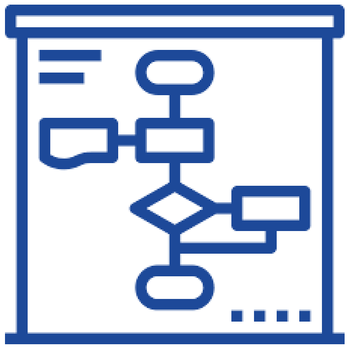
INTRODUCCIÓN A LA PROGRAMACIÓN



OTOÑO, 2022

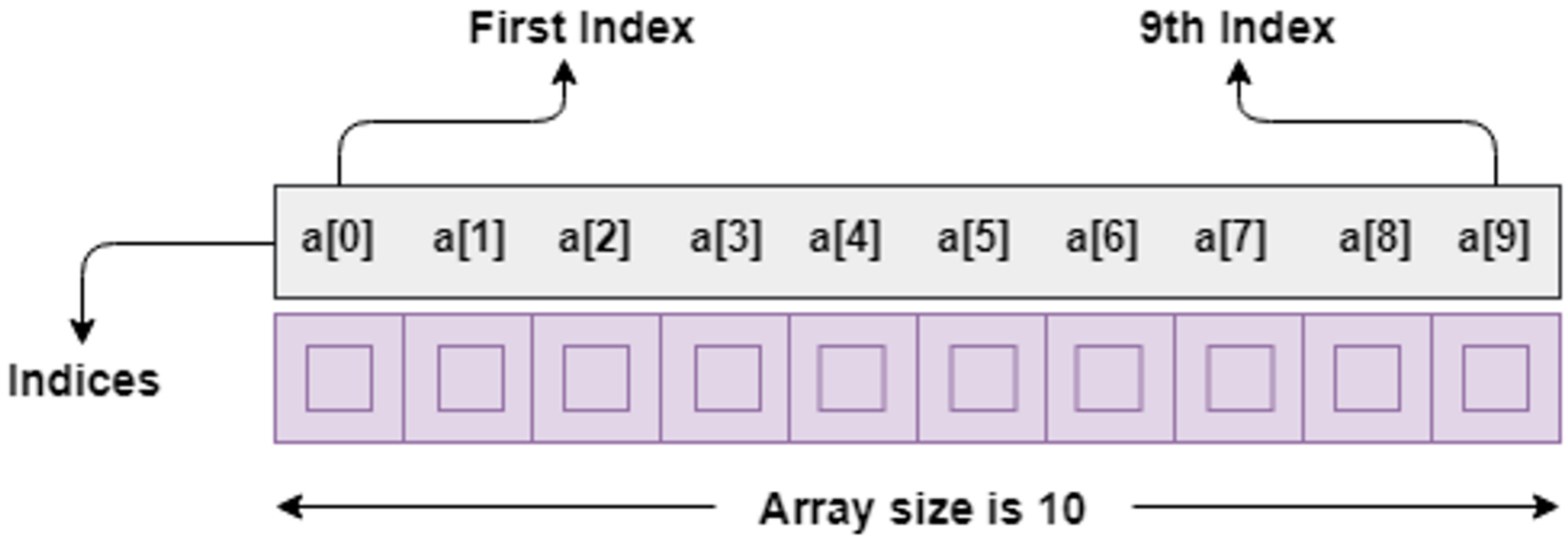
UNIVERSIDAD TECNOLÓGICA DE CHILE
INSTITUTO PROFESIONAL
CENTRO DE FORMACIÓN TÉCNICA
nacap

UNIDAD III

Arreglos, Numpy

[Material Complementario]

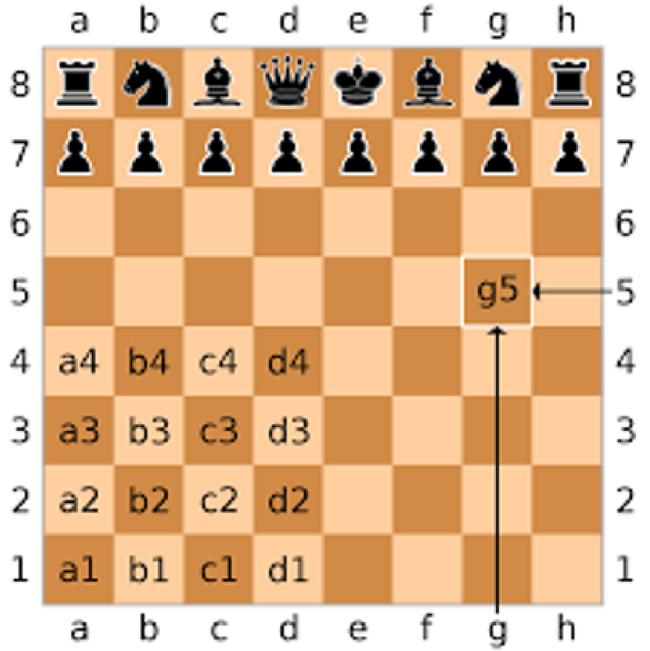
Listas -> 1 dimensión



Listas Anidadas:

Las listas pueden constar de escalares (es decir, números) y elementos de una estructura mucho más compleja

A menudo encontramos estos **arreglos** (elementos) en nuestras vidas. Probablemente el mejor ejemplo de esto sea un **tablero de ajedrez**.



