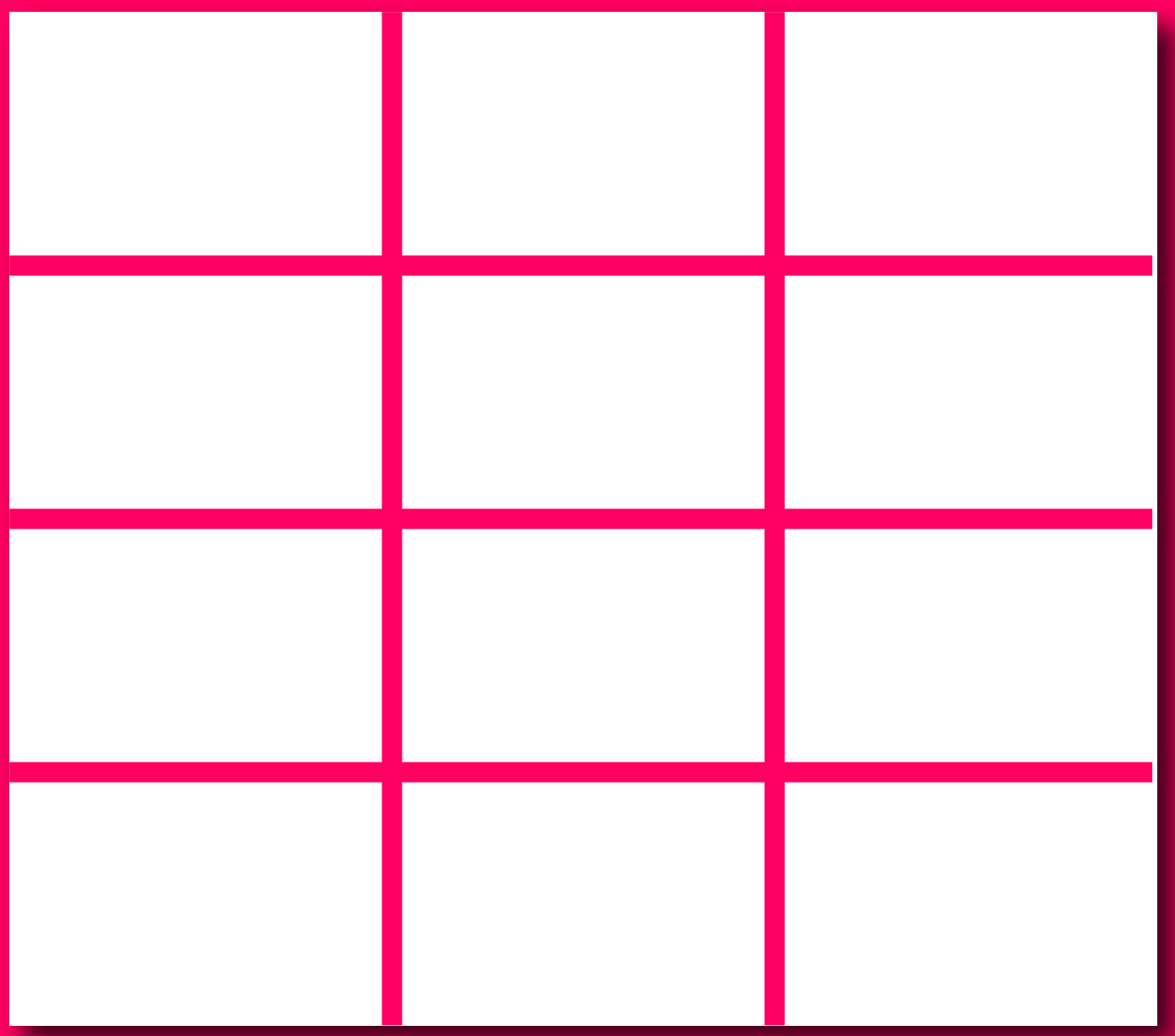
El futuro digital es de todos

MinTIC

Columnas



Filas

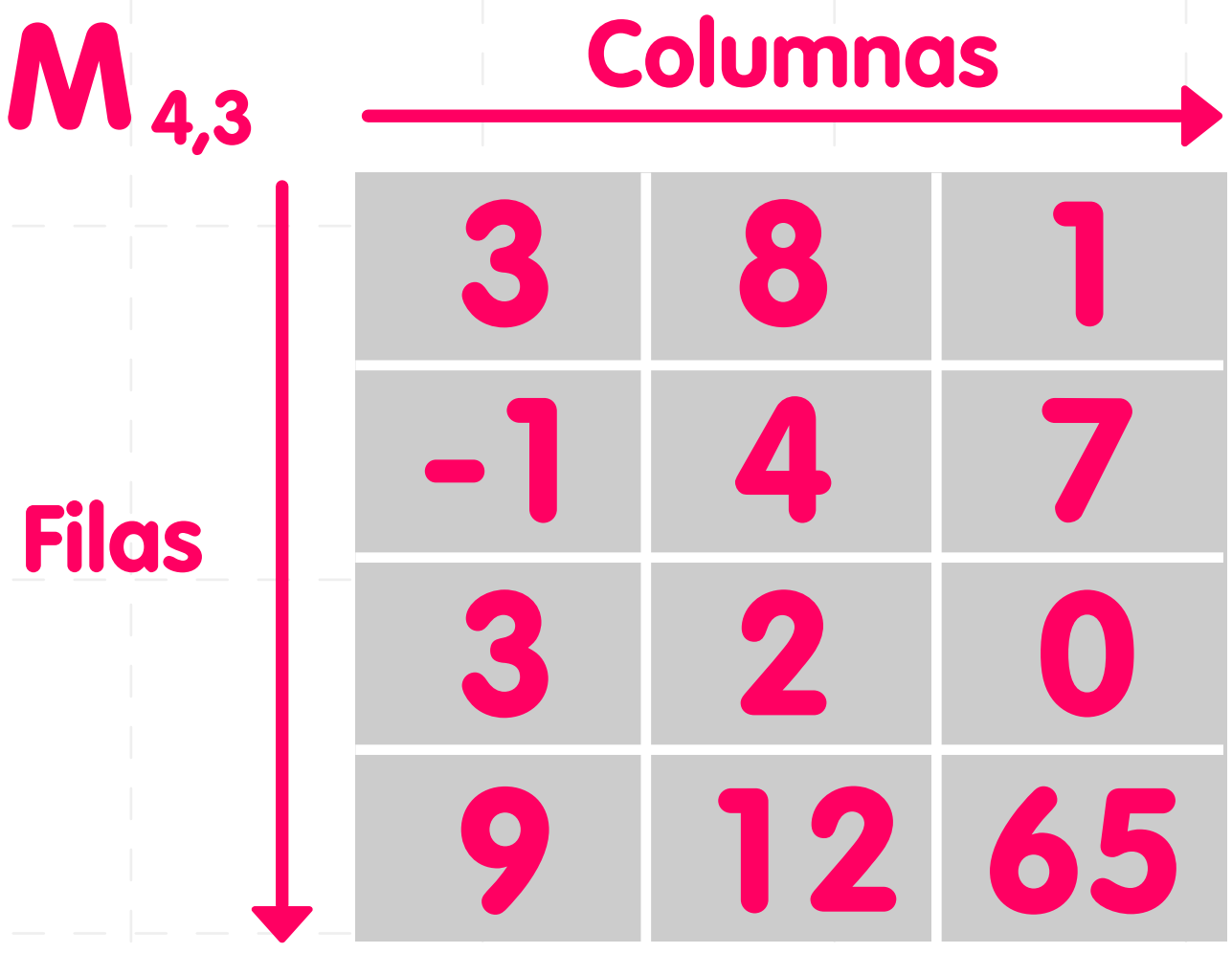
Matrices

Universidad de Caldas

Hola:

Continuamos con el tema de estructura de datos y de los algoritmos requeridos para poder manejar estas estructuras. Esta vez el turno es para las matrices.

Una matriz o un arreglo bidimensional es un conjunto de datos que tienen un mismo tipo de datos y se organiza en filas y columnas. Se debe tener en cuenta varios detalles y es que, normalmente, la definición de arreglo bidimensional o matriz es para almacenar elementos de un mismo tipo, sin embargo, hay que tener en cuenta también que la diversidad de tipos en Python es tan grande que en determinado momento podemos tener en una posición de esa matriz o de arreglo bidimensional un elemento de un tipo y después, en otra posición otro elemento de otro tipo. Estamos hablando de un elemento que no es una estructura de datos abstracta. Es importante tener en cuenta que Python no tiene una definición exclusiva para matrices.



Una matriz o arreglo bidimensional se puede representar con el identificador llamado “M” y le estamos dando el número de filas que va a tener y el número de columnas. Aquí tenemos específicamente una matriz con cuatro (4) filas y tres (tres) columnas. Esta es la representación de lo que es un arreglo bidimensional, es decir, es una organización matricial con filas y columnas.

