



GOBIERNO DEL  
ESTADO DE MÉXICO



**TECNOLÓGICO DE ESTUDIOS SUPERIORES DE IXTAPALUCA  
ORGANISMO PÚBLICO DESCENTRALIZADO DEL GOBIERNO DEL ESTADO DE  
MEXICO**

---

**PRACTICA #2.**

**DIVISIÓN**

**INGENIERÍA EN SISTEMAS COMPUTACIONALES.**

**QUE PRESENTAN:**

**GERARDO MARTINEZ BARRIOS.**

**ORLANDO RAMIREZ CURIEL.**

**JUAN MANUEL SEGOVIA GARCIA.**

---

Ing. Ebner Juárez Elías  
Asesor Técnico

---

Ing. Ebner Juárez Elías  
Asesor Metodológico

Noviembre 24 de 2021.

## **Introducción**

La importancia del análisis de datos para la toma de decisiones.

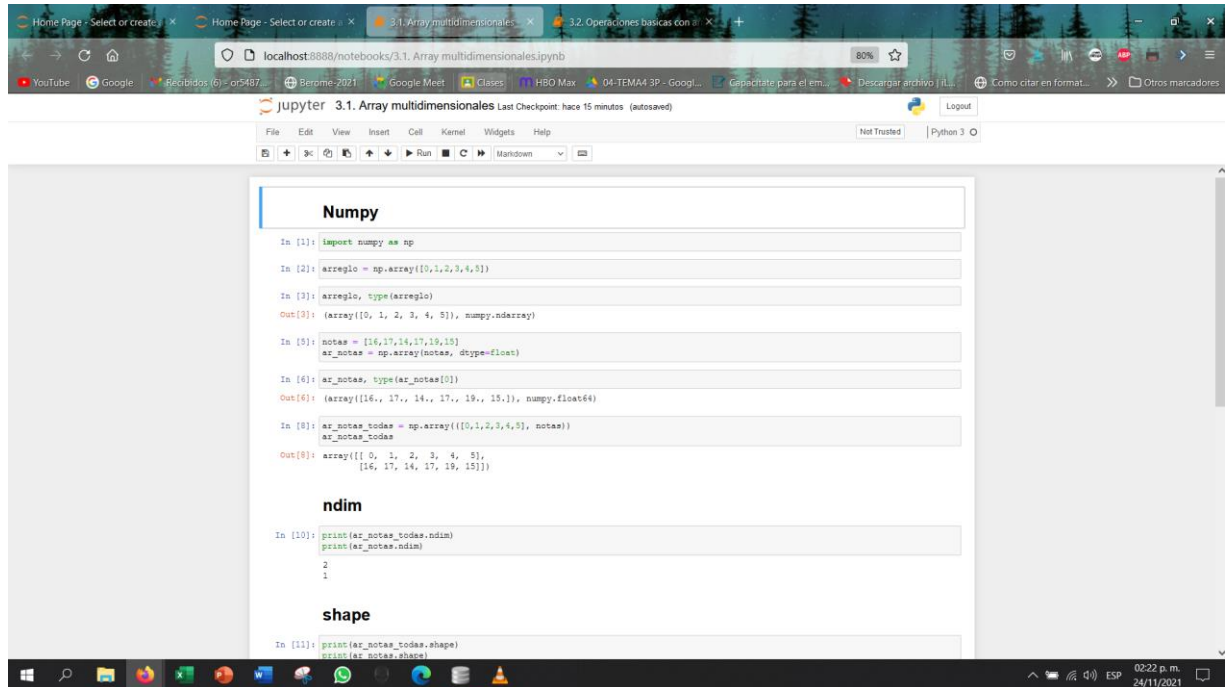
Antes de tomar cualquier decisión es vital tener una visión completa del panorama para así explorar más en las diferentes opciones que tenemos. Sin conocer nuestros alcances, nuestras pérdidas y opciones, difícilmente tomaremos una decisión correcta, y es que en los negocios casi nunca se trata de seguir una corazonada. De esta manera, el análisis de datos nos ofrece una imagen completa y real de lo que le está pasando a nuestra empresa en términos de ventas, rentabilidad, utilidad, pérdidas, inventarios, seguimiento y mucho más.

Hablar sobre la información que podemos obtener cuando se lleva un correcto control de un negocio, ya sea a través de una hoja de cálculo de Excel o algo más robusto como un sistema administrativo, sería un tema interminable, y más bien dependerá de lo que cada empresa quiera analizar y de sus objetivos para poder extraer la información necesaria de sus bases de datos para así tomar acción. Pero, ¿Cómo se logra extraer la información correcta para el análisis de datos?

Para contestar esta pregunta son esenciales dos cosas, la primera sería una herramienta que te permita compilar la información y datos en un mismo programa y la segunda, una persona que sea capaz de extraerla e interpretarla.

# Desarrollo

## 3.1 Creamos arrays multidimensionales con np.array



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
Numpy

In [1]: import numpy as np

In [2]: arreglo = np.array([0,1,2,3,4,5])

In [3]: arreglo, type(arreglo)
Out[3]: (array([0, 1, 2, 3, 4, 5]), numpy.ndarray)

In [5]: notas = [14,17,14,17,19,15]
ar_notas = np.array(notas, dtype=float)

In [6]: ar_notas, type(ar_notas[0])
Out[6]: (array([14., 17., 14., 17., 19., 15.]), numpy.float64)