Arreglos bidimensionales (Matriz)

En matemáticas, es posible definir una matriz como:

Matriz. Una matriz de dimensiones **m×n** es una tabla formada por elementos dispuestos en **m** filas y **n** columnas de la forma:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

NOTA: Los elementos de la matriz se representan con doble subíndice, A_{mn}, donde:
- el primero indica la fila a la que pertenece.
- segundo, indica a la columna que pertenece.

Arreglos -> conocidos como matrices o lista de listas,

En Python cualquier tabla se puede representar como matrices (lista de listas).

Ejemplo:
Matriz A:

A = [[1, 2], [3, 4]]

Acciones sobre arreglos:

Se tiene la siguiente matriz: $A=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$

La matriz A definida en Python: A = [[1, 2], [3, 4]]

Nota: Para encontrar cualquier elemento de una lista bidimensional, debes usar dos coordenadas:

Una vertical (número de fila).
Una horizontal (número de columna).

#Acceder a los elementos de una matriz:

Se utilizará la sintaxis A[][], donde primero indicamos la fila y, a continuación, la columna.

Ejemplo:

print(A[0][1]) # **2**

$$A=\begin{pmatrix}A_{0,0} & A_{0,1}\\ A_{1,0} & A_{1,1}\end{pmatrix}$$

Ejercicio:

a) Obtenga los 3 valores restantes de la matriz “A”:
b) Imprima las listas de la matriz “A” por separado.
c) Cambiar el número 4 de la matriz A por un 0.

Acciones sobre arreglos:

Se tiene la siguiente matriz: $A=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$

La matriz A definida en Python: A = [[1, 2], [3, 4]]

#Iterar en los elementos de una matriz:

$A=\begin{pmatrix}A_{0,0} & A_{0,1}\\ A_{1,0} & A_{1,1}\end{pmatrix}$

#Sea A una matriz,

A = [[1, 2],[3, 4]]

#conocer los elementos de la fila de la matriz A:

```
for fila in A1:
    for elemento in fila:
        print(elemento)
```

Recorrer los elementos de la matriz A:

```
for row in range(len(A)): # la variable “i”, iterará en las filas de la matriz A
       for column in range(len(A[row])): #la variable “j”, iterará en las columnas de la matriz A
       print(A[row][column]) #Se imprimirán los escalares (elementos) de la matriz A
```

Acciones sobre arreglos: Ejercicio:

Se tiene la siguiente matriz:

$$A1=\begin{pmatrix} 3 & 29 & 23 \\ 6 & 2 & 11 \end{pmatrix}$$

a) Representar la matriz A1 en un script de Python.
b) Imprimir las dos listas por separado de la matriz A1.
c) Utilizando el bucle for iterar e imprimir todos los elementos de la matriz A1.
d) Utilizando el bucle for iterar e imprimir sólo los elementos de la segunda columna de la matriz A1.

Acciones sobre arreglos: Sumar Matrices:

Se tiene la siguiente matriz:

$$A=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$$

Suma de matrices:

Para poder sumar dos matrices, necesitamos que tengan la misma dimensión. Entonces, dadas A y B dos matrices con dimensión **m×n**, su suma será una matriz de dimensión **m×n** y sus elementos se obtienen del siguiente modo:

$$A+B = (a_{ij})_{m\times n} + (b_{ij})_{m\times n} = (a_{ij}+b_{ij})_{m\times n}$$

$$A=\begin{pmatrix}A_{0,0} & A_{0,1}\\ A_{1,0} & A_{1,1}\end{pmatrix}$$

$$B=\begin{pmatrix}B_{0,0} & B_{0,1}\\ B_{1,0} & B_{1,1}\end{pmatrix}$$

$$A+B\begin{pmatrix}A_{0,0}+ & B_{0,0} & A_{0,1}+ & B_{0,1}\\ A_{1,0}+ & B_{1,0} & A_{1,1} & B_{1,1}\end{pmatrix}$$