A continuación veremos una matriz, sus elementos y las posiciones en que se encuentran los elementos. La matriz tiene N filas por M columnas, siendo N y M números enteros mayores a cero (0).

$M_{N,M}$

$M_{0,0}$	$M_{0,1}$	$M_{0,2}$	... ..	... ..	$M_{0,j}$	... ..	... .	...	$M_{0,M-1}$
$M_{1,0}$	$M_{1,1}$	$M_{1,2}$	... ..	... ..	$M_{1,j}$	... ..	... .	...	$M_{1,M-1}$
$M_{2,0}$	$M_{1,1}$	$M_{1,2}$	... ..	... ..	$M_{1,j}$	... ..	... .	...	$M_{2,M-1}$
... ..	... ..	... ..	... ..	... ..	... ..	... ..	... .	...	... ..
... ..	... ..	... ..	... ..	... ..	... ..	... ..	... .	...	... ..
$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	... ..	... ..	$M_{i,j}$	... ..	... .	...	$M_{i,M-1}$
... ..	... ..	... ..	... ..	... ..	$M_{0,j}$	... ..	... .	...	... .
$M_{N-1,0}$	$M_{N-1,1}$	M_{N-2}	... ..	... ..	$M_{N-1,j}$	... ..	... .	...	$M_{N-1,M-1}$

Este tipo de organización ya la conocemos. Lo primero que vemos allí son N filas, pero desde un punto de vista de programación se indexan desde la fila cero (0) hasta la fila N-1. Y tenemos M columnas que también van desde la columna cero (0) hasta la columna M-1.

El primer elemento importante está en la posición cero (0), es decir, ubicar un elemento en esa estructura matricial 0,0 significa que está en la fila cero (0) y en la columna cero (0). Si hacemos un recorrido por filas el segundo sería en la fila cero (0), pero en la columna uno (1) y seguiremos viendo los elementos en la fila cero (0) hasta la columna M-1. Por el otro lado, lo que se queda aquí constante es la columna y pasamos a la fila uno (1) columna cero (0), fila dos columnas cero (0) y así sucesivamente hasta que lleguemos a la última fila que es la N-1, pero en la misma columna.

Fila N-1 columna uno (1), fila N-1 columna dos (2), hasta que tenemos la posición N-1,M-1 que es la última posición de la matriz.