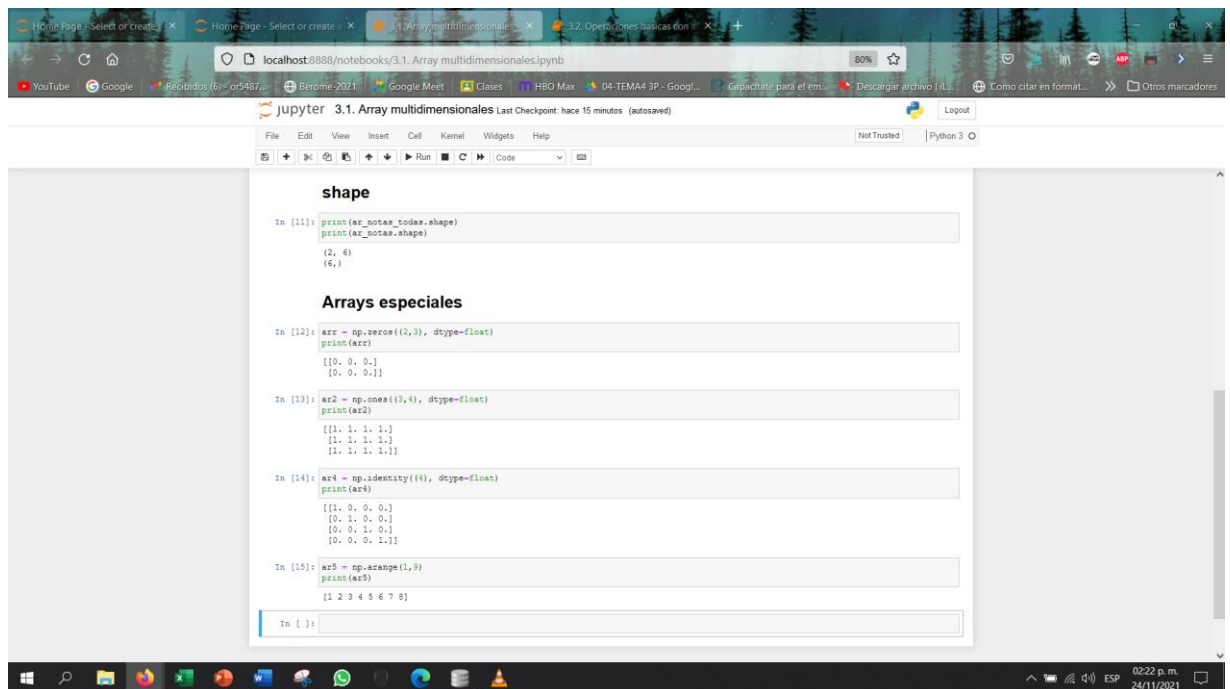
In [8]: ar_notas_todas = np.array([0,1,2,3,4,5], notas)
ar_notas_todas
Out[8]: array([[0, 1, 2, 3, 4, 5],
               [14, 17, 14, 17, 19, 15]])

ndim

In [10]: print(ar_notas_todas.ndim)
         print(ar_notas.ndim)
2
3

shape

In [11]: print(ar_notas_todas.shape)
         print(ar_notas.shape)
```



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
shape

In [11]: print(ar_notas_todas.shape)
         print(ar_notas.shape)
(2, 6)
(6,)

Arrays especiales

In [12]: ar1 = np.zeros((2,3), dtype=float)
         print(ar1)
[[0. 0. 0.]
 [0. 0. 0.]]

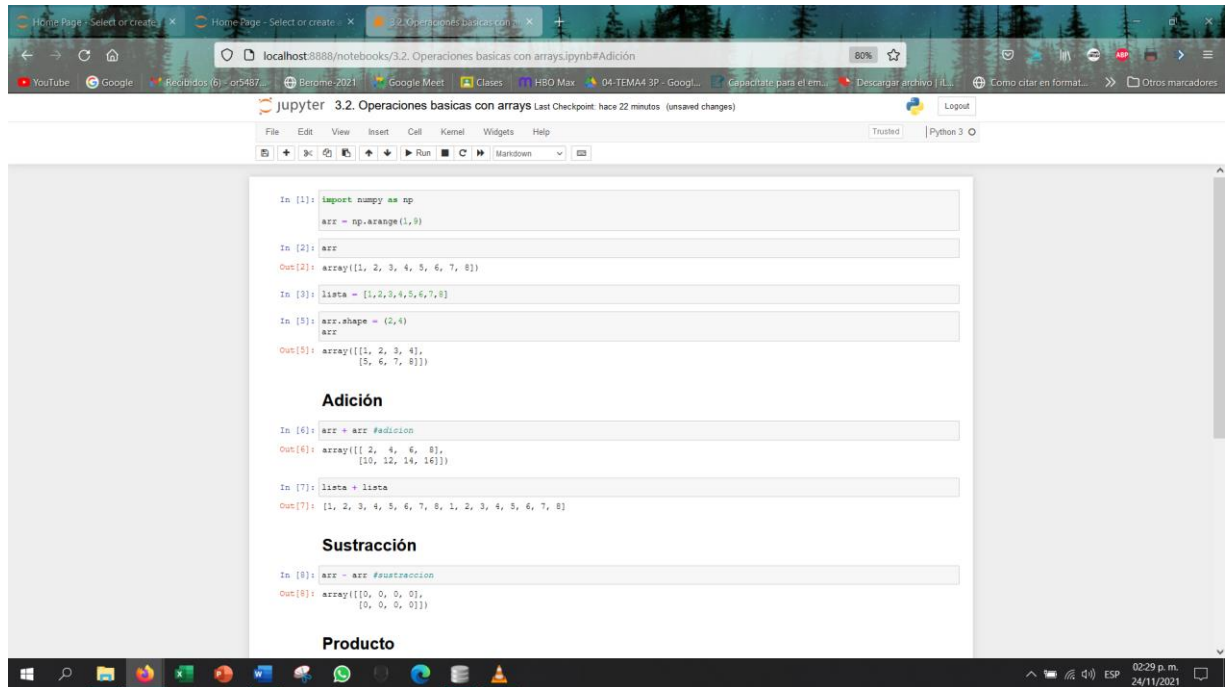
In [13]: ar2 = np.ones((3,4), dtype=float)
         print(ar2)
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]

In [14]: ar4 = np.identity((4), dtype=float)
         print(ar4)
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

In [15]: ar5 = np.arange(1,9)
         print(ar5)
[1 2 3 4 5 6 7 8]

In [ ]:
```

