

Programación

Clase 10

U

Arranglos

4 Algoritmos con Matrices



Objetivo

Comprender nuevas maneras de poblar una matriz.

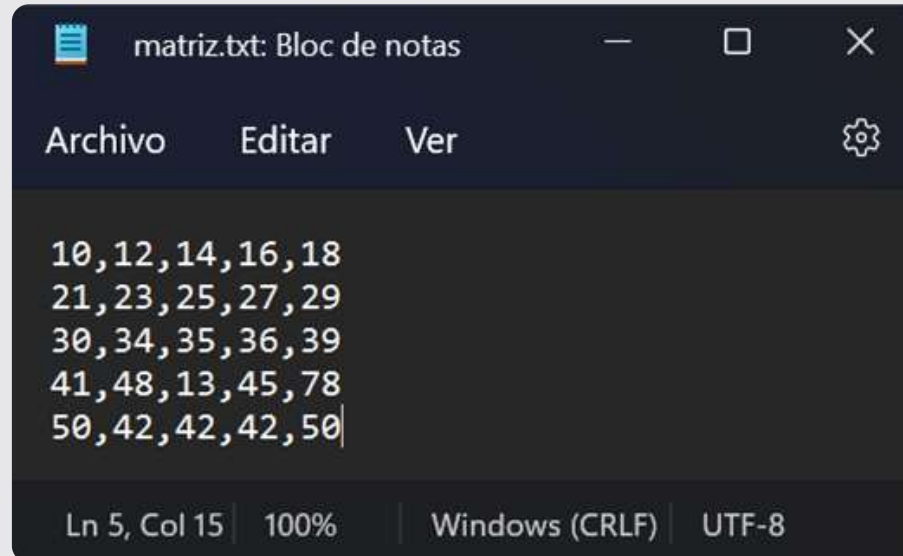
Interiorizar funcionalidades adicionales para trabajar con matrices.

Comprender cómo adaptar los algoritmos vistos anteriormente a las matrices.

Aplicar los conceptos para la resolución de problemas.

Cargando datos a una matriz

Revisemos cómo cargar los elementos de un archivo en una matriz.
Supongamos que queremos cargar los siguientes datos en una matriz:



```
matriz.txt: Bloc de notas
Archivo  Editar  Ver
10,12,14,16,18
21,23,25,27,29
30,34,35,36,39
41,48,13,45,78
50,42,42,42,50
Ln 5, Col 15 | 100% | Windows (CRLF) | UTF-8
```

The image shows a screenshot of a text editor window titled "matriz.txt: Bloc de notas". The window has a dark theme and a menu bar with "Archivo", "Editar", and "Ver". The main text area contains a 5x5 matrix of numbers, with the cursor at the end of the fifth row. The status bar at the bottom shows "Ln 5, Col 15 | 100% | Windows (CRLF) | UTF-8".

Cargando datos

```
import numpy as np
mtx = np.zeros([5,5])
arch = open('matriz.txt')
linea = arch.readline().strip()
fila = 0
while linea != '':
    partes = linea.split(',')
    for col in range(len(partes)):
        mtx[fila][col] = partes[col]
    filas += 1
    linea = arch.readline().strip()
print(mtx)
```

¿Lo podemos hacer de otra forma?

¿Qué sucede si no conozco la cantidad de elementos?

Cargando datos 1

Si un archivo de texto que contiene una matriz tuviera este formato, ¿podrías leerlo y guardarlo?

A continuación vienen 5 líneas

```
5, 3
10, 12, 14
21, 23, 25
30, 34, 35
41, 48, 13
50, 42, 42
Holaaa
```

Y cada línea tiene 3 columnas

Cargando datos 2

Si un archivo de texto que contiene una matriz tuviera este formato, ¿podrías leerlo y guardarlo?

A continuación vienen 5 líneas

Esta línea tiene 3 columnas

```
5
3,10,12,14
2,21,23
4,30,34,35,99
1,48
2,42,42
Hola
Mundo!
```

Cargando datos 3

Si un archivo de texto que contiene una matriz tuviera este formato, ¿podrías leerlo y guardarlo?

A
continuación
vienen 5
líneas

No se indica
cuántas
columnas
tiene

```
5
10,12,14
21,23
30,34,35,99
48
42,42
Hola
Mundo!
```

Encontrando elementos

¿Cómo encontramos la posición de un elemento en una lista?

