: 4%

Learning Outcome:
By completing this assignment, you will gain skills relating to

- basic programming constructs,
- expressions and decisions,
- getting input from user,
- validating input,
- algorithm development and testing.

Task:
In this assignment, you will write a **complete** program in Python that computes rental costs for a car rental company. Your program is expected to prompt the user for input and validate it before computing the results. Your program should make use of expressions, decisions and basic input/output in Python.

Functional Specifications:

1. The program will compute and display information for a company which rents vehicles to its customers. For a specified customer, the program will compute and display the amount of money charged for that customer’s vehicle rental. The program will prompt the user to enter the following items for a given customer (in the specified order):
   a. The customer’s name (a string)
   b. The customer's classification code (a character).
   c. The number of days the vehicle was rented (an integer).
   d. The vehicle's odometer reading at the start of the rental period (an integer).
   e. The vehicle's odometer reading at the end of the rental period (an integer).

   It will then process that customer information and display the results.

2. The program will compute the amount of money that the customer will be billed, based on the customer's classification code, number of days in the rental period, and number of kilometers driven.

3. The program will recognize both upper case and lower case letters for the classification codes; the codes and related information are as follows:
   a. Code 'B' (budget)
      i. base charge: $20.00 for each day,
      ii. kilometers driven charge: $0.30 for each kilometer driven.

   b. Code 'D' (daily)
      i. base charge: $50.00 for each day,

    ii. kilometers driven charge: no charge if the average number of kilometers driven per day is 100 kilometers or less; otherwise, $0.30 for each kilometer driven above the 100 kilometer per day limit.

  c. Code 'W' (weekly)
    i. base charge: $200.00 for each week or fraction of a week (i.e., each fraction of a week is treated as a week, for example 8 days is treated as 2 weeks),
    ii. kilometers driven charge:
      - There is no additional charge if the average number of kilometers driven per week is 1000 kilometers or less;
      - If the average number of kilometers driven per week exceeds 1000 kilometers but does not exceed 2000 kilometers, then there is an additional $50.00 charge per week;
      - If the average number of kilometers driven per week exceeds the 2000 kilometer per week limit, then there is an additional $100.00 charge per week plus $0.30 for each kilometer driven over the 2000 kilometer per week average for each week.

4. The program will compute the number of kilometers driven by the customer during the rental period.

5. For each customer, the program will display a summary with the following information:
  a. The customer’s name,
  b. The customer's classification code,
  c. The number of days the vehicle was rented,
  d. The vehicle's odometer reading at the start of the rental period,
  e. The vehicle's odometer reading at the end of the rental period,
  f. The number of kilometers driven during the rental period,
  g. The amount of money billed to the customer for the rental period,

All output should be appropriately labeled and formatted. The amount of money billed should be displayed with a dollar sign and will be rounded to two fractional digits (for example, $125.99 or $43.87).

6. The program should also detect and report invalid classification codes. When an invalid classification code is detected, the program will display an error message as well as the invalid code and customer’s name. After displaying this information the program should end.

7. The program will assume that all other user inputs are valid and correct. That is, the program will not check the number of days or odometer readings for validity

Non-functional Specifications:

1. Include brief comments in your code identifying yourself, describing the program, and describing key portions of the code.

2. Assignments are to be done individually and **must be your own work**. Software may be used to detect cheating.
3. Use Python coding conventions and good programming techniques, for example:
   i. Meaningful variable names
   ii. Conventions for naming variables and constants
   iii. Use of constants where appropriate
   iv. Readability: indentation, white space, consistency

The name of the file you submit should be your UWO userid_Assign1.py. For instance, my assignment would be oola_Assign1.py. Make sure you attach your python file to your assignment; DO NOT put the code inline in the textbox.

Make sure that you develop your code with Python 3.5 as the interpreter. TAs will not endeavor to fix code that uses earlier versions of Python.

What You Will Be Marked On:

1. Functional specifications:
   - Does the program behave according to specifications?
   - Does the program handle invalid input?
   - Is the output according to specifications?
2. Non-functional specifications: as described above
3. Assignment submission: via OWL assignment submission