3.2 Importamos las librerías numpy y np y con nuestras listas en el array hacer las operaciones con los códigos normales como +,\*,-,etc.



```
In [1]: import numpy as np
arr = np.arange(1,9)

In [2]: arr
Out[2]: array([1, 2, 3, 4, 5, 6, 7, 8])

In [3]: lista = [1,2,3,4,5,6,7,8]

In [5]: arr.shape = (2,4)
arr
Out[5]: array([[1, 2, 3, 4],
               [5, 6, 7, 8]])

Adición

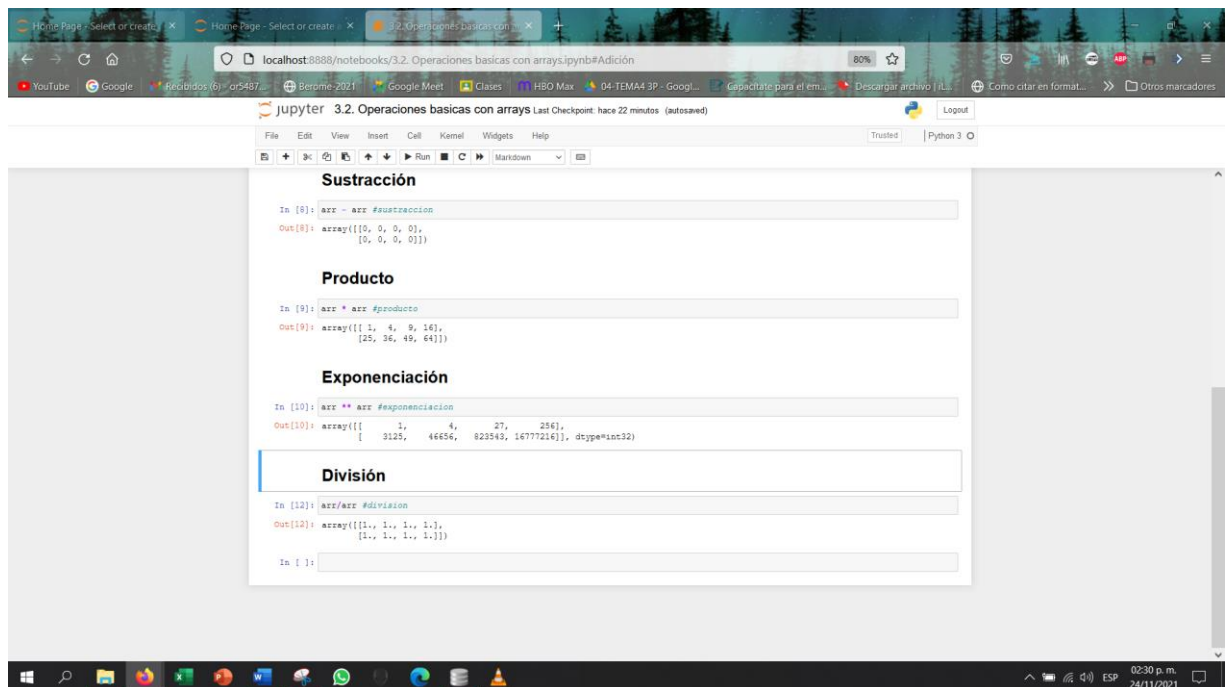
In [6]: arr + arr #adición
Out[6]: array([[ 2,  4,  6,  8],
               [10, 12, 14, 16]])

In [7]: lista + lista
Out[7]: [1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8]

Sustracción

In [8]: arr - arr #sustracción
Out[8]: array([[0, 0, 0, 0],
               [0, 0, 0, 0]])

Producto
```



```
Sustracción

In [8]: arr - arr #sustracción
Out[8]: array([[0, 0, 0, 0],
               [0, 0, 0, 0]])

Producto

In [9]: arr * arr #producto
Out[9]: array([[ 1,  4,  9, 16],
               [25, 36, 49, 64]])

Exponenciación

In [10]: arr ** arr #exponenciación
Out[10]: array([[ 1,  4, 27, 256],
               [3125, 46656, 823543, 16777216]], dtype=int32)

División

In [12]: arr/arr #división
Out[12]: array([[1., 1., 1., 1.],
               [1., 1., 1., 1.]])

In [ ]:
```

3.3 Una vez establecido nuestro producto que serían nuestras variables empezamos a realizar las operaciones para resolver matrices

Productos de arrays

```
In [1]: import numpy as np
a = np.array([1, 2, 3], float)
b = np.array([0.5, 1, 1], float)
np.dot(b, a)

Out[1]: 5.0

In [2]: a.dot(b)

Out[2]: 5.0
```

Productos de matrices

```
In [3]: a = np.array([[5, 2], [4, 0]], float)
b = np.array([[2, 4], [5, 3]], float)
print(a, "\n -----")
print(b, "\n -----")

[[5.  2.]
 [4.  0.]]
-----
[[2.  4.]
 [5.  3.]]
-----

In [4]: np.dot(a, b)

Out[4]: array([[20., 26.],
              [48., 40.]])

In [5]: a @ b

Out[5]: array([[20., 26.],
              [48., 40.]])
```

Determinante de una matriz

```
In [8]: a @ b

Out[8]: array([23., 31., 54.])

In [9]: print(np.matmul(a, b))

[23. 31. 54.]
```

Determinante de una matriz

```
In [10]: a = np.array([[9, 5], [3, 4]])
print(a, "\n -----")
np.linalg.det(a)

[[8 5]
 [3 4]]
-----

Out[10]: 17.0
```

Auto-valores y auto-vectores

```
In [11]: vals, vecs = np.linalg.eig(a)
print(vals, "\n -----")
print(vecs, "\n -----")

[10.35989994  1.64110106]
[[ 0.90440309 -0.61810602]
 [ 0.42667392  0.78609474]]
-----
```

Determinante de una matriz

```
In [9]: print(np.matmul(a, b))

[23. 31. 54.]
```

Determinante de una matriz

```
In [10]: a = np.array([[9, 5], [3, 4]])
print(a, "\n -----")
np.linalg.det(a)

[[8 5]
 [3 4]]
-----

Out[10]: 17.0
```

Auto-valores y auto-vectores

```
In [11]: vals, vecs = np.linalg.eig(a)
print(vals, "\n -----")
print(vecs, "\n -----")

[10.35989994  1.64110106]
[[ 0.90440309 -0.61810602]
 [ 0.42667392  0.78609474]]
-----
```

3.4 En esta ocasión realizaremos funciones universales unitarias que son las que comprenden todas las operaciones que reciben un solo array como argumento.

