Creado por:

Isabel Maniega

Matrices

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
matriz = np.array([
    [10, 20],
    [30, 40]
])
matriz
```

Out[2]:
```
array([[10, 20],
       [30, 40]])
```

In [3]:
```
matriz[0][0] # fila 0 columna 0
```

Out[3]:
```
10
```

In [5]:
```
matriz[0][1] # fila 0 columna 1
```

Out[5]:
```
20
```

SET

In [6]:
```
listado = [10, 10, 20, 30, 40, 40, 50]
listado
```

Out[6]:
```
[10, 10, 20, 30, 40, 40, 50]
```

In [7]:
```
df = pd.DataFrame({"listado": listado})
df
```

Out[8]:

	listado
0	10
1	10
2	20
3	30
4	40
5	40
6	50

In [10]:
```
df.listado.value_counts()
```

Out[10]:
```
40    2
10    2
20    1
30    1
50    1
Name: listado, dtype: int64
```

In [11]:
```
set(listado)
```

Out[11]:
```
{10, 20, 30, 40, 50}
```

In [12]:
```
listado2 = {"casa", "casa"}
set(listado2)
```

Out[12]:
```
{'casa'}
```

In [13]:
```
listado3 = {"Casa", "casa"}
set(listado3)
```

Out[13]:
```
{'Casa', 'casa'}
```

In [14]:
```
listado4 = list((10, 10, 10, 10, 10, 10))
listado4
```

Out[14]:
```
[10, 10, 10, 10, 10, 10, 10]
```

In [15]:
```
set(listado4)
```

Out[15]:
```
{10}
```

In [16]:
```
type(set(listado4))
```

Out[16]:
```
set
```

In [18]:
```
listado5 = list(set(listado4))
```

In [19]:
```
listado5
```

Out[19]:
```
[10]
```

Lectura de un archivo (.csv) con Pandas

Iris Dataset

In [20]:
```
df = pd.read_csv("iris.csv")
df
```

Out[20]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [21]:
```
df.head()
```

Out[21]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [23]:
```
df.tail()
```

Out[23]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

In [24]:
```
df.describe()
```

Out[24]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [25]:
```
len(df), df.shape
```

Out[25]:
```
(150, (150, 5))
```

In [26]:
```
type(df)
```

Out[26]:
```
pandas.core.frame.DataFrame
```

In [28]:
```
df.species.value_counts()
```

Out[28]:
```
virginica    50
versicolor   50
setosa       50
Name: species, dtype: int64
```

Otro archivo de Iris que podemos encontrarnos

In [29]:
```
df_2 = pd.read_csv("iris.csv")
df_2
```

Out[29]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

Otra forma de ver el Iris dataset

In [30]:
```
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
```

In [31]:
```
col_names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Class']
iris = pd.read_csv(csv_url, names = col_names)
iris
```

Out[31]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

Titanic dataset

In [5]:
```
df_titanic = pd.read_csv("./ejercicios/train.csv")
df_titanic
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Números primos

es aquel que solo es divisible por sí mismo y por la unidad

entonces el resto debería salir != 0

(en todos esos casos que se van a indicar a continuación)

EJEMPLO:

Para el 5 sólo me interesa:

5%2, 5%3, 5%4

el 1

In [7]:
```
# el 1 no se tiene en cuenta
```

el 2

In [8]:
```
2/2, 2/1 # Si
```

Out[8]:
```
(1.0, 2.0)
```

In [9]:
```
2%2, 2%1 # Si
```

Out[9]:
```
(0, 0)
```

el 3

In [10]:
```
3/3, 3/2, 3/1 # Si
```

Out[10]:
```
(1.0, 1.5, 3.0)
```

In [11]:
```
3%3, 3%2, 3%1 # SI (dividir por sí mismo y la unidad)
```

Out[11]:
```
(0, 1, 0)
```

In [12]:
```
4/4, 4/3, 4/2, 4/1 # No la es
```

Out[12]:
```
(1.0, 1.3333333333333333, 2.0, 4.0)
```

In [13]:
```
4%4, 4%3, 4%2, 4%1 # NO
```

Out[13]:
```
(0, 1, 0, 0)
```

el 5

In [14]:
```
5/5, 5/4, 5/3, 5/2, 5/1 # Si
```

Out[14]:
```
(1.0, 1.25, 1.6666666666666667, 2.5, 5.0)
```

In [15]:
```
5%5, 5%4, 5%3, 5%2, 5%1
```

Out[15]:
```
(0, 1, 2, 1, 0)
```

el 6

In [16]:
```
6/6, 6/5, 6/4, 6/3, 6/2, 6/1 # NO
```

Out[16]:
```
(1.0, 1.2, 1.5, 2.0, 3.0, 6.0)
```

In [17]:
```
6%6, 6%5, 6%4, 6%3, 6%2, 6%1 # NO
```

Out[17]:
```
(0, 1, 2, 0, 0, 0)
```

el 7

In [18]:
```
7/7, 7/6, 7/5, 7/4, 7/3, 7/2, 7/1 # Si
```

Out[18]:
```
(1.0, 1.1666666666666667, 1.4, 1.75, 2.3333333333333335, 3.5, 7.0)
```

In [20]:
```
7%7, 7%6, 7%5, 7%4, 7%3, 7%2, 7%1 # Si
```

Out[20]:
```
(0, 1, 2, 3, 1, 1, 0)
```

Cosas importantes a tener en cuenta

¿cómo podríamos ir viendo 1 a 1?

- me creo un listado de los restos (todos menos el 1 y el propio número)

NOTA: el 2 le aprendizamos por defecto

- Buscar si hay por lo menos un 0 en ese listado de restos

(código justo debajo)

EJEMPLO CONCRETO PARA EL **NUMERO 7**

In [21]:
```
listado_restos_7 = [7%6, 7%5, 7%4, 7%3, 7%2]
listado_restos_7
```

Out[21]:
```
[1, 2, 3, 1, 1]
```

In [22]:
```
if 0 not in listado_restos_7:
    print("el numero es primo")

#1 numero 7 es primo
```

Recordar el uso del append

In [23]:
```
listado = []
listado
```

Out[23]:
```
[]
```

In [24]:
```
listado.append(2)
listado
```

Out[24]:
```
[2]
```

¿Cómo obtener los POSIBLES números del 3 al 200 (200 no incluido)?

In [25]:
```
listado_completo = np.arange(3, 200, 1)
listado_completo
```

Out[25]:
```
array([  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,  64,  65,  66,  67,
        68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,  79,  80,
        81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,  92,  93,
        94,  95,  96,  97,  98,  99, 100, 101, 102, 103, 104, 105, 106,
       107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119,
       120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132,
       133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145,
       146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158,
       159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171,
       172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184,
       185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197,
       198, 199])
```

Creado por:

Isabel Maniega