Acciones sobre arreglos: Sumar Matrices:

```python
#Analizar el siguiente código:

A = [[1, 0, -3], [2, 0, 1]]
B = [[-1, -2, 0], [-2, 3, 0]]

m = len(A) # 2
n = len(A[0]) #3

if len(A) == len(B):
  C = []

  for i in range(m):
    C.append([])
    for j in range(n):
      C[i].append(A[i][j] + B[i][j])

  print(C)

else:
  print("No se puede realizar la suma, pues las dimensiones de las matrices no coinciden.")
```


NumPy

NumPy

- ✓ NumPy: Numbers Python
- ✓ NumPy: es una biblioteca de Python para realizar cálculos numéricos a gran escala.
- ✓ NumPy: Es utilizada para operaciones científicas con Python.
- ✓ Numpy: Nos permite trabajar con arreglos unidimensionales y multidimensionales
- ✓ NumPy: Es extremadamente útil, especialmente en Machine Learning.

Instalación de NumPy:

Mediante PIP (Python Installer Packages):

Se puede instalar NumPy utilizando PIP, realizar los siguientes pasos:

1. Ingresar a CMD, puede realizar este proceso desde ejecutar. (Tecla Windows + R) Escribiremos en la ventana de ejecutar la palabra cmd.

2. Ubicados en CMD, escribiremos el siguiente comando:

>pip install numpy

3. **Para utilizar numPy en nuestro código, debemos llamar el módulo NumPy en Visual Studio Code:**

Import numpy as np # por lo general se le asigna el alias np a NumPy

4. **Para actualizar NumPy desde cmd con PIP:**

>pip –m install numpy --upgrade

```
C:\Users\rodri>pip install numpy --upgrade
Requirement already satisfied: numpy in c:\python\lib\site-packages (1.20.3)
Collecting numpy
  Downloading numpy-1.21.1-cp39-cp39-win_amd64.whl (14.0 MB)
     |████████████████████████████████| 14.0 MB 6.8 MB/s
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.20.3
    Uninstalling numpy-1.20.3:
      Successfully uninstalled numpy-1.20.3
Successfully installed numpy-1.21.1
```

Nota uso PIP:

Actualizar PIP: en cmd *>python -m pip install -U pip*
Ver listado de paquetes instalados con PIP en Sistema: *>pip list*
Ver ayuda en PIP: *>pip --help*

NumPy:

El objeto principal de NumPy es la matriz multidimensional homogénea.

Es una tabla de elementos (generalmente números), todos del mismo tipo, indexados por una tupla de enteros no negativos.

En NumPy las cotas se denominan ejes

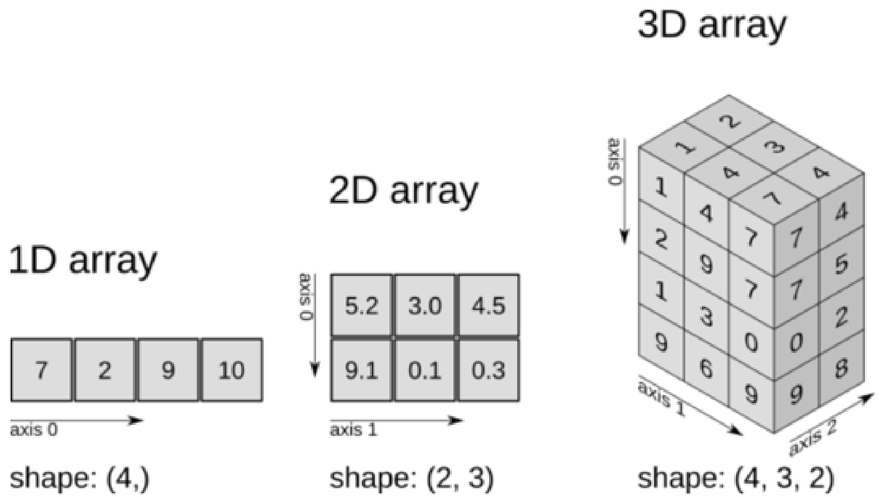
Por ejemplo, las coordenadas de un punto en el espacio **[1, 2, 1]** tienen un eje.

Ese eje tiene 3 elementos, por lo que decimos que tiene una longitud de 3.