Podemos hacer algo similar en las matrices pero requerimos:

- Ciclo for en las filas
- Ciclo for en las columnas
- Condicional
- Variables de posición
- Recorrer la matriz!

Recorriendo una matriz

Para esto, necesitamos saber las dimensiones de la matriz. Tenemos dos alternativas

Alternativa 1

- Usar un par de variables extras para almacenar sus dimensiones
- Y pasar esas variables cada vez que necesitemos recorrerla

```
def sumar_matriz_v1(matriz, nfilas, ncolumnas):  
    s = 0  
    for f in range(nfilas):  
        for c in range(ncolumnas):  
            s += matriz[f][c]  
    return s  
print(sumar_matriz_v1(mtx, 5, 5))
```

Recorriendo una matriz

La forma anterior funciona, pero a veces no es tan cómodo andar pasando dos variables extras en todos lados.

Alternativa 2:

```
mtx = np.zeros([5,5])
print('Esta matriz tiene', mtx.shape[0], 'filas')
print('Esta matriz tiene', mtx.shape[1], 'columnas')

mtx = np.zeros([2,3])
print('Esta matriz tiene', mtx.shape[0], 'filas')
print('Esta matriz tiene', mtx.shape[1], 'columnas')

mtx = np.zeros([21,17])
print('Esta matriz tiene', mtx.shape[0], 'filas')
print('Esta matriz tiene', mtx.shape[1], 'columnas')
```

cuántas columnas tiene

Esta matriz tiene 5 filas
Esta matriz tiene 5 columnas

Esta matriz tiene 2 filas
Esta matriz tiene 3 columnas

Esta matriz tiene 21 filas
Esta matriz tiene 17 columnas

Ojo con shape()

.shape(n) retornará el tamaño “REAL” de la matriz

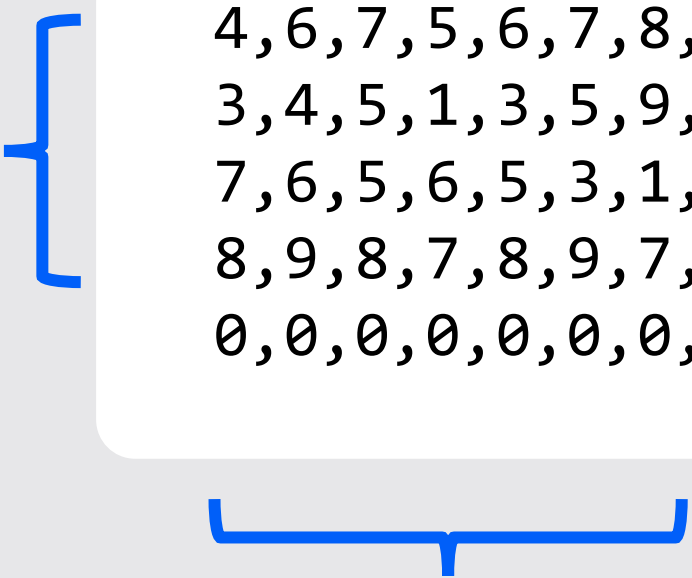
Si estamos resolviendo un problema, y hemos declarado una matriz de manera sobredimensionada (y no la estamos usando completa)

.shape(n) igual seguirá indicando el tamaño REAL

En este caso, vamos a necesitar un par de variables para indicar hasta dónde estamos ocupando la matriz

Ojo con shape()

filas
== 4



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

columnas
== 7

`mtx.shape[0]`
== 5

`mtx.shape[1]`
== 12

Encontrando elementos

```
def encontrar_elemento(elemento, matriz):  
    filas = matriz.shape[0]  
    cols = matriz.shape[1]  
    for i in range(filas):  
        for j in range(cols):  
            if elemento == matriz[i][j]:  
                return [i, j]  
    return [-1, -1]  
print(encontrar_elemento(350, mtx))  
print(encontrar_elemento(45, mtx))
```

¿Qué devuelve shape de una matriz?

¿Por qué el -1,-1?

¿Podemos hacerlo de otra forma?

```
[-1, -1]  
[3, 3]
```

Máximo y mínimo en una matriz

Para encontrar el máximo o el valor mínimo en una matriz, tendremos que recorrerla elemento por elemento para luego descubrir el máximo o el mínimo.

