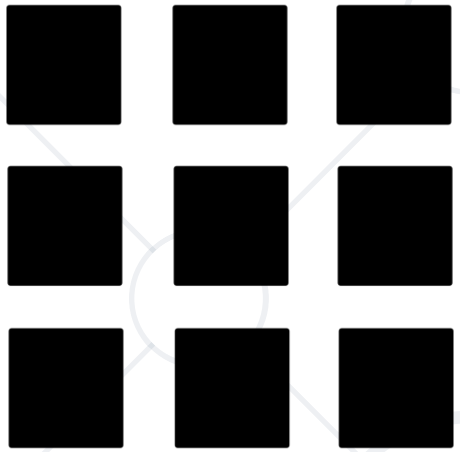
Multidimensional Lists



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#python-advanced

Table of Contents

Definition and Usage

Lists within Lists

What is Multidimensional List?

- There can be more than one additional **dimension** to lists
- Multi-dimensional lists are the **lists within lists**
- Examples:
  - A **grid** is a basic example of **two-dimensional** list
  - A **cube** is a basic example of **three-dimensional** list

Usage

- When dealing with **graphics** (pixels on the screen are in a grid formation)
- When working with **tabular** data
- Game development
- Other cases when you want **each item** of your **list** to be another **list** (Example: list of students, each of which has many tests)

Creating Multidimensional Lists

Loops and Comprehension

Creating MD List with Zeros

- Using **loops**

```
matrix = []
for i in range(3):
    matrix.append([])
    for j in range(2):
        matrix[i].append(0)
# [[0, 0], [0, 0], [0, 0]]
```

Creating 3X3 Grid with Numbers

- Using **loop**

```
matrix = []
for i in range(3):
    matrix.append([])
    for j in range(1, 4):
        matrix[i].append(j)
# [[1, 2, 3], [1, 2, 3], [1, 2, 3]]
```

Problem: Sum Matrix Elements

- Write a program that **reads a matrix** from the console and prints:
  - Sum of all **matrix elements**
  - The **matrix**
- On the first line you will get matrix sizes in the format **[rows, columns]**

```
3, 6
7, 1, 3, 3, 2, 1
1, 3, 9, 8, 5, 6
4, 6, 7, 9, 1, 0
```

```
76
[[7, 1, 3, 3, 2, 1],
[1, 3, 9, 8, 5, 6],
[4, 6, 7, 9, 1, 0]]
```

Solution: Sum Matrix Elements

```
rows, cols = [int(x) for x in input().split(", ")]

matrix = []

for row in range(rows):
    lines = [int(x) for x in input().split(", ")]
    matrix.append(lines)

# Find the sum and print it
# Print the matrix
```

MD List Comprehensions

- **Nested List Comprehensions** are nothing but a list comprehension within another list comprehension
- It is quite similar to nested **for** loops
- Usually, we use nested comprehensions when working with **multidimensional lists** (matrices)

Examples

- Creating a matrix with zeros

```
matrix = [[0 for j in range(2)] for i in range(3)]
```

- Creating a matrix with numbers

```
matrix = [[j for j in range(1, 4)] for i in range(3)]
```

- Flattening a matrix

```
matrix = [[1, 2, 3], [4, 5, 6]]
flattened = [num for sublist in matrix for num in sublist]
# [1, 2, 3, 4, 5, 6]
```

Problem: Even Matrix

- On the first line you will get the **rows of the matrix**
- On the next **N** rows, you will get **elements for each column** separated with **", "**
- Print a **new matrix** only with the numbers that are even

```
2
1, 2, 3
4, 5, 6
```

```
[[2], [4, 6]]
```

Solution: Even Matrix

```
rows = int(input())
matrix = []

for i in range(rows):
    row = input().split(", ")
    matrix.append(list(map(int, row)))
evens = [[x for x in row if x % 2 == 0] for row in matrix]

print(evens)
```

Problem: Flattening Matrix

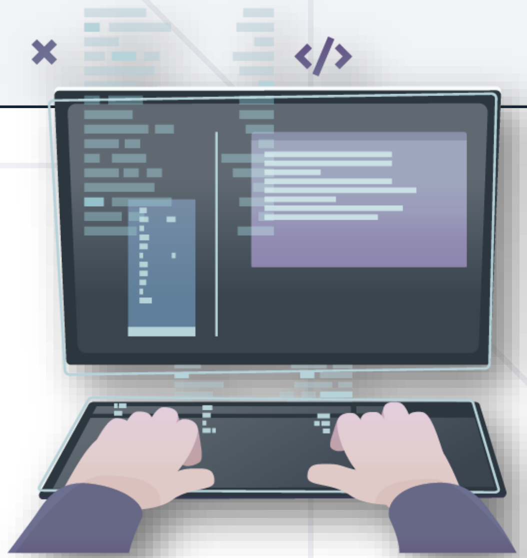
- On the first line you will receive the **number of a matrix's rows**
- On the next **N** rows, you will get **elements for each column** separated with "**,** "
- Prints the **flattened version** of it (a list with all the values)

```
2
1, 2, 3
4, 5, 6
```

```
[1, 2, 3, 4, 5, 6]
```

Solution: Flattening Matrix

```
rows_count = int(input())
matrix = [map(int, input().split(', ')) for _ in range(rows_count)]
flattened = [num for row in matrix for num in row]
print(flattened)
```



x[2][3]