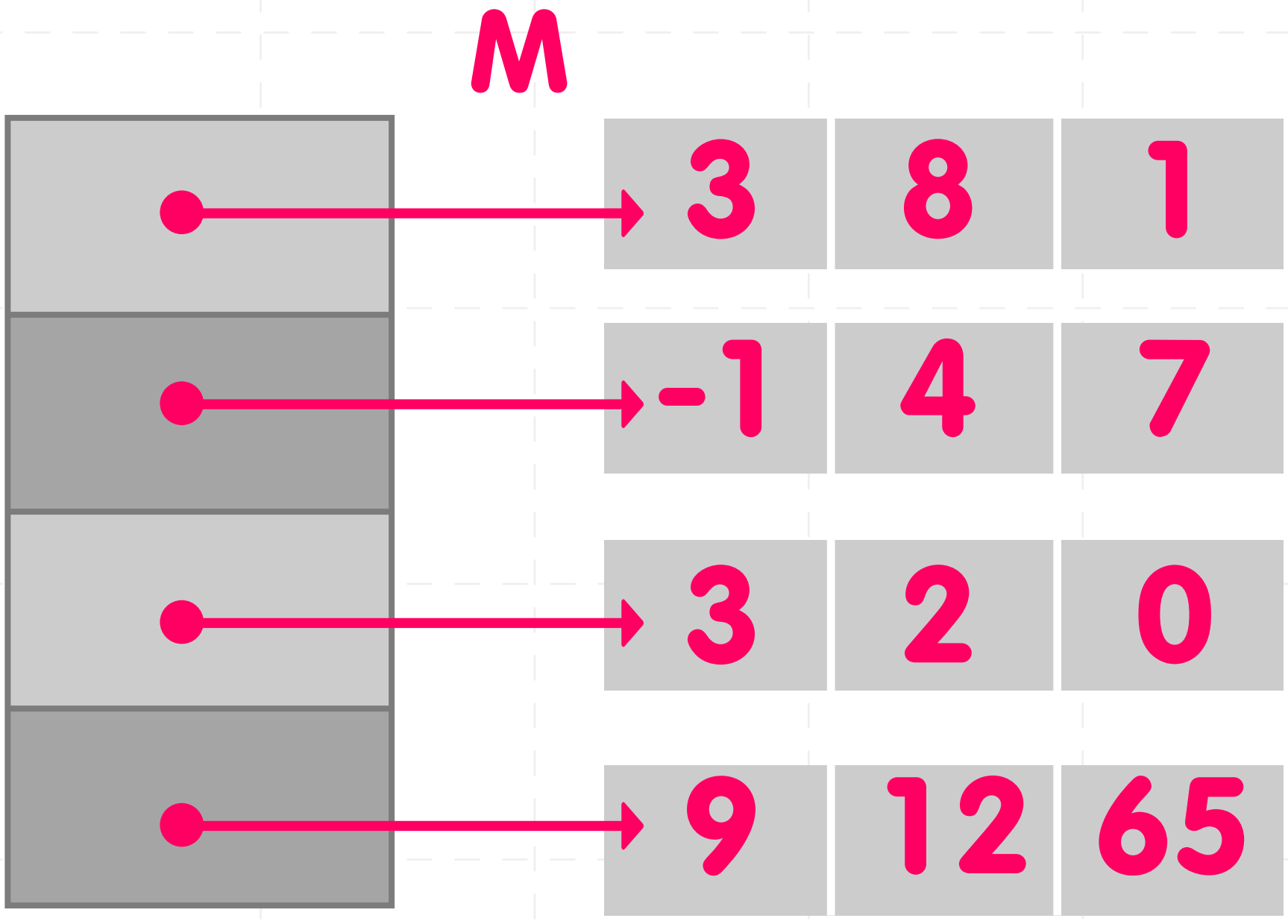
Cualquier elemento con los elementos I-J, donde I va desde cero (0) hasta N-1 es el número de filas y con J que va desde cero (0) hasta M-1. Podemos indexar cualquier elemento que se encuentre en la matriz.

Para el manejo de matrices o arreglos bidimensionales en Python no existe una estructura como podemos ver en las listas, los diccionarios y las tuplas, directa para el manejo de matrices en Python.

M

3	8	1
-1	4	7
3	2	0
9	12	65

Esta matriz que tiene cuatro (4) filas y tres (3) columnas y la forma en que Python lo maneja es creando una lista o una lista de listas, que la forma de representar arreglos en Python. En este ejemplo vemos cada posición de la lista M es otra lista



M = [[3, 8, 1] , [-1, 4, 7] , [3, 2, 0] , [9, 12, 65]]

Los corchetes iniciales nos dan la lista principal y luego los corchetes pequeños van agrupando cada uno de los elementos de cada una de las filas. Esta es la forma en que nos permite la representación y por ende la manipulación de arreglos bidimensionales o matrices.

```
M = [ [ 3, 8, 1 ] , [ -1, 4, 7 ] , [ 3, 2, 0 ] , [ 9, 12, 65 ]]
M [ 0 ] [ 0 ]
3
M [ 0 ] [ 2 ]
1
M [ 2 ] [ 0 ]
3
```

Esta manipulación nos permite hacer cualquier operación que necesitemos con matrices, sin embargo, entre esas que tenemos, por ejemplo los recorridos que son un patrón muy conocido ya que ese recorrido de matrices por filas que es el más común.

3	8	1
-1	4	7
3	2	0
9	12	65

```
M = [ [ 3, 8, 1 ] , [ -1, 4, 7 ] , [ 3, 2, 0 ] , [ 9, 12, 65 ]]
for i in range (0,4) :
    for j in range (0,3) :
        print (M[i] [j] , end = ´\ t ´)
print ( ´\ t ´)
```

Utilizamos RANGE que hace parte del ciclo for. En un rango de 0 a 4 se recorre las filas y en un rango de 0 a 3 recorre las columnas.

NumPy es una librería que permite realizar arreglos y matrices y la utilizaremos para entender mejor el contenido sobre matrices en general.

En conclusión, vimos los arreglos bidimensionales llamados matrices y que en Python no existe una estructura que se llama matriz, pero que se puede representar como una lista de listas.Además vimos como es el proceso de indexación fila-columna de los elementos que permite asignar o recuperar información.Y por último vimos un algoritmo de recorrido por las matrices y Numpy que es una librería para el manejo de matrices.

Universidad de Caldas