Función	Descripción
<code>abs</code> , <code>fabs</code>	Calcular el valor absoluto
<code>sqrt</code>	Calcular la raíz cuadrada
<code>square</code>	Calcular el cuadrado
<code>exp</code>	Calcular la exponencial $e^x$
<code>log</code> , <code>log10</code> , <code>log2</code> , <code>log1p</code>	Calcula el Logaritmo natural, en base a 10, 2, y $\log(x + 1)$
<code>isnan</code>	Devuelve un array booleano indicando si cada elemento es NaN (Not a Number)
<code>sin</code> , <code>cos</code> , <code>tan</code>	Devuelve un array con el resultado de aplicar la función seno, coseno, y tangente en los valores
<code>mean</code> , <code>median</code> , <code>std</code> , <code>var</code>	Calcula la media aritmética, la mediana, la desviación estándar y la varianza en un array

The screenshot shows a Jupyter Notebook titled "3.5. Funciones universales binarias" running on a local host. The notebook contains the following code and outputs:

```

In [1]: import numpy as np
        arr = np.array([5, 36, 17, 18, 9])
        arr_2 = np.array([8, 24, 17, 19, 9])

In [2]: np.add(arr, arr_2)
Out[2]: array([13, 60, 34, 37, 18])

In [4]: np.subtract(arr, arr_2)
Out[4]: array([-3, 12,  0, -1,  0])

In [5]: np.multiply(arr, arr_2)
Out[5]: array([ 40, 864, 289, 342,  81])

In [6]: np.divide(arr, arr_2)
Out[6]: array([0.625,  1.5,   1.,  0.94736842,  1.   ])

In [7]: np.array_equal(arr, arr_2)
Out[7]: False

In [8]: np.min(arr, arr_2)
Out[8]: array([ 5, 24, 17, 18,  9])

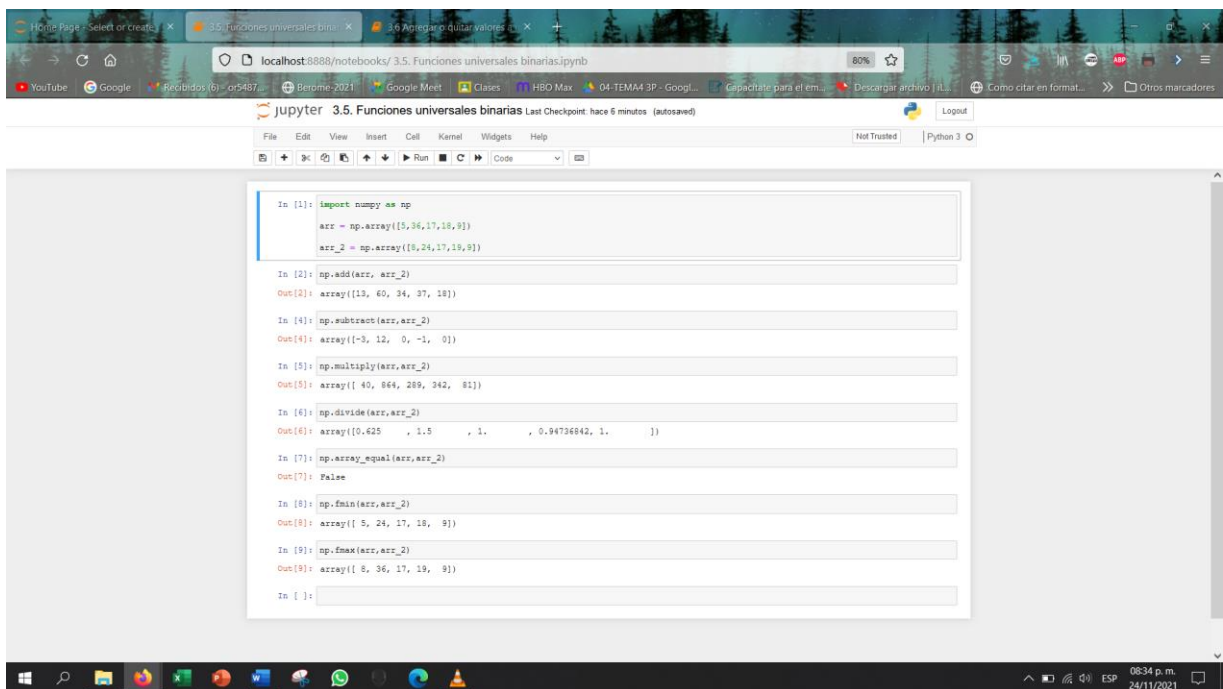
In [9]: np.max(arr, arr_2)
Out[9]: array([ 8, 36, 17, 19,  9])

In [ ]:

```

3.5 Ahora realizaremos las funciones universales binaria que a diferencia de las unitarias toman dos arrays y devuelven uno o más arrays

Función	Descripción
<code>add</code>	Suma entre elementos correspondientes entre arrays
<code>subtract</code>	Resta entre elementos de arrays
<code>multiply</code>	Multiplica arrays
<code>divide</code> , <code>floor_divide</code>	Divide o división truncada
<code>array_equal</code>	Devuelve <code>True</code> si los elementos del array son iguales entre si
<code>power</code>	Eleva cada elemento del primer array a la potencia indicada en el segundo array
<code>fmin</code>	Devuelve el mínimo entre cada elemento. <code>fmin</code> ignora los <code>NaN</code>
<code>fmax</code>	Devuelve el máximo entre cada elemento. <code>fmax</code> ignora los <code>NaN</code>