En el ejemplo que se muestra a continuación, la matriz “a” tiene 2 ejes. El primer eje tiene una longitud de 2, el segundo eje tiene una longitud de 3.

a=[[1, 2, 3], [4, 5, 6]]

$$a=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{pmatrix}$$



Aplicación de NumPy:

```
# Array de una dimensión
a1 = np.array([1, 2, 3])
print(a1)

#[1 2 3]

# Array de dos dimensiones
a2 = np.array([[1, 2, 3], [4, 5, 6]])
print(a2)

# [[1 2 3]
#  [4 5 6]]

# Array de tres dimensiones
a3 = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
print(a3)

# [[[ 1  2  3]
#   [ 4  5  6]]

#  [[ 7  8  9]
#   [10 11 12]]]
```

Convertir arreglos en arreglos NumPy:

Para crear un array a partir de la lista o tupla lista podemos usar np.array(lista o tupla). El número de dimensiones del array dependerá de las listas o tuplas anidadas en lista:

- Para una lista de valores se crea un array de una dimensión, también conocido como **vector**.
- Para una lista de listas de valores se crea un array de dos dimensiones, también conocido como **matriz**.
- Para una lista de listas de listas de valores se crea un array de tres dimensiones, también conocido como **cubo**.
- No hay límite en el número de dimensiones del array más allá de la memoria disponible en el sistema.

```
# Transformar una lista en un arreglo NumPy:

a=[[1, 2, 3], [4, 5, 6]]
a=np.array(a)
print(a)

#Resultado:
[[1 2 3]
 [4 5 6]]
```

a=[[1, 2, 3], [4, 5, 6]]

a=[[1, 2, 3], [4, 5, 6]]

NumPy:

1. **Clase ndarray: Alías (array),** Los atributos más importantes de un ndarray objeto son:

   a) ndarray.ndim: Indica el número de ejes (dimensiones) de la matriz.

   Ejemplo: ***print(a.ndim)*** # Mostrará como resultado 2 dimensiones.

   b) ndarray.shape: Indica las dimensiones de la matriz. Esta es una tupla de números enteros que indica el tamaño de la matriz en cada dimensión.
   Para una matriz con **n** filas y **m** columnas, shape será **(n,m)**. La longitud de shape por lo tanto tupla es el número de ejes, **ndim**.

   Ejemplo: ***print(a.shape)*** # Mostrará como resultado (2, 3), dos filas y 3 columnas

   c) ndarray.size: Indica el número total de elementos de la matriz. Esto es igual al producto de los elementos de shape.

   Ejemplo: ***print(a.size)*** # Mostrará como resultado 6 elementos.

   d) ndarray.dtype: Es un objeto que describe el tipo de elementos de la matriz. Uno puede crear o especificar dtype usando tipos estándar de Python. Además, NumPy proporciona tipos propios. **numpy.int32, numpy.int16 y numpy.float64** son algunos ejemplos.

   Ejemplo: ***print(a.dtype)*** # Mostrará como resultado int32

e) ndarray.itemsize: Indica el tamaño en bytes de cada elemento de la matriz. Por ejemplo, una matriz de elementos de tipo float64 tiene itemsize 8 (= 64/8), mientras que uno de tipo complex32 tiene itemsize 4 (= 32/8).

Ejemplo: **print(a.itemsize)** # Mostrará como resultado 4 (=32/8).

f) ndarray.data: el búfer que contiene los elementos reales de la matriz. Normalmente, no necesitaremos usar este atributo porque accederemos a los elementos en una matriz usando facilidades de indexación.

Ejemplo: ***print(a.data)*** # Mostrará como resultado un valor similar a: <memory at 0x00000269A1F372B0>

NumPy:

2. Crear Arrays:

Se puede crear una matriz a partir de una lista o tupla de Python normal utilizando la función array. El tipo de la matriz resultante se deduce del tipo de elementos en las secuencias.

Ejemplo:

```
b=(1,4,3,5,10)
b=np.array(b) # [ 1  4  3  5 10]
```

A menudo, los elementos de una matriz se desconocen originalmente, pero se conoce su tamaño.

- NumPy ofrece varias funciones para crear matrices con contenido inicial de marcador de posición.
- Éstos minimizan la necesidad de aumentar las matrices, una operación costosa.

- La función zeros crea una matriz llena de ceros, la función ones crea una matriz llena de unos y la función empty crea una matriz cuyo contenido inicial es aleatorio y depende del estado de la memoria.

```
ceros=np.zeros((2,2))
print(ceros)

#Resultado
[[0. 0.]
 [0. 0.]]
```

```
unos=np.ones((2,2))
print(unos)

#Resultado
[[1. 1.]
 [1. 1.]]
```

NumPy:

2. Crear Arrays:

Para crear secuencias de números, NumPy proporciona la función **arange** que es análoga a la de Python incorporada range, pero devuelve una matriz.

Ejemplo crear matriz de 1D:

```
d1=np.arange(4)
print(d1) #[0 1 2 3]
```

Ejemplo crear matriz de 2D:

```
d2=np.arange(6).reshape(2, 3) #reshape (r,c)
print(d2)

#[[0 1 2]
 [3 4 5]]
```

Ejemplo crear matriz de 3D:

```
d3=np.arange(8).reshape(2, 2, 2) #reshape (s, r, c)
print(d3)

#[[[0 1]
  [2 3]]

 [[4 5]
  [6 7]]]
```

inacap.cl