# Seminar Worksheet 13 Solution: Nested Lists

CS 101 Algorithmic Problem Solving

Fall 2023

Name(s): ______________________________

HU ID *(e.g., xy01042)*: ______________________________

## 1. My Bill

Ali goes grocery shopping and receives his receipt in the form of a nested list, denoted as $X$, where each nested list contains the name of the item bought and its price.

Define a function *Bill*($X$) that takes this receipt $X$ as a parameter and returns the total amount of money Ali spent on grocery shopping.

**Constraints**

- $1 \leq len(X) \leq 20$

**Interaction**

The input comprises a single line containing a nested list $X$.

The output must contain a single number denoting the total money spent by Ali.

**Sample**

| Input | Output |
|---|---|
| [['Milk', 10.99 ], ['Cake', 12.50], ['Tapal Family Mixture', 450], ['Shan Bombay Biryani', 70]] | 543.49 |
| [['Shampoo', 100 ], ['Soap', 120], ['Coffee', 200]] | 420 |

In the first case, we add prices given in each nested list so 10.99 + 12.5 + 450 + 70 gives 543.49.

In the second case, we add prices given in each nested list so 100 + 120 + 200 gives 420.

**Exercise** In the space provided, indicate the outputs for the given inputs.

| Input | Output |
|---|---|
| [['Milk', 109 ], ['Fruits', 125], ['Biryani Masala', 45], ['Vegetables', 270]] | 549 |
| [['Mangoes', 1000]] | 1000 |
| [['Mangoes', 1000], ['Juice', 350], ['Milk', 400]] | 1750 |

**Problem Identification**
Briefly explain the underlying problem you identified in the above question that led you to your solution.

**Answer:**

Input: $X$
Output: The sum of the prices of all items purchased by Ali

**Pseudocode**

```
def Bill(X):
    sum = 0
    for i in X:
        sum += i[1]
    print(sum)
```

**Dry Run**
Using any one of the inputs provided in the Exercise section above, dry run your pseudocode in the space below.

1. **Initialize** *sum* to 0.
2. **Start the loop:**
   a. **First iteration:** $['Milk', 109]$ - Add 109 to *sum*.
   b. **Second iteration:** $['Fruits', 125]$ - Add 125 to *sum*.
   c. **Third iteration:** $['BiryaniMasala', 45]$ - Add 45 to *sum*.
   d. **Fourth iteration:** $['Vegetables', 270]$ - Add 270 to *sum*.

**Loop ends, and the total sum (549) is printed, which is the expected output**

## 23. Homework Help

Sarah is studying matrices in her math class. She has gotten homework on the topic of 'Matrix Subtraction' but she wants to write a Python code to do her homework quickly.

In Python, we can implement a matrix as a nested list (a list inside a list). We can treat each element as a row of the matrix.

Write a function, matrixSubtraction, that takes two integer matrices $A$ and $B$ in the form of nested lists and returns a matrix whose entries are the *subtraction* of the corresponding entries in matrices $A$ and $B$. If matrix subtraction is not possible then return -1.

**Constraints**

- $len(A) = len(B)$
- $1 \leq len(A) \leq 10$
- $1 \leq len(B) \leq 10$

**Interaction**

The input comprises a single line containing space-separated nested lists each representing $A$ and $B$.

The output must contain a nested list representing the result of matrix subtraction.

**Sample**

| Input | Output |
|---|---|
| [[4,8],[3,7]] , [[1,0],[2,5]] | [[3,8],[1,2]] |
| [[2,0],[8,6]] , [[1],[5]] | -1 |

In the first case, we perform simple matrix subtraction to obtain the resulting matrix.

In the second case, subtraction is not possible since the matrices are of different sizes.

### Exercise

In the space provided, indicate the outputs for the given inputs.

| Input | Output |
|---|---|
| [[2,0],[8,6]] , [[1,0],[5,2]] | [[1,0], [3,4] |
| [[2,0],[8,6],[4,9]] , [[1,2],[5,6]] | -1 |
| [[2,1],[8,6],[4,9]] , [[1,1],[5,6],[2,7]] | [[1,0], [3,0], [2,2]] |

### Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

**Answer:**

Input: Integer matrices $A$ and $B$
Output: Subtraction of the corresponding entries in matrices A and B

### Pseudocode

```
def matrixSubtraction(A,B):
   if len(A) == len(B):
       C = [[] for x in range (len(A))]
       for i in range(len(A)):
           if len(A[i]) == len(B[i]):
               for j in range(len(A[i])):
                   C[i].append (A[i][j] - B[i][j])
           else:
               return -1
       return C
   return -1
```

### Dry Run

Using any one of the inputs provided in the Exercise section above, dry-run your pseudocode in the space below.