The screenshot shows a Jupyter Notebook interface with a browser window at the top displaying the URL `localhost:8888/notebooks/3.5. Funciones universales binarias.ipynb`. The notebook title is "3.5. Funciones universales binarias". The code cell contains the following Python code:

```
In [1]: import numpy as np
arr = np.array([5, 36, 17, 18, 9])
arr_2 = np.array([8, 24, 17, 19, 9])

In [2]: np.add(arr, arr_2)
Out[2]: array([13, 60, 34, 37, 18])

In [4]: np.subtract(arr, arr_2)
Out[4]: array([-3, 12,  0, -1,  0])

In [5]: np.multiply(arr, arr_2)
Out[5]: array([ 40, 864, 289, 342,  81])

In [6]: np.divide(arr, arr_2)
Out[6]: array([0.625,  1.5,  1.,  0.94736842, 1.])

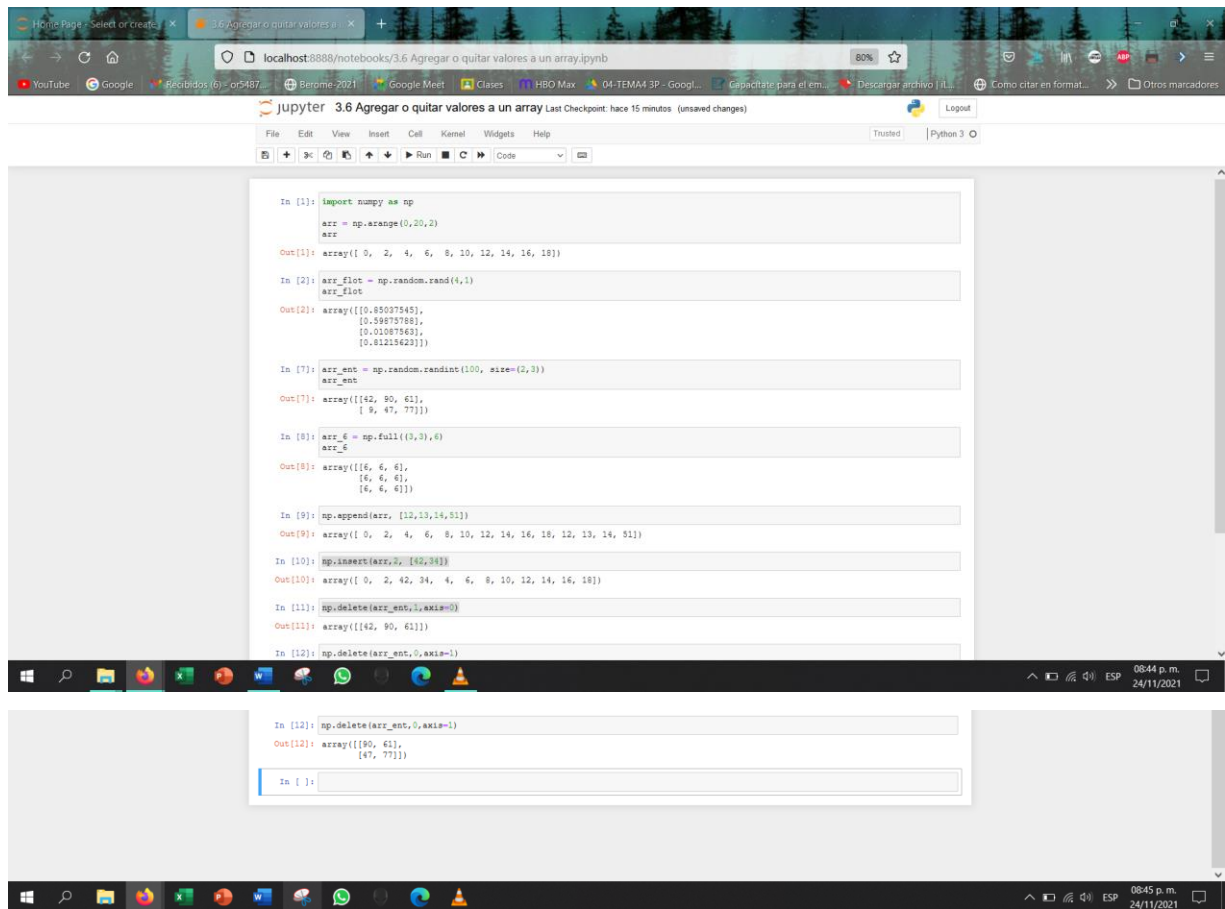
In [7]: np.array_equal(arr, arr_2)
Out[7]: False

In [8]: np.fmin(arr, arr_2)
Out[8]: array([ 5, 24, 17, 18,  9])

In [9]: np.fmax(arr, arr_2)
Out[9]: array([ 8, 36, 17, 19,  9])

In [ ]:
```

3.6 Aquí vamos a ver como agregamos y quitamos valores a un array sencillamente con los comandos `np`



The screenshot displays a Jupyter Notebook titled "3.6 Agregar o quitar valores a un array.ipynb" running on a local host. The notebook contains 12 code cells demonstrating various NumPy array operations. The first cell imports NumPy and creates an array. Subsequent cells show random number generation, array creation with specific values, and operations like appending, inserting, and deleting elements from arrays. The outputs of each cell are displayed below the code, showing the resulting arrays and their shapes. The interface includes a top menu bar with options like File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The bottom status bar shows the system time as 08:44 p.m. on 24/11/2021.

```
In [1]: import numpy as np
arr = np.arange(0,20,2)
arr
Out[1]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])

In [2]: arr_float = np.random.rand(4,1)
arr_float
Out[2]: array([[0.85037545],
               [0.59875788],
               [0.01087683],
               [0.82215623]])

In [7]: arr_int = np.random.randint(100, size=(2,3))
arr_int
Out[7]: array([[42, 90, 61],
               [ 9, 47, 77]])

In [8]: arr_6 = np.full((3,3),6)
arr_6
Out[8]: array([[6, 6, 6],
               [6, 6, 6],
               [6, 6, 6]])

In [9]: np.append(arr, [12,13,14,51])
Out[9]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 12, 13, 14, 51])

In [10]: np.insert(arr, 2, [42,34])
Out[10]: array([ 0,  2, 42, 34,  4,  6,  8, 10, 12, 14, 16, 18])

In [11]: np.delete(arr_int, 1, axis=0)
Out[11]: array([[42, 90, 61]])

In [12]: np.delete(arr_int, 0, axis=1)
Out[12]: array([[90, 61],
               [47, 77]])

In [ ]:
```

