Creado por:

Isabel Maniega

Importar librerías necesarias

In [39]:
```
# Instalación: pip install numpy
import numpy as np
import pandas as pd
```

El presente contenido es muy similar al software de pago MATLAB o incluso OCTAVE

Array de 5 x 5

In [42]:
```
a = np.array([
    [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]
])
a
```

Out[42]:
```
array([[ 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]])
```

Imprimir desde la 3ª columna hasta el final

In [3]:
```
a # mostrar la información de la matriz
```

Out[3]:
```
array([[ 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]])
```

In [4]:
```
# ojo, empecamos contando 0...(0-1-2) hasta la columna 2 (la tercera)
# : antes del igual indica todas las filas
# todas las filas, las columnas de 0 hasta 2 (2 no incluida)
a[:, :2]
```

Out[4]:
```
array([[ 1,  2],
       [ 6,  7],
       [11, 12],
       [16, 17],
       [21, 22]])
```

In [5]:
```
# todas las columnas de las 2 primeras filas
a[:2]
```

Out[5]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [6]:
```
a[:2, :]
```

Out[6]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [7]:
```
a[:, 1:2]
```

Out[7]:
```
array([[ 2],
       [ 7],
       [12],
       [17],
       [22]])
```

In [8]:
```
# NOTA: esta parte será importante para el tema de visualización de los datos en dataframe,
# ver el tema de df.loc o df.iloc
```

Type...

In [10]:
```
type(a[:,2:])
```

Out[10]:
```
numpy.ndarray
```

Imprimo desde la primera columna hasta la 2ª (incluida)

In [11]:
```
a
```

Out[11]:
```
array([[ 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]])
```

In [12]:
```
# Opción 1

a[:, :2]
```

Out[12]:
```
array([[ 1,  2],
       [ 6,  7],
       [11, 12],
       [16, 17],
       [21, 22]])
```

In [13]:
```
# Opción 2
a[:, 0:2]
```

Out[13]:
```
array([[ 1,  2],
       [ 6,  7],
       [11, 12],
       [16, 17],
       [21, 22]])
```

Imprimo las pares

In [14]:
```
a
```

Out[14]:
```
array([[ 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]])
```

In [15]:
```
# ":" antes de la coma equivale a todas las filas
# inicio:final:incremento (si añades un segundo ":" es poner el incremento)
# en el final si no ponemos nada es el final
```

In [16]:
```
a[:, 1::2]
```

Out[16]:
```
array([[ 2,  4],
       [ 7,  9],
       [12, 14],
       [17, 19],
       [22, 24]])
```

In [17]:
```
a[:, 1::3]
```

Out[17]:
```
array([[ 2,  5],
       [ 7, 10],
       [12, 15],
       [17, 20],
       [22, 25]])
```

Imprimir las impares

In [18]:
```
a
```

Out[18]:
```
array([[ 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]])
```

In [19]:
```
a[:, 0::2]
```

Out[19]:
```
array([[ 1,  3,  5],
       [ 6,  8, 10],
       [11, 13, 15],
       [16, 18, 20],
       [21, 23, 25]])
```

In [20]:
```
a[:, 0:2:2]
```

Out[20]:
```
array([[ 1],
       [ 6],
       [11],
       [16],
       [21]])
```

In [21]:
```
a[:, 0:3:2]
```

Out[21]:
```
array([[ 1,  3],
       [ 6,  8],
       [11, 13],
       [16, 18],
       [21, 23]])
```

Secuencia de Fibonacci

Cada numero es la suma de los 2 anteriores:

0-1-1-2-3-5-8-13-21-34...

obviamente tomando como referencia 0 y 1

0 + 1 es 1

1 + 1 es 2

1 + 2 es 3

así sucesivamente

Usos de Fibonacci:

- Mercados de valores bursátiles
- Y otros muchos usos, se recomienda, buscar mas información.

Funciones, con "2 variables de entrada"

In [26]:
```
def funcion_suma(x, y):
    z = x + y
    return z
```

In [27]:
```
# es similar a realizarlo:
def funcion_suma_2(x, y):
    return x + y
```

In [28]:
```
funcion_suma(5, 4)
```

Out[28]:
```
9
```

In [29]:
```
funcion_suma_2(5, 4)
```

Out[29]:
```
9
```

In [30]:
```
def funcion_multiple(x, y):
    s = x + y
    m = x * y
    return s, m
```

In [33]:
```
s, m = funcion_multiple(5, 4)
print('Valor de s:', s, ' y valor de m: ', m)
```

```
Valor de s: 9  y valor de m:  20
```

Función Lambda (repaso)

In [34]:
```
def funcion_suma_1(x):
    return x +10

# Llamada de la función
funcion_suma_1(15)
```

Out[34]:
```
25
```

In [36]:
```
# con lambda:
(lambda x: x+10)(5)
#       5: 5+10
# --> retorno 15
```

Out[36]:
```
15
```

In [37]:
```
# lambda con dos variables
(lambda x, y: x + y)(5, 4)
```

Out[37]:
```
9
```

In [38]:
```
# lambda con 3 variables
(lambda x, y, z: x + y + z)(5, 4, 1)
```

Out[38]:
```
10
```

DATAFRAMES -Gestión-

Crear un dataframe

In [42]:
```
df = pd.DataFrame({"x": [10, 20, 30, 40, 50], "y": [1, 1, 0, 1, 0]})
df
```

Out[42]:

	x	y
0	10	1
1	20	1
2	30	0
3	40	1
4	50	0

In [43]:
```
# Creamos un df a partir del df inicial
# seleccionamos los valores de la columnma y que tengan el valor de 1
# y asignamos un nuevo dataframe de nombre df_uno
df_uno = df[df["y"] == 1]
df_uno
```

Out[43]:

	x	y
0	10	1
1	20	1
3	40	1

In [44]:
```
# otra opción:

df_uno_1 = df[df.y == 1]
df_uno_1
```

Out[44]:

	x	y
0	10	1
1	20	1
3	40	1

In [45]:
```
df_cero = df[df['y'] == 0]
df_cero
```

Out[45]:

	x	y
2	30	0
4	50	0

In [46]:
```
maximo = df.max()
maximo
```

Out[46]:
```
x    50
y     1
dtype: int64
```

In [47]:
```
minimo = df.min()
minimo
```

Out[47]:
```
x    10
y     0
dtype: int64
```

In [51]:
```
# Uno y otro
df_comparacion = df[(df['x'] < 50) & (df["x"] > 10)]
df_comparacion
```

Out[51]:

	x	y
1	20	1
2	30	0
3	40	1

In [57]:
```
# Uno u otro
df_comparacion = df[(df['x'] > 40) | (df["x"] > 20)]
df_comparacion
```

Out[57]:

	x	y
2	30	0
3	40	1
4	50	0

Creado por:

Isabel Maniega