¿Qué sucede si hay más de uno?

¿Cómo podemos resolver el algoritmo?

El máximo en una matriz

```
def max_matriz(matriz):  
    filas = matriz.shape[0]  
    cols = matriz.shape[1]  
    maximo = -10**10  
    for i in range(filas):  
        for j in range(cols):  
            if maximo < matriz[i][j]:  
                maximo = matriz[i][j]  
    return maximo  
  
print(max_matriz(mtx))
```

¿Cómo descubro la posición del máximo?

¿Qué pasa si hay más de una?

Los máximos en una matriz

```
def maximos_matriz(matriz, posiciones):  
    filas = matriz.shape[0]  
    cols = matriz.shape[1]  
    maximo = -10**10  
    for i in range(filas):  
        for j in range(cols):  
            if maximo < matriz[i][j]:  
                posiciones.clear() # limpiamos  
                maximo = matriz[i][j]  
                posiciones.append([i,j])  
            elif maximo == matriz[i][j]:  
                posiciones.append([i,j])  
    return maximo
```

```
lista = []  
print(maximos_matriz(mtx, lista))  
print(lista)
```

Una solución posible...

Mínimo en una matriz

```
def min_matriz(matriz):  
    filas = matriz.shape[0]  
    cols = matriz.shape[1]  
    minimo = 10**10  
    for i in range(filas):  
        for j in range(cols):  
            if minimo > matriz[i][j]:  
                minimo = matriz[i][j]  
    return minimo  
  
print(min_matriz(mtx))
```

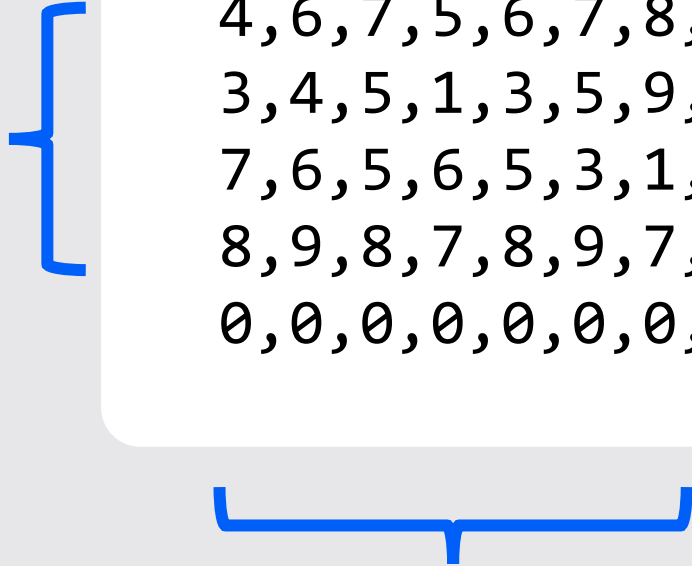
De acuerdo a cómo usamos las matrices, este código tiene un bug

O tal vez no, depende del punto de vista

Y tiene que ver con que la matriz vacía está llena de ceros, y si no tenemos cuidados, podemos considerar esos ceros como mínimo

Mínimo en una matriz

filas
== 4



4	6	7	5	6	7	8	0	0	0	0	0	0	0
3	4	5	1	3	5	9	0	0	0	0	0	0	0
7	6	5	6	5	3	1	0	0	0	0	0	0	0
8	9	8	7	8	9	7	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