3.7 Ahora realizaremos transformaciones con los comandos de np como as type, sort , reshape y fattlen entre otros



Home Page - Select or create... x 3.7 Transformaciones - Jupyter x Google x +

localhost:8888/notebooks/%203.7%20Transformaciones.ipynb 80%

YouTube Google Ficheros (b) or5487 Berome-2021 Google Meet Clases HBO Max 04-TEMA4 3P - Googl... Capacitate para el em... Descargar archivo | IL... Como citar en format... Otros marcadores

Jupyter 3.7 Transformaciones Last checkpoint: hace una hora (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

### As type

```
In [2]: import numpy as np
arr_ent = np.random.randint(100, size=(3,4))
arr_ent
```

```
Out[2]: array([[ 9, 35, 96, 17],
               [63, 30, 58, 40],
               [66, 19, 34, 73]])
```

```
In [3]: arr_ent.astype(float)
Out[3]: array([[ 9., 35., 96., 17.],
               [63., 30., 58., 40.],
               [66., 19., 34., 73.]])
```

### Sort

```
In [4]: arr_ent
Out[4]: array([[ 9, 35, 96, 17],
               [63, 30, 58, 40],
               [66, 19, 34, 73]])
```

```
In [5]: arr_ent.sort()
arr_ent
```

```
Out[5]: array([[ 9, 17, 35, 96],
               [30, 40, 58, 63],
               [19, 34, 66, 73]])
```

```
In [6]: arr_ent.sort(axis=0)
arr_ent
```

```
Out[6]: array([[ 9, 17, 35, 63],
               [19, 34, 58, 73],
               [30, 40, 66, 96]])
```

Windows Search File Explorer Firefox Microsoft Word WhatsApp Telegram Docker Desktop

9:39 p.m. 24/11/2021

Home Page - Select or create... x 3.7 Transformaciones - Jupyter x Google x +

localhost:8888/notebooks/%203.7%20Transformaciones.ipynb 80%

YouTube Google Ficheros (b) or5487 Berome-2021 Google Meet Clases HBO Max 04-TEMA4 3P - Googl... Capacitate para el em... Descargar archivo | IL... Como citar en format... Otros marcadores

Jupyter 3.7 Transformaciones Last checkpoint: hace una hora (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
[19, 34, 66, 73]]
```

```
In [6]: arr_ent.sort(axis=0)
arr_ent
```

```
Out[6]: array([[ 9, 17, 35, 63],
               [19, 34, 58, 73],
               [30, 40, 66, 96]])
```

### Reshape

```
In [9]: arr_ent
Out[9]: array([[ 9, 17, 35],
               [63, 19, 34],
               [58, 73, 30],
               [40, 66, 96]])
```

```
In [13]: arr_ent = arr_ent.reshape(4,3)
arr_ent
```

```
Out[13]: array([[ 9, 17, 35],
               [63, 19, 34],
               [58, 73, 30],
               [40, 66, 96]])
```

### Flatten

```
In [12]: plano_arr = arr_ent.flatten()
print(arr_ent)
print(plano_arr)
```

```
[[ 9 17]
 [35 63]
 [19 34]
 [58 73]
 [30 40]
 [66 96]]
[ 9 17 35 63 19 34 58 73 30 40 66 96]
```

Windows Search File Explorer Firefox Microsoft Word WhatsApp Telegram Docker Desktop

9:41 p.m. 24/11/2021

Home Page - Select or create x 2.7 Transformaciones - Jupyter x Google x

localhost:8888/notebooks/%203.7%20Transformaciones.ipynb 80% ☆

YouTube Google Pájaros (6) - cr5487... Berome 2021 Google Meet Clases HBO Max 04-TEMA4 3P - Googl... Gacacite para el em... Descargar archivo | L... Como citar en format... Otros marcadores

Jupyter 3.7 Transformaciones Last checkpoint: hace una hora (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
[35 43]
[18 34]
[58 73]
[30 40]
[66 96]]
[ 9 17 35 63 19 34 58 73 30 40 66 96]
```

### To list

```
In [15]: lista_arr = plano_arr.tolist()
lista_arr
type(lista_arr)

Out[15]: list
```

### Separar y juntar arrays

```
In [16]: arrays= np.split(arr_ent, 4)
arrays

Out[16]: [array([[ 9, 17, 35]]),
array([[63, 19, 34]]),
array([[58, 73, 30]]),
array([[40, 66, 96]])]
```

```
In [17]: np.concatenate((arrays[1],arrays[0]),axis=0)

Out[17]: array([[63, 19, 34],
[ 9, 17, 35]])
```

