CS1026: Assignment 1

Good Morning Canada!

Due: October 7th 2020, 9pm
Weight: 5%

>Learning Outcomes

By completing this assignment, you will gain skills relating to

- Basic Python programming constructs
- Expressions and decisions
- Getting input from users
- Validating input
- Algorithm development and testing; designing test cases
- Following program specifications.

>Tasks

In this assignment, you will write a **complete** program in Python that computes the cost of breakfast at the **Good Morning Canada!** restaurant. Your program is expected to prompt the user for input and validate it before computing the results. Your program will make use of expressions, decisions, and input/output in Python. You should name your program `Assign1.py`.

>Functional Specifications

1. The program will prompt the user for various pieces of information about the desired breakfast. The required information is described below. Some of the information is dependent on the type of breakfast being ordered. Once all the information has been entered, the program will compute and display the amount of money (including tax) charged for the customer's breakfast.

2. The customer can build a custom breakfast from the following individual food and beverage items: egg ($0.99 each), bacon ($0.49 per strip), sausage ($1.49 each), hash brown ($1.19 each), toast ($0.79 per slice), coffee ($1.49 per cup), and tea ($1.09 per tea bag). For example, the customer can order two eggs with two pieces of toast and a cup of coffee.

3. Alternatively, the customer can order a small, regular, or big breakfast:

i. The small breakfast includes one egg, one hash brown, two slices of toast, two strips of bacon, and one sausage.

ii. The regular breakfast includes two eggs, one hash brown and two slices of toast, four strips of bacon, and two sausages.

iii. The big breakfast includes three eggs, two hash browns, four slices of toast, six strips of bacon, and three sausages.

iv. The prices for these breakfast choices are based on the prices of the individual food items that compose them.

v. The customer can add any of the food and drink items listed above to a small, regular, or big breakfast order. For example, a customer may order a regular breakfast and add a coffee and an additional sausage to it.

4. The program will display the available choices in the following way:

```
Enter item (q to terminate): small breakfast, regular breakfast, big
breakfast, egg, bacon, sausage, hash brown, toast, coffee, tea:
```

and the user will type the customer's choice. For instance, if the customer wants a big breakfast, the user will type: `big breakfast <enter>`. The program will then ask for the quantity (how many big breakfasts). The program continues to ask for additional menu items by re-displaying the choices. When the customer is done ordering, the user enters `q`, the program then displays the pre-tax total, the tax, and the total with tax. Other scenarios are possible, in which the customer decides to customize the breakfast order.

5. The program will compute the total cost of the order with additional taxes of 13%. All the costs are to be rounded to the nearest penny, and displayed with a dollar sign and two decimal positions. For example, a total cost of 13.66666 will be displayed as $13.67. All numeric output should have 2 decimal places (e.g. 1.01, 0.00, etc).

6. The program will compute the prices for the small, regular, and big breakfast based on the prices of the items that they include. You are **not allowed** to pre-compute the costs of these breakfasts and then hard code these literal numbers in the program.

7. The program must be able to compute the total breakfast cost for an entire table of customers. For example, three customers may be sitting at the same table, where customer John orders the big breakfast with coffee, customer Jane orders the regular breakfast with an extra two strips of bacon and tea, and customer Linda chooses to have two eggs with two slices of toast and a strip of bacon. The user would enter the quantities for menu item, e.g. 1 big breakfast, 1 regular breakfast, 1 coffee, 1 tea, 2 eggs, 2 slices of toast, 3 strips of bacon.

8. The program must request user input **in the same order as per the example runs below. That is to say, the program will ask for the menu item and then for its quantity, not the other way around**.

9. Your program must accept inputs whether they contain upper-case or lower-case characters (for example, `BiG breaKfasT` and `big breakfast` should both be accepted). Additionally, your program must be robust to leading and trailing spaces, including cases when multiple spaces separate words in input lines (for example `small breakfast` and `small breakfast` should both be accepted). A Python function that performs this type of input formatting is provided to you:

```
def  formatInput(textLine) :
     textLine = textLine.lower().strip()
     wordList = textLine.split()
     textLine = " ".join(wordList)
     return textLine
```

10. Your program must also detect and report invalid input; that is, the input must match one of the keywords or phrases exactly (ignoring upper case and spaces). **When an invalid input is detected, the program will display an error message, and prompt for the input until the user enters a correct input.**

11. Your program must be robust to users entering input other than numbers when quantities are requested. **That is, you should input the string and validate that the input is numeric.** This can be done using `isnumeric()`. So to test whether the value of a variable quantity is actually a number you can do `quantity.isnumeric()`. This will return True if it is a number and False otherwise.

12. Finally, an automated testing program will run a number of test cases against your program. Some examples of output and test cases are presented below; **these are NOT comprehensive - you should create your own test cases and thoroughly test your program.**

>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 academic dishonesty (cheating).

3. Use Python coding conventions and good programming techniques. For example:

- Meaningful variable names
- Conventions for naming variables and constants
- Use of constants where appropriate
- Readability, indentation, and consistency

The name of the file you will submit must be your `Assign1.py`.

Make sure you attach your Python source file to your assignment submission; do not put the code inline in the textbox.

Make sure that you develop your code with Python 3.8 as the interpreter; failure to do so may result in the testing program failing (see below).

>Marking of the Assignment

1. **Your program will be executed by an automated testing program.** This testing program assumes that:
   - The program name is `Assign1.py`.
   - That you are using Python 3.8.
   - That you have submitted it via OWL by uploading it.

Failure to adhere to these constraints will likely cause the testing program to fail. This may require a remarking of your program which will include a 10% penalty.

2. Functional specifications:
   - Does the program behave according to the specifications found in the assignment document?
   - Does the program handle invalid input?
   - Is the output according to specifications? Has your program followed the instructions for input and output?

3. Non-functional specifications as described above

>Example executions and output

Ordering a Small Breakfast (input displayed in green in all examples, for your convenience):

```
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: small breakfast
Enter quantity :1
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: q

Cost :    6.23
Tax :     0.81
Total :   7.04

Process finished with exit code 0
```

Ordering a Custom Breakfast:

```
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: egg
Enter quantity :2
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: toast
Enter quantity :2
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: hash brown
Enter quantity :1
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: coffee
Enter quantity :1
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: q

Cost :    6.24
Tax :     0.81
Total :   7.05

Process finished with exit code 0
```

Ordering for a Table of Three Customers:

```
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: big breakfast
Enter quantity :2
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: small breakfast
Enter quantity :1
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: coffee
Enter quantity :1
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: tea
Enter quantity :2
Enter item (q to terminate): small breakfast, regular breakfast, big breakfast, egg,
bacon, sausage, hash brown, toast, coffee, tea: q

 Cost :    41.74
  Tax :     5.43
Total :    47.17

Process finished with exit code 0
```