- Check if the number of rows in $A$ is equal to the number of rows in $B$:
  - Both matrices have 2 rows, satisfying the condition.
- Initialize matrix $C$ to an empty matrix with the same dimensions as $A$.
- Loop through rows ($i$):
  - Check if the number of columns in $A[i]$ is equal to the number of columns in $B[i]$:
    - For the first row, both have 2 columns, satisfying the condition.
  - Subsequently, loop through each column ($j$):
    - For $i = 0, j = 0$: Compute $2 - 1$ and append the result, 1, to the first row of $C$.
    - For $i = 0, j = 1$: Compute $0 - 0$ and append the result, 0, to the first row of $C$.
    - For $i = 1, j = 0$: Compute $8 - 5$ and append the result, 3, to the second row of $C$.

    * For $i = 1, j = 1$: Compute $6 - 2$ and append the result, 4, to the second row of $C$.

- Conclude by returning $C$ as the output.

$C = [[1, 0], [3, 4]]$, **which is our expected output.**

## 3. When did I eat this?

Fatima keeps a record of what she eats in her lunch every working day (that is from Monday to Friday) for each week of the month. She maintains this information in the form of a nested list, where each nested list represents a week and each element of the nested list represents what she ate on that particular day. It is assumed that she eats something unique every day. She wants to find out which week and which day she ate a particular lunch $X$.

Write a function named *when_and_what* that takes a nested list $N$ and the name of the food $X$ as parameters. The function should determine on which day and in which week she ate $X$. If she did not eat $X$ at all, the function should return -1.

**Constraints**

- $1 \leq len(N) \leq 5$

**Interaction**

The input comprises a single line containing a string and a nested list denoting the $X$ and $N$ respectively.

The output must consist of a single number denoting the number of the week and a string denoting which day she ate, separated by a comma.

**Sample**

| Input | Output |
|---|---|
| ’granola bar’, [[’apple’, ’sandwich’, ’chicken skewer’, ’mango’, ’juice’], [’noodles’, ’pasta’, ’watermelon’, ’smoothie’, ’cup cake’], [’fried rice’, ’burger’, ’granola bar’, ’eggs’, ’melon’]] | 3, Wednesday |
| ’eggs’, [[’apple’, ’sandwich’, ’chicken skewer’, ’mango’, ’juice’], [’noodles’, ’pasta’, ’watermelon’, ’smoothie’, ’cup cake’]] | -1 |

In the first case, the word ’granola bar’ appears in the 3rd nested list and on index 2 which corresponds to Wednesday.

In the second case, the word ’eggs’ does not appear in any of the nested lists hence -1 is printed.

**Exercise**

In the space provided, indicate the outputs for the given inputs.

| Input | Output |
|---|---|
| ’pasta’, [[’apple’, ’sandwich’, ’chicken skewer’, ’mango’, ’juice’], [’noodles’, ’pasta’, ’watermelon’, ’smoothie’, ’cup cake’], [’fried rice’, ’burger’, ’granola bar’, ’eggs’, ’melon’]] | 2, Tuesday |
| ’melon’, [[’apple’, ’sandwich’, ’chicken skewer’, ’mango’, ’juice’], [’noodles’, ’pasta’, ’watermelon’, ’smoothie’, ’cup cake’]] | -1 |

**Problem Identification**

Briefly explain the underlying problem you identified in the above question that led you to your solution.

**Answer:**

Input: String $X$ and Nested List $N$
Output: -1 if X does not exist in N, otherwise, a single number denoting the number of the week and a string denoting which day she ate, separated by a comma.

**Pseudocode**

```
def when_and_what(X, N):
    days = [’Monday’,’Tuesday’,’Wednesday’,’Thursday’,’Friday’]
    for i in range(len(N)):
        for j in range(len(N[i])):
            if X == N[i][j]:
                return str(i+1) + ’, ’ + days[j]
    return -1
```

**Dry Run**

Using any one of the inputs provided in the Exercise section above, dry run your pseudocode in the space below.

1. Initialize the `days` list with the names of the weekdays.
2. Start the outer loop for rows ($i$) in the matrix $N$.
3. Start the inner loop for columns ($j$) in each row.
4. Check if the element at the current position ($N[i][j]$) is equal to the given input $X$ (’pasta’).
5. For the second row ($i = 1$), in the second column ($j = 1$), we find ’pasta’.
6. Return the concatenation of the row number ($1+1 = 2$) and the corresponding weekday (’Tuesday’), which is our expected output.

**This returns ‘2, Tuesday’, which is the expected output**

## 4. Eidi

Being busy last Eid, you had no time to collect your Eidi, let alone take full stock of it. You tasked your little brother to collect on your behalf. Being lazy, he stuffed the gift envelopes one inside another in an arbitrary fashion. Now that you have more time, you ask your brother to count your Eidi and text you the amount. Being a prankster, and knowing full well that you are taking an introductory programming course, he sends you a list containing Eidi amounts in rupees. However, he also included nested lists of Eidis, representing envelopes within envelopes.

For example, the list [10000, [7000, 8000], 9000, [5500, 6000], [4000, 10000, [5500, 5000]]] represents a large envelope containing PKR 10,000, then an envelope with PKR 7,000 and PKR 8,000, then another PKR 9,000, then another envelope with PKR 5,500 and PKR 6,000, and then another envelope with PKR 4,000, PKR 10,000, which contains another envelope with PKR 5,500 and PKR 5,000. Total Eidi received is PKR 70,000.