columnas
== 7

En este caso, cero **NO** es
el mínimo

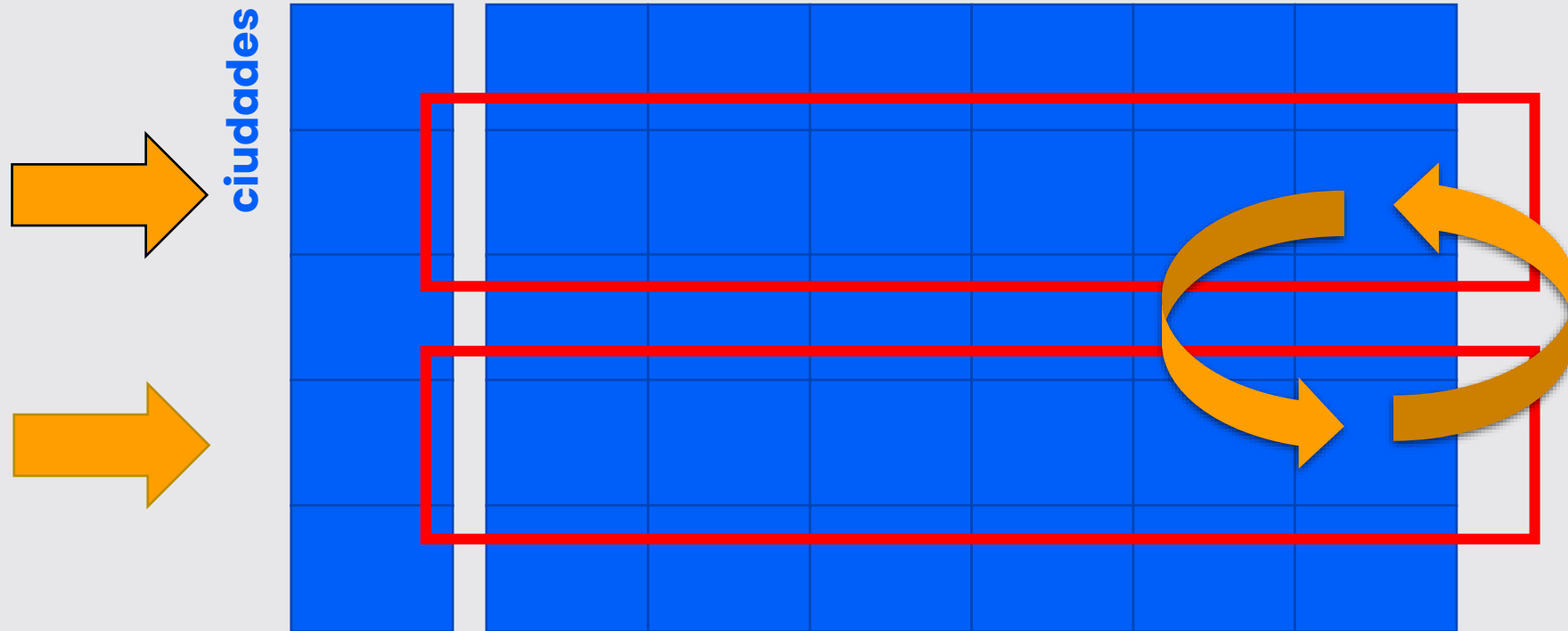
Ordenando matrices

Si sucede que hay que ordenar una lista, y esa lista tiene asociada una matriz, se debe proceder con cuidado.

Para mantener la estructura funcionando, todos los elementos de las filas de la matriz deben intercambiarse.

ciudades

Ordenando



Ordenando

```
import numpy as np
def imprimir(mensaje, lista, matriz):
    print(mensaje)
    for i in range(len(lista)):
        linea = str(lista[i]) + ":"
        # len(matriz[i]) retorna el tamaño de
        # la fila
        for j in range(len(matriz[i])):
            linea += " " + str(matriz[i][j])
        print(linea)

def intercambiar(lista, indice1, indice2):
    aux = lista[indice1]
    lista[indice1] = lista[indice2]
    lista[indice2] = aux
```

```
marcas = [ 50, 10, 60, 20 ]
# En vez de usar zeros (que retorna una matriz
# llena de ceros) usaremos random, que retorna
# una matriz llena de números aleatorios. De
# esta forma será más evidente si el algoritmo
# funciona o no.
#matriz = np.zeros([4, 6])
matriz = np.random.rand(4, 6)

imprimir("inicial", marcas, matriz)

for a in range(len(marcas) - 1):
    for b in range(a + 1, len(marcas)):
        if marcas[a] < marcas[b]:
            intercambiar(marcas, a, b)

            for c in range(6):
                aux = matriz[a][c]
                matriz[a][c] = matriz[b][c]
                matriz[b][c] = aux

imprimir("final", marcas, matriz)
```