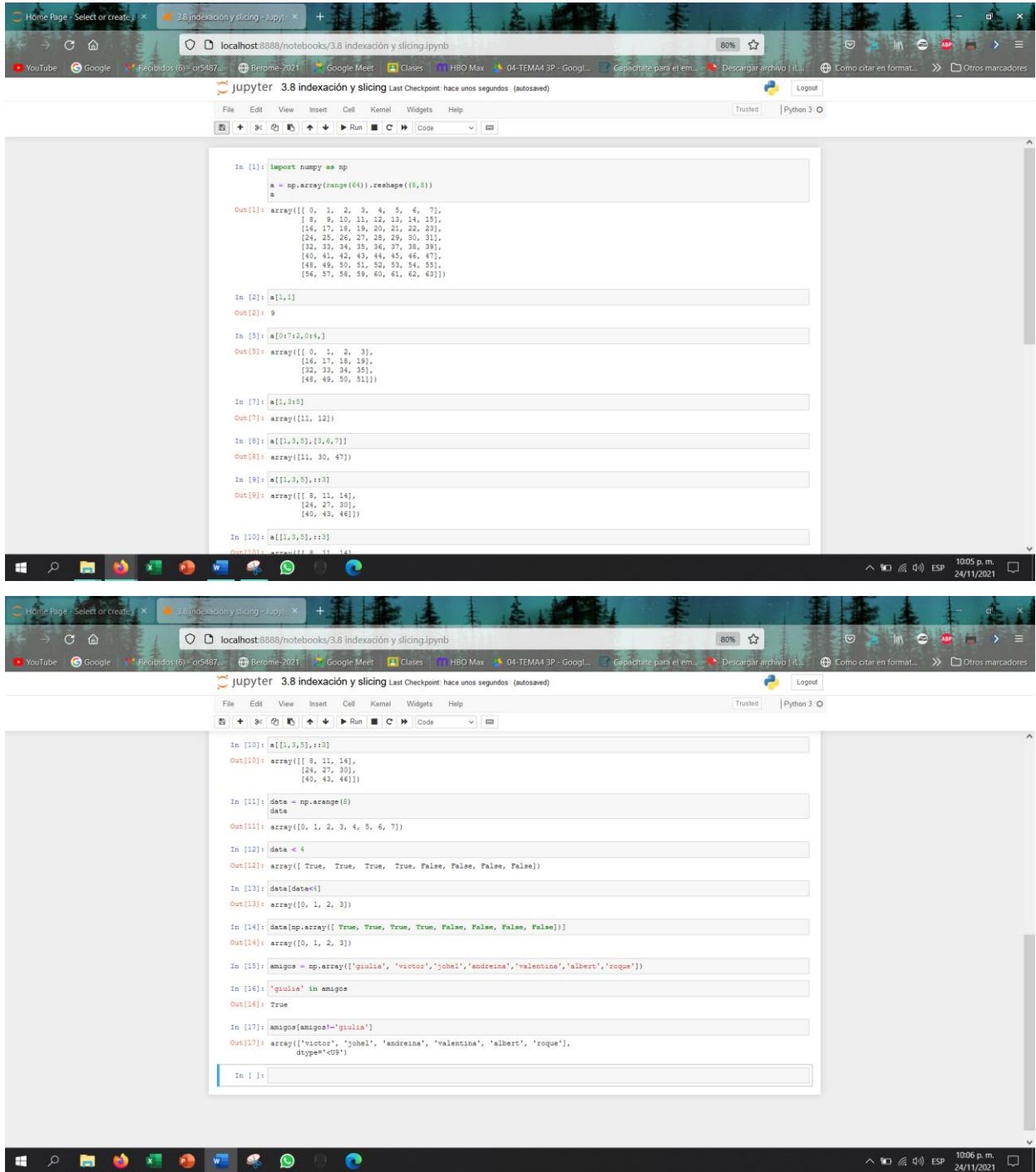
### La matriz transpuesta

```
In [18]: arr_ent

Out[18]: array([[ 9, 17, 35],
[63, 19, 34],
[58, 73, 30],
[40, 66, 96]])
```

Windows taskbar: 09:41 p.m. 24/11/2021

### 3.8 Por ultimo vamos a realizar la indexación y slicing (filtrado)



The image displays two screenshots of a Jupyter Notebook interface, showing the process of creating and manipulating NumPy arrays for indexing and slicing.

**Top Screenshot:** The notebook is titled "3.8 indexación y slicing". It shows the following code and outputs:

```
In [1]: import numpy as np
a = np.array(range(64)).reshape((8,8))
a
```

Out[1]: array([[ 0, 1, 2, 3, 4, 5, 6, 7],
 [ 8, 9, 10, 11, 12, 13, 14, 15],
 [16, 17, 18, 19, 20, 21, 22, 23],
 [24, 25, 26, 27, 28, 29, 30, 31],
 [32, 33, 34, 35, 36, 37, 38, 39],
 [40, 41, 42, 43, 44, 45, 46, 47],
 [48, 49, 50, 51, 52, 53, 54, 55],
 [56, 57, 58, 59, 60, 61, 62, 63]])

```
In [2]: a[1,1]
Out[2]: 9
```

```
In [3]: a[0:7:2,0:4]
Out[3]: array([[ 0,  1,  2,  3],
 [16, 17, 18, 19],
 [32, 33, 34, 35],
 [48, 49, 50, 51]])
```

```
In [7]: a[1,3:5]
Out[7]: array([11, 12])
```

```
In [8]: a[[1,3,5],[3,6,7]]
Out[8]: array([11, 30, 47])
```

```
In [9]: a[[1,3,5],:3]
Out[9]: array([[ 8, 11, 14],
 [24, 27, 30],
 [40, 43, 46]])
```

```
In [10]: a[[1,3,5],:3]
Out[10]: array([[ 8, 11, 14],
 [24, 27, 30],
 [40, 43, 46]])
```

**Bottom Screenshot:** The notebook continues with the following code and outputs:

```
In [10]: a[[1,3,5],:3]
Out[10]: array([[ 8, 11, 14],
 [24, 27, 30],
 [40, 43, 46]])
```

```
In [11]: data = np.arange(8)
data
Out[11]: array([0, 1, 2, 3, 4, 5, 6, 7])
```

```
In [12]: data < 4
Out[12]: array([ True,  True,  True,  True, False, False, False, False])
```

