IE5600 – Applied Programming for Industrial Systems
AY 2023/24 Semester 1
Practical Lab 04 – Basic Python Data Structures

Section A – Tutorial Discussion

Question A1 – Applying Computational Problem Solving to Your Workplace

Recall that in Practical Lab 01 Question A3, you were asked to survey your workplace and identify any <u>one</u> opportunity or problem in which you could apply computational problem solving to create a useful software application. At that point in time, you had probably conceptualised a software application that is relatively small, e.g., a simple calculator program.

Now that you have learnt about control flow statements and basic data structures in Python, please survey your workplace again and identify another opportunity or problem in which you could apply these advanced programming tools to solve the problem.

Think about how control flow statements and basic data structure would play an essential role in the problem solving process. Provide a few concrete examples and elaborate on the details as much as possible without having to write any code.

Section B – Programming Exercises

Question B1 – Palindrome Checking the Easy and Fun Way (Basic)

In Python, a `str` is a sequence of characters that is structurally similar to a `list` of characters except that `str` is immutable. In other word, we can only use indexing to select an individual character in a `str` but we cannot modify an individual character.

Recall that in Practical Lab 02 Question B3, you have written a Python program to check whether a word with at most 5 alphabets is a palindrome. Write a new program that removes the word length limit to check whether a word or phrase of any number of alphabets is a palindrome. In the case of a phrases, you should strip away all whitespaces and special characters before performing the check

Sample Input	Sample Output
Madam	Is a palindrome
Apple	Is not a palindrome
Racecar	Is a palindrome
Rotator	Is a palindrome
no lemon, no melon	Is a palindrome

| Eva, can I see bees in a cave? | Is a palindrome |
| The brown fox jumps over the lazy dog! | Is not a palindrome |

Question B2 – Mean Calculator (Basic)

Write a program that asks user to input a **list** of positive numbers (including floating point numbers), as many as required, one at a time. Thereafter, the program should calculate the three mean numbers of the dataset.

The formulas of the three mean numbers are provided below:

- Arithmetic mean: $A = \frac{x_1 + \cdots + x_n}{n}$

- Geometric mean: $G = \sqrt[n]{x_1 \cdot \ldots \cdot x_n}$

- Harmonic mean: $H = \frac{n}{\frac{1}{x_1} + \cdots + \frac{1}{x_n}}$

Sample Input	Sample Output
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	5.500, 4.529, 3.414
15, 55, 9, 63, 80, 100, 45, 63, 26, 75	53.100, 42.813, 30.843
3.142, 55.5, 80, 90, 10, 65.6, 75.5, 45.5, 30, 25.5	48.074, 34.161, 17.155

Question B3 – Concatenate Two Lists Index-wise (Basic)

Write a program that asks user to input two **list** of string values separately. For each **list**, the string values are separated or delimited by a single space character.

Then concatenate the two **list** index-wise, i.e., the string values at index 0 of both **list** would be concatenated together. Create a third new list to contain the concatenated results starting from the 0th index item from both **list**, then the 1st index item, and so on till the last element. If either one of the original **list** has insufficient items, the remaining new values would just be the original values of the longer **list**.

Sample Input	Sample Output
M na i Ke y me s lly	['My', 'name', 'is', 'Kelly']
I am shorter I am longer by three words	['II', 'amam', 'shorterlonger', 'by', three, 'words']

Question B4 – Concatenate Any Number of Lists Index-wise (Intermediate)

Write a program that asks user to input any number of `list` of string values separately. For each `list`, the string values are separated or delimited by a single space character.

Then concatenate all the `list` index-wise, i.e., the string values at index 0 of all `list` would be concatenated together. Create an additional new list to contain the concatenated results starting from the 0th index item from all `list`, then the 1st index item, and so on till the last element. If any of the original `list` has insufficient items, the remaining new values would just be the original values of the those `list` that are long enough.

Sample Input	Sample Output
H na w Ke e m a ll r e s y	['Her', 'name', 'was', 'Kelly']
I am shorter I am longer by three words I am much longer by four words	['III', 'amamam', 'shorterlongermuch', 'bylonger', 'threeby', 'wordsfour', 'words']

Question B5 – Students' Age Database (Intermediate)

Suppose the University would like to write a simple Python program to track the age of students. It would suffice to store the name of student in addition to age in year. You should assume that each student has an unique name. Some common tasks that the program should perform include querying the age of a student, computing the average age of all students, and counting the number of students of each age.

Which Python basic data structure – `list`, `dict`, `set` and `tuple`, would be most appropriate for writing this program?

Write the program using the data structure that you have chosen to perform the following use cases. You should **NOT** need to apply any formal sorting or searching algorithms. Please use only the built-in methods that are provided by the appropriate Python basic data structure. Do **NOT** import any Python libraries.

S/N	Use Case	Description/Business Rules	Sample Input/Output
1	Input name and age of a student.	• Prompt user to input the name and age in year. • Save the name and age into the data structure. • There is no need to save the data into a file or database. • There is no need to validate age. • If a name already exists, display an error message, and discard the data.	N/A

S/N	Use Case	Description/Business Rules	Sample Input/Output
2	Query the age of a student.	• Prompt user to input a name and display the age of the corresponding student if it exists in the data structure. • You may assume that the name is entered in the correct casing. • If the name does not exist, print out a warning message.	N/A
3	Compute the average age of all students.	• Compute the arithmetic mean of the age of all students in the data structure and print out the average age in one decimal place.	N/A
4	Count the number of students of each age	• Print out a list of ages in the data structure in ascending order, and for each age, count the number of students of that age,	Age Count 18 10 19 5 21 21 22 19 23 10 25 10

Question B6 – Students' Information Database (Advanced)

Suppose the University would like to extend the Python program in Question B3 into a general students' information database that store only the name, age, gender, and telephone number of students. In the real-world, it is likely that multiple students would share the same name.

Suggest an appropriate data structure that can be used in Python to write this program. You should clearly state the dimensionality and type(s) of the data structure.

Write the program using the data structure that you have proposed to perform the following use cases. You should **NOT** need to apply any formal sorting or searching algorithms. Please use only the built-in methods that are provided by the appropriate Python basic data structure. Do **NOT** import any Python libraries.

S/N	Use Case	Description/Business Rules	Sample Input/Output
1	Input information of a student.	• Prompt user to input the name, age in year, gender (M for Male and F for Female), and telephone number. • Save the data into the data structure. • There is no need to save the data into a file or database. • There is no need to validate the input data.	N/A

S/N	Use Case	Description/Business Rules	Sample Input/Output
2	Query the information of a student.	• Prompt user to input a name and then display the number of matching students first. • You may assume that the name is entered in the correct casing. • Thereafter, display the age, gender and telephone of the corresponding student(s) if it exists in the data structure. • If there are multiple students with the same name, print out the information for all matching students. • If the name does not exist, print out a warning message.	N/A

3	Count the number of students of each gender and age, i.e., cross tabulation.	• Print out a cross tabulation of gender and age in the data structure in ascending order, showing the count of students for each subgroup.	Gender Age Count
			Female 18 10
			          19 5
			          21 21
			          22 19
			          23 10
			          25 10
			Male 20 8
			          21 20
			          22 15
			          23 15