Resultado

inicial

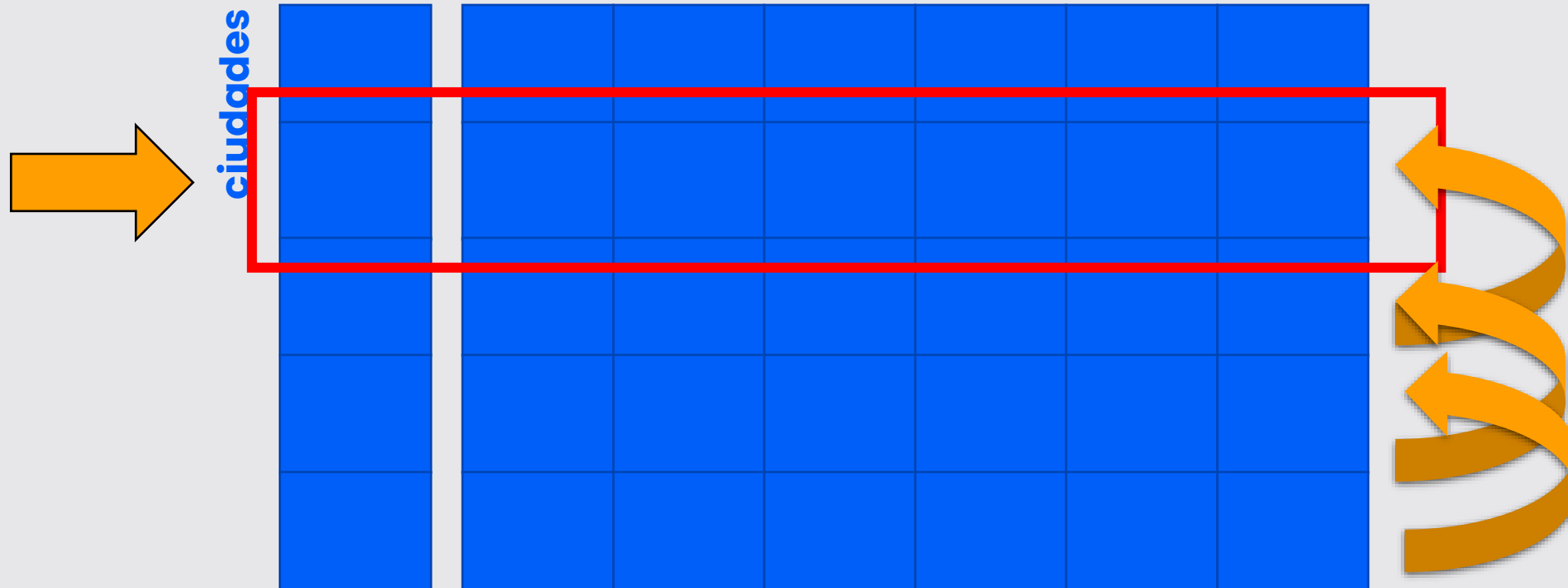
```
50: 0.0617611306444 0.927595244301 0.640445879977 0.15076906007 0.3910834342 0.59238281094
10: 0.671093703774 0.90482674226 0.0400745110967 0.209469709008 0.693388411492 0.0011082130936
60: 0.192370193521 0.378939754568 0.0123926628423 0.512387147461 0.926969251483 0.370015287436
20: 0.189682597814 0.145930556196 0.98289549959 0.668009872742 0.86581451392 0.233682915391
```

final

```
60: 0.192370193521 0.378939754568 0.0123926628423 0.512387147461 0.926969251483 0.370015287436
50: 0.0617611306444 0.927595244301 0.640445879977 0.15076906007 0.3910834342 0.59238281094
20: 0.189682597814 0.145930556196 0.98289549959 0.668009872742 0.86581451392 0.233682915391
10: 0.671093703774 0.90482674226 0.0400745110967 0.209469709008 0.693388411492 0.0011082130936
```

Eliminando columnas o filas

Si por alguna razón necesitamos “eliminar” una fila o columna, tenemos que mover TODOS los elementos que están “después” que el elemento eliminado



Eliminando columnas o filas

Fíjate que en el caso anterior, había una lista paralela a la matriz.

En este caso, para mantener la sincronía entre ambas, TAMBIÉN se debe eliminar en la lista.

Algo equivalente pasará si queremos eliminar columnas.

ciudades

Trabajo autónomo mínimo

Revisar capítulo 9 libro guía.

Resolver problemas 56, 57, 61, 65, 74.