Traversing and Manipulation

Accessing Elements

- To access an element in a two-dimensional list for example, you should give the **row** and the **column** of the element

```
matrix = [[1, 2], [3, 4], [5, 6]]
print(matrix[1][0]) # 3
```

- Example with 3-dimensional list

```
matrix = [[[1, 2], [3, 4]], [[5, 6], [7, 8]]]
print(matrix[0][1][1]) # 4
```

Traversing Elements

- Using loops to traverse multidimensional lists

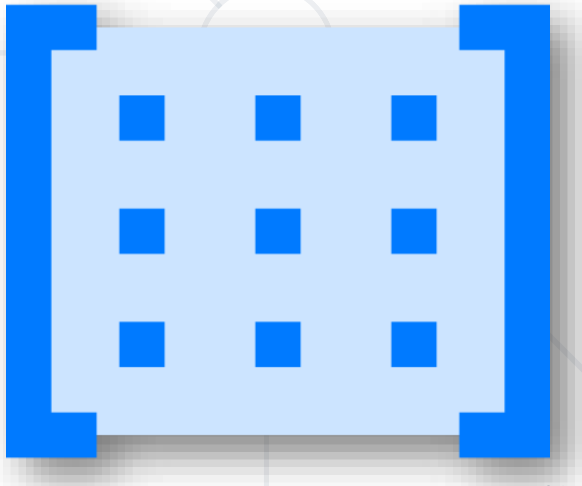
```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for i in range(len(matrix)):
    for j in range(len(matrix[i])):
        print(matrix[i][j], end=" ")
    print()
# 1 2 3
# 4 5 6
# 7 8 9
```

Traversing Elements

- Using comprehension to traverse multidimensional lists

```
[print(num) for num in [j for j in matrix]]
# [1, 2, 3]
# [4, 5, 6]
# [7, 8, 9]
```

- It is **bad practice** to use comprehensions for multidimensional lists, since the code becomes **messy** and **unreadable**



Problem: Sum Matrix Columns

- Write a program that reads a matrix from the console
- Print the **sum** for each **column**
- On the first line you will get matrix sizes in format

  [rows, columns]
- On the next rows lines, you will get elements for each column **separated** with a space

Solution: Sum Matrix Columns

```
rows_count, columns_count = [int(x) for x in input().split(', ')]

matrix = []

for _ in range(rows_count):
    row = [int(x) for x in input().split()]
    matrix.append(row)

result = []

for column_index in range(columns_count):
    column_sum = 0
    for row_index in range(rows_count):
        column_sum += matrix[row_index][column_index]
    result.append(column_sum)

[print(x) for x in result]
```

Changing Values

- Example: Increasing each value by 1

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for i in range(len(matrix)):
    for j in range(len(matrix)):
        matrix[i][j] += 1
print(matrix)
# [[2, 3, 4], [5, 6, 7], [8, 9 ,10]]
```

Problem: Primary Diagonal

- Write a program that finds the **sum of matrix primary diagonal**
- On the **first line**, you are given the integer **N** - the size of the square matrix
- The next **N lines** holds the values for **every row** - **N** numbers separated by a space

Solution: Primary Diagonal

```
size = int(input())
matrix = [[0] * size for row in range(0, size)]

for x in range(0, size):
    line = list(map(int, input().split()))
    for y in range(0, size):
        matrix[x][y] = line[y]

sum_diagonal = sum(matrix[size - i - 1][size - i - 1] for i in range(size))
print(sum_diagonal)
```

Problem: Symbol in Matrix

- Read an integer **N**, representing **rows** and **cols** of a **matrix**
- On the next **N lines**, you will receive rows of the matrix
- Each row consists of ASCII characters.
- You will receive a symbol. Find the **first occurrence,** starting from the **top left,** of that symbol in the matrix and **print its position** in the format: "`({row}, {col})`"
- If there is no such symbol print an error message "`{symbol} does not occur in the matrix`"

Solution: Symbol in Matrix

```python
size = int(input())

matrix_of_chars = []

for _ in range(0, size):
    matrix_of_chars.append(list(input()))

symbol = input()
location = ()
found = False

# Continue on next slide
```

Solution: Symbol in Matrix

```
for row in range(0, size):
    if found:
        break
    for col in range(0, size):
        if matrix_of_chars[row][col] == symbol:
            location = (row, col)
            found = True

if found:
    print(location)
else:
    print(f"{symbol} does not occur in the matrix")
```

Other Nested Structures

Sets in Lists, Lists and Sets as Dictionary Values

Nested Structures

- We can also have sets inside of lists

```
sets_of_numbers = [
    {1, 2, 3},
    {3, 4, 5}
]
```

- or tuples in lists

```
tuples_collection = [
    ("peter", "mary"),
    (22, 19)
]
```

Nested Structures

- We can also have lists as dictionary values

```
students_and_grades = {
    "peter": [4.50, 5.00, 4.95],
    "anna": [6.00, 5.65, 5.80]
}
```

- or tuples as dictionary values

```
words_and_characters = {
    "bob":  ("b", "o", "b"),
    "anna": ("a", "n", "n", "a")
}
```

Summary

- Multidimensional Lists
  - Lists within lists
  - Traversing
  - Manipulation
  - Using loops and comprehension

Software University

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

SmartIT

Coca-Cola HBC Bulgaria

SOFTWARE GROUP

DXC TECHNOLOGY

createX

Postbank

Решения за твоето утре

SUPER HOSTING .BG

DRAFT KINGS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License