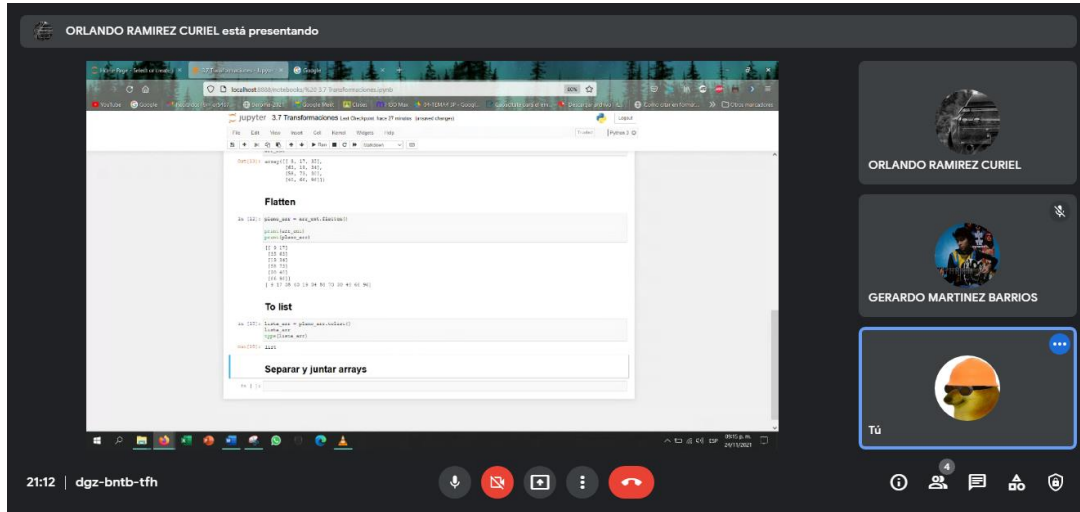
```
In [13]: data[data<4]
Out[13]: array([0, 1, 2, 3])
```

```
In [14]: data[np.array([ True,  True,  True,  True, False, False, False, False])]
Out[14]: array([0, 1, 2, 3])
```

```
In [15]: amigos = np.array(['gisela', 'victor', 'johel', 'andreina', 'valentina', 'albert', 'roque'])
In [16]: 'gisela' in amigos
Out[16]: True
In [17]: amigos[amigos=='gisela']
Out[17]: array(['victor', 'johel', 'andreina', 'valentina', 'albert', 'roque'],
      dtype='<O>')
```

```
In [ ]:
```

## Evidencia de equipo.



## Conclusión

El paquete es conocido por una estructura de datos muy útil llamado arreglo de NumPy. NumPy también permite a los desarrolladores de Python realizar en forma rápida una amplia variedad de cálculo numéricos.

Recordando la competencia: conocer los conceptos fundamentales de los sistemas basados en conocimientos, así como el estado del arte de las áreas relacionadas al mismo lo cual consideramos comprendida y la práctica de numpy la consideramos completada al 100%.

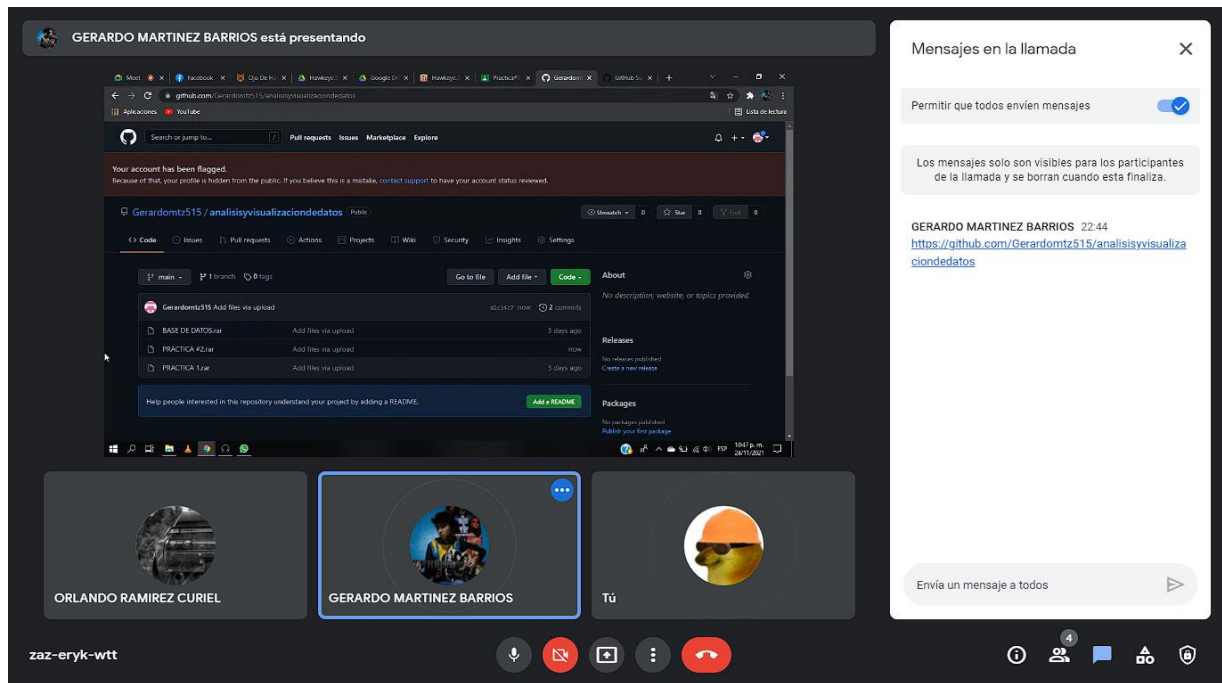
Github:

Juan: <https://github.com/SegoviaJuan/Analisis-y-Visualizacion-de-Datos->

Orlando: <https://github.com/OelandoRamirezCuriel-1/Analisis-y-Visualizacion-de-datos.git>

Gerardo: <https://github.com/Gerardomt515/analisisyvisualizaciondedatos>

En caso de fallar el link de mi compañero gerardo tiene un fallo en su cuenta la cual github la puso en privado



## Bibliografías:

Academy, P. L. (2020). *Python para Principiantes: 2 Libros en 1: Programación de Python para principiantes + Libro de trabajo de Python*. PROGRAMMING LANGUAGES ACADEMY.

Bailey, J., Djermanovic, D., & Dominguez, O. A. (2021). *Saving Data on Android (Second Edition): Learn Jetpack DataStore, Room, Firebase & SQLite with Kotlin*. Razeware LLC.

Siahaan, V., & Sianipar, R. H. (2019). *Learn SQLite with Python: Building Database-Driven Desktop Projects*. Independently published.