Two-dimensional Lists and Dictionaries

Bachelor's Degree in Electrical and Mechanical Engineering

Carlos III University of Madrid

uc3m | Universidad Carlos III de Madrid
Departamento de Informática

Definitions

- **Two-dimensional list** is a list of lists
- **Dictionary** is a disordered, modifiable and indexed collection. There are no duplicate elements.

Two-dimensional Lists (I)

- In Python, any **table** can be represented as a list of lists (a list, where each element is itself a list).

$$V=\begin{bmatrix} 12 & 7 & 21 & 31 & 11 \\ 45 & -2 & 14 & 27 & 19 \\ -3 & 15 & 36 & 71 & 26 \\ 4 & -13 & 55 & 34 & 15 \end{bmatrix}$$

Two-dimensional Lists (II)

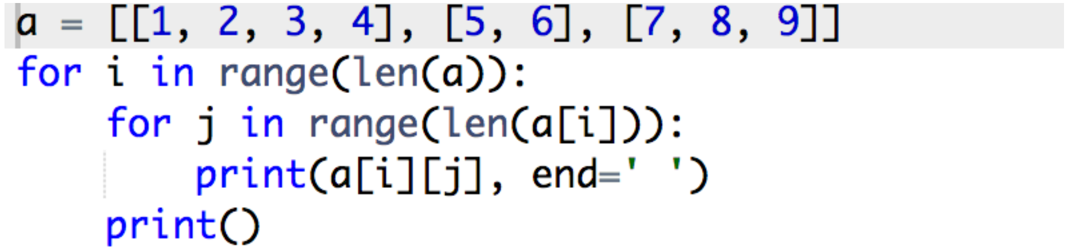
A program that creates a numerical table with two rows and three columns, and then makes some manipulations with it:

```
a = [[1, 2, 3], [4, 5, 6]]
print(a[0])
print(a[1])
b = a[0]
print(b)
print(b[0])
print(a[0][0])
a[0][1] = 7
print(a)
print(b)
b[2] = 9
print(a[0])
print(b)
```

- The first element of a is a list of numbers [1, 2, 3]
- The first element of this second list is a[0] [0] == 1;
- Also a[0] [1] == 2, a[0] [2] == 3, a[1] [0] == 4, a[1] [1] == 5, a[1] [2] == 6.

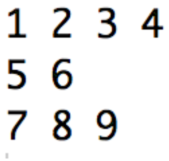
Two-dimensional Lists (III)

Print the list of elements in the two-dimensional list:



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

Output:



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

Two-dimensional Lists (IV)

Print the elements in the list:

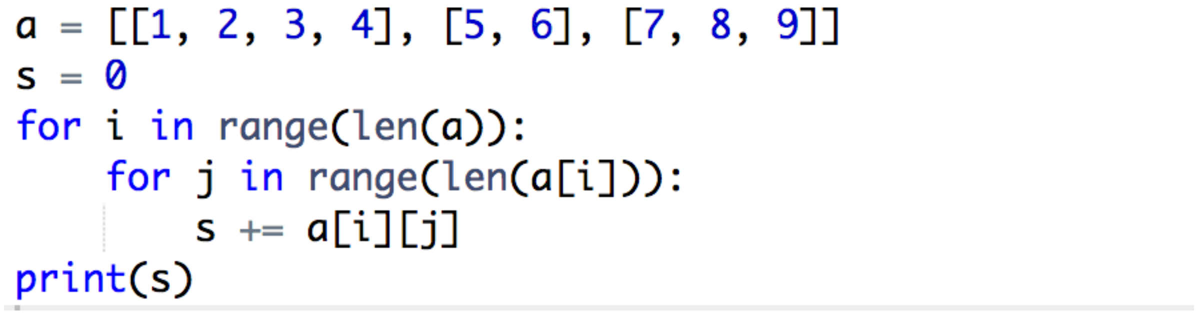
```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
for row in a:
    for elem in row:
        print(elem, end=' ')
    print()
```

Output:

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

Two-dimensional Lists (V)

- This is how you can use 2 nested loops to calculate the sum of all the numbers in the 2-dimensional list:

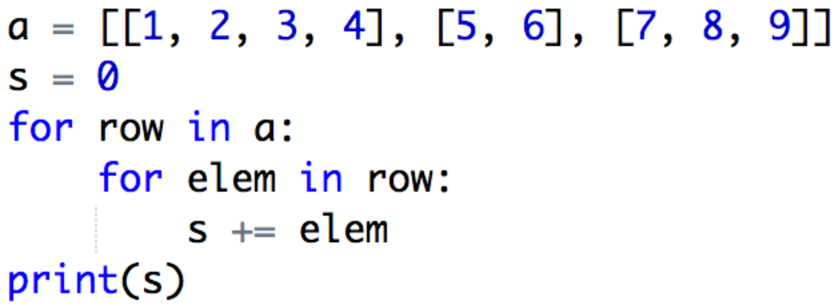


```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for i in range(len(a)):
    for j in range(len(a[i])):
        s += a[i][j]
print(s)
```

Output: 45

Two-dimensional Lists (VI)

- Or the same with iterating by elements, not by the variables i and j:



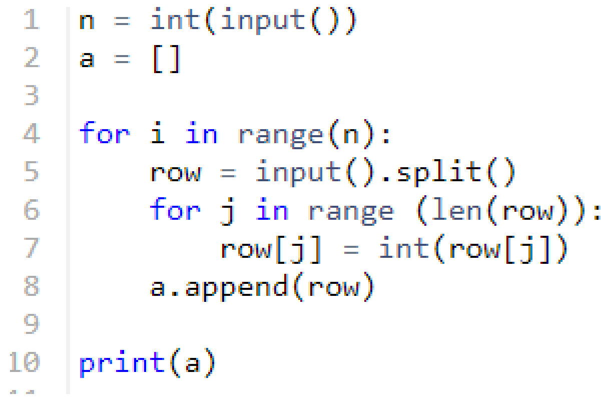
```python
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for row in a:
    for elem in row:
        s += elem
print(s)
```

Output: 45

Input

Let's say that a program takes a two-dimensional input matrix in the form of **n** rows, each of which contains **m** numbers separated by spaces.

```
Input:
3
1 2 3 4
5 6
7 8 9
```



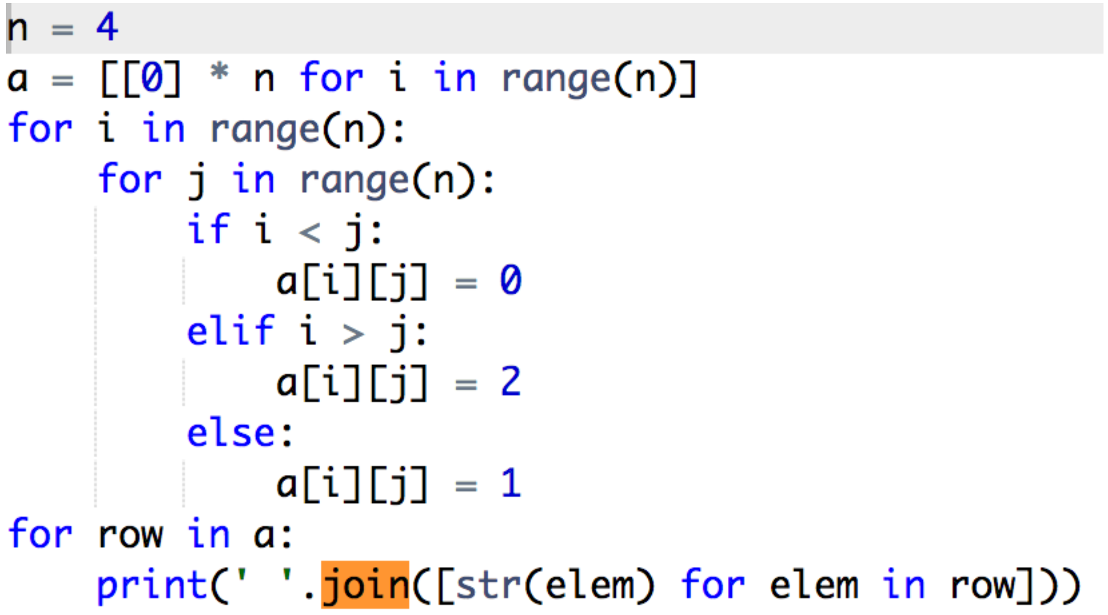
```python
n = int(input())
a = []

for i in range(n):
    row = input().split()
    for j in range (len(row)):
        row[j] = int(row[j])
    a.append(row)

print(a)
```

Method *.join* (I)

Suppose we have a square matrix. We have to put the elements of the main diagonal equal to 1, the elements above the diagonal equal to 0, and the remaining ones equal to 2. Example for a 4x4 matrix the output shall be:

```
1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1
```

Method *.join* (II)



```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    for j in range(n):
        if i < j:
            a[i][j] = 0
        elif i > j:
            a[i][j] = 2
        else:
            a[i][j] = 1
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

Sum

```
X=[[1,0,0],
   [0,2,0],
   [0,0,3]]

Y=[[0,0,1],
   [0,1,0],
   [1,0,0]]



# result is 3x3
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]

# iterate through rows of X
for i in range(len(X)):
   # iterate through columns of Y
   for j in range(len(Y[0])):
       result[i][j]= X[i][j] + Y[i][j]

for r in result:
   print(r)
```

Multiplication

```python
X=[[1,0,0],
   [0,2,0],
   [0,0,3]]

Y=[[0,0,1],
   [0,1,0],
   [1,0,0]]



# result is 3x3
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]

# iterate through rows of X
for i in range(len(X)):
   # iterate through columns of Y
   for j in range(len(Y[0])):
       # iterate through rows of Y
       for k in range(len(Y)):
           result[i][j] += X[i][k] * Y[k][j]

for r in result:
   print(r)
```

Dictionary (I)

Dictionary (II)

- A dictionary is a disorderly, modifiable and indexed collection.
- The **dictionary**, which **defines a one-to-one relationship between keys and values**.

```
thisdict = {
  "apple": "green",
  "banana": "yellow",
  "cherry": "red"
}
print(thisdict)
> {'apple': 'green', 'banana': 'yellow', 'cherry': 'red'}
```

Dictionary (III)

Let’s change the color of “apple” to “red”

```
thisdict = {
  "apple": "green",
  "banana": "yellow",
  "cherry": "red"
}
thisdict["apple"] = "red"
```

dict(): Constructor, Remove, Print, Length

```
thisdict = dict(apple="green", banana="yellow", cherry="red")
thisdict["damson"] = "purple"
print(thisdict)
> {'apple': 'green', 'banana': 'yellow', 'cherry': 'red', 'damson': 'purple'}
```

```
del(thisdict["banana"])
print(thisdict)
> {'apple': 'green', 'cherry': 'red'}
```

```
thisdict = dict(apple="green", banana="yellow", cherry="red")
print(len(thisdict))
> 3
```