EMAT10007 – Introduction to Computer Programming

Exercises – Week 7. Reading and Writing Files

7.1 Reading and Writing Files

Essential Questions

Exercise 1 - Writing files

1. Write the following table to a new file called samples.txt:

Sample	Mass(kg)
A	0.600
B	0.455
C	0.550
D	0.505
E	0.550

   Hint: Remember to close the file!
2. Add the new entries: (Sample F: Mass = 0.505 kg, Sample G: Mass = 0.535 kg) to the table in samples.txt.

Exercise 2 - Reading files

1. Write a Python program to read the contents of the table you have just created in samples.txt and print its contents to the console in Spyder.
2. We can convert the iterable file object returned by `open` as a list of lists by casting using `list(<object_name>)`.
   Use this method and then print the third row of the table (excluding the headings).
3. Print the numerical data in the column Mass as a list of `float` values, shown in grams (g)
   Hint: Exclude the column heading from the list. Convert string data to numerical data.
4. Print the minimum value of mass in the table (**Hint:** You can use Python's built in `min` function)

Exercise 3 - Reading and writing files

1. Edit the table in samples.txt so that the column headings are in upper case letters.
   Hint: You can use Python's built-in `upper` method (example below) which can be used to convert string data to uppercase letters (there is an equivalent `lower` method for converting to lower case).

```
txt = "Hello"
x = txt.upper()
print(x)
```

>> `HELLO`

2. Write a program that:
   - opens the file price_per_item.csv and prints the contents to display the data
   - adds another line (you can make up a new entry to the list of foods and prices).
   - prints the new contents to confirm the new line has been added
3. Use your answer to Exercise 3.2 to write a function that:
   - opens the file price_per_item.csv and prints the contents to display the data
   - asks the user to input a new food item and price
   - adds the new line to price_per_item.csv
4. Store the function in a separate file and call it from your main program.

Advanced Questions

(A) Write a program that edits the Mass of sample B to be 0.485 kg in samples.txt.

(B) Write a program that edits the table saved in samples.txt so that all Mass data is rounded to 2 decimal places (the nearest 10g)
Hint: Use the built-in `round` function.

7.2 Imported Modules for Reading and Writing Files:

Essential Questions

Exercise 4 - Writing csv files

1. Use the `csv` module to write the table in Exercise 1.1 to a csv file, samples.csv.
2. Add the new entries: (Sample F: Mass = 0.505 kg, Sample G: Mass = 0.535 kg) to the table in samples.csv.
3. Us the `csv` module to write a list of numbers to a text file, using spaces as the delimiter.

Exercise 5 - Reading csv files

1. Read the data in douglas_data.csv and print it to the Console in Spyder.
2. Print the maximum value (**Hint:** Python built in `max` function) of `density` in the data set.

3. The Python function `sorted()` takes an iterable (e.g. list, string) as an argument and returns it as a sorted list. The argument `reverse` (default value `False` determines whether the items are sorted in ascending or descending (reverse) order.

```
nums = [8, 1, 2]
print(sorted(nums, reverse=True))
```

>> `[8, 2, 1]`

Read the data in sample.csv (created in Exercise 4). Print the values in the Mass column of the imported data in ascending order.

4. In statistics, the standard deviation (SD), σ, is a measure of the amount of variation in a set of values. Low SD indicates that values in a data set tend to be close to the mean, μ, high SD indicates that values are spread over a wider range. The SD is found by taking the square root of the variance, where the variance is the average of the squared difference from the mean μ.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}}$$

Write a program that reads the file price_per_item.csv and finds the mean and the SD of the values in the 'item_price' column.

Exercise 6 - Reading and writing a csv file

1. Write a program that uses the `csv` module to:
   - open the file price_per_item.csv and prints the contents to display the data
   - asks the user to input a new food item and price
   - adds the new line to price_per_item.csv

Advanced Questions

(A) The volume of each sample in samples.csv (created in Exercise 4) in cm^3 is: A = 336, B = 231, C = 350, D = 272, E = 300, F = 312, G = 255. Write a program that:

- reads the contents of the samples.csv file .
- finds the density of each sample in kg/m^3.
- writes two new columns to the table: volume(cm3) and density(kg/m3)

Hints

- Open the file using a mode specifier that lets you read *and* write e.g. ''r+'.
- Numerical data is imported from a csv file as string data.
- After reading, the position is at the end of the file. To erase it's contents from some position use `truncate(position)`. To return to the beginning of the file use `seek(0)`.
- Data is stored in a Python program as data structures such as lists. Often we want to organise our data as columns, not rows. We can't write a column explicitly in Python, as we would in Excel. Instead data can be rearranged into lists that when written to a file, will arrange the data in columns using a loop (+ list comprhension)

```
places = [1, 2, 3, 4, 5]
names = ['Elena', 'Sajid', 'Tom', 'Farhad', 'Manesha']
scores = [550, 480, 380, 305, 150]
```

```
data = [places, names, scores]

with open('sample_data/scores.csv', 'w', newline='') as f: # no gap betwee

writer = csv.writer(f)

for i in range(len(places)):
    writer.writerow([d[i] for d in data])
    # OR
    #writer.writerow([places[i], names[i], scores[i]])
```

[An identical process can be used to to the inverse operation : we can transform imported data arranged in columns into lists so that it's easier to use in the Python program]

(B) `sorted()` can be also used to sort one list using the values in another list by using `zip` to group the two lists element wise.

```
nums = [8, 1, 2]
txt = ['a', 'b', 'c']
lists = zip(nums, text)
s_lists = sorted(lists) # [(1, 'b'), (2, 'c'), (8, 'a')]
s_nums = [i[0] for i in s_lists] # [1, 2, 3]
s_txt = [i[1] for i in s_lists] # ['b', 'c', 'a']
```

Using this example, write a program that:

- asks the user's for a player's name and score.
- include the new player and score in the table to that the scores remain in descending order.
- writes the sorted table to scores.txt.
- prints a message to the Console to tell the user if the new score is the highest in the table e.g. `'[Player name] got a new high score of [score]!'`