Do yourself a favor, and write a Python function named `calculateTotalEidi` for yourself that takes a nested list $N$ sent by your brother and calculates the total eidi from the given nested list $N$.

### Constraints

- $2 \leq len(N) \leq 10$

### Interaction

The input comprises a single line containing a nested list $N$.

The output must contain a single line containing the total eidi.

### Sample

| Input | Output |
|---|---|
| [650, 100, 750, [850, [[200, 800,[[700]], 400]]]] | 4450 |

In the case above, adding all the elements of each of the nested lists as $650 + 100 + 750 + 850 + 200 + 800 + 700 + 400$ gives 4450.

### Exercise

In the space provided, indicate the outputs for the given inputs.

| Input | Output |
|---|---|
| [500, 100, 750, [850, [[200, 200,[[700, 100]], 400]]]] | 3800 |
| [100, [[500, 800, 50, 700, 450, 800], 50], [800], 100, 800] | 5150 |

### Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

### Answer:

Input: Nested list $N$
Output: Sum of all elements of each of the nested lists in $N$.

### Pseudocode

```
def calculateTotalEidi(N):
  sum = 0
  for item in N:
      if isinstance(item, list):
          sum += calculateTotalEidi(item)
      else:
          sum +=(item)
  return sum
```

### Dry Run

Using any of the inputs provided in the Exercise section above, dry run your pseudocode in the space below.

1. Initialize the variable `sum` to 0.
2. Start the loop through the elements of the input list $N$.

3. For each element:
   - Check if it's a list using `isinstance(item, list)`.
   - If it's a list, recursively call the `calculateTotalEidi` function on that list and add the result to the `sum`.
   - If it's not a list, add the element directly to the `sum`.
4. Return the final `sum`.

Now, let's apply this to the given input $[500, 100, 750, [850, [[200, 200, [[700, 100]], 400]]]]$:

- For the first element, 500, add it to the `sum`.
- For the second element, 100, add it to the `sum`.
- For the third element, 750, add it to the `sum`.
- For the fourth element (a nested list):
  - For the first element in the nested list, 850, add it to the `sum`.
  - For the second element in the nested list (a nested list):
    - For the first element in this nested list, 200, add it to the `sum`.
    - For the second element in this nested list, 200, add it to the `sum`.
    - For the third element in this nested list (a nested list):
      - For the first element in this nested list, 700, add it to the `sum`.
      - For the second element in this nested list, 100, add it to the `sum`.
    - For the fourth element in this nested list, 400, add it to the `sum`.

The final sum is calculated as $500 + 100 + 750 + 850 + 200 + 200 + 700 + 100 + 400 = 3800$.

**3800 is our expected output**

## Debugging

### 5. Nested List Printer

Write a Python function called *print_nested_list* that takes a 2D list $A$ as input and formats its elements row by row. Your implementation should be able to handle lists of different sizes.

**Constraints**

- $2 \leq len(A) \leq 10$

**Interaction**

The input comprises a single line containing a nested list $A$.

The output must give the physical representation of a nested list with each row in a new line and a column separated by a space.

**Sample**

| Input | Output |
|---|---|
| [[2,3],[4,5],[6,7]] | 2 3 |
| | 4 5 |
| | 6 7 |
| [[1, 2, 3], [4, 5], [6, 7, 8, 9]] | 1 2 3 |
| | 4 5 |
| | 6 7 8 9 |

In the first case, there are 3 nested lists which means 3 rows and each nested list has 2 elements which means 2 columns.

In the second case, there are 3 nested lists and each nested list has 3, 2, and 4 elements respectively.

**Proposed Solution**

```
def print_nested_list(A)
    for nested in range(len(A)):
        for element in nested:
        print(nested)
    print()
```

**Dry Run**

Using the inputs provided in the Sample section above, dry run the proposed code solution below.

- Given input $([[1, 2, 3], [4, 5], [6, 7, 8, 9]])$, values in the loop variable *nested* will be the indices (0, 1, 2) of the outer list.
- The inner loop will try to iterate over the items in *nested*, but it would instead find indices of type `int`, producing a `TypeError` since ’int’ object is not iterable.

**Error Identification**

Briefly explain the errors you identified in the proposed code solution. Mention the line number and the errors in each line.

- Line 2: The outer loop incorrectly iterates over the indices (0, 1, 2) of the outer list A.
- Line 3: The inner loop is intended to iterate over the elements in the current nested list. However, due to the mistake, it iterates over the indices of the nested list instead of the elements. This causes a TypeError since ’int’ object is not iterable.
- Line 3: If it was iterating over the elements in the current nested list,they would be printed on a new line, which is not our desired output.

**Correct Solution**

Rewrite the lines of code you mentioned above with their errors corrected.

```
def print_nested_list(A):
   for nested in A:
       for element in nested:
           print(element, end = ’ ’)
       print()
```