Autores: María Inmaculada Rodríguez García, María Gema Carrasco García, Javier González Enrique, Juan Jesús Ruiz Aguilar, Ignacio J. Turias Domínguez. Universidad de Cádiz

2.2. *Dataframes* con Panda en Python y registros en MATLAB

Un **Dataframe** es una matriz multidimensional formada por series en Pandas

Cada columna es una serie, que tendrá su índice y su nombre.

- Podemos etiquetar los índices.

Documentación: Documentación DataFrames

- **En MATLAB** el equivalente a *dataframes* son tablas de más de dos columnas, como una matriz con diferentes tipos de datos. En algunos casos, el uso de registros también puede permitir realizar una equivalencia con Python.

```
% Crear una tabla ejemplo:
 T = table(categorical({'M';'F';'M'}),[45;32;34],logical([1;0;0]),...
   'VariableNames',{'Gender','Age','Vote'},...
   'RowNames',{'NY';'CA';'MA'})
```

```
T=3×3 table
               Gender    Age    Vote
               ______    ___    _____

        NY       M       45     true
        CA       F       32     false
        MA       M       34     false
```

Documentación Command Window: help table

Ejercicio 69_01. Definición de *Dataframe* en Python

```
# Importamos las librerías necesarias
import numpy as np
import pandas as pd
df = pd.DataFrame({"ex1": [10, 3, 6, 4], "ex2": [10, 5, 8, 5], "ex3": [8, 6, 8,2]},
index = ["Alex", "Pedro", "Paula", "Miguel"])
print(df)
```

```
            ex1    ex2    ex3
    Alex     10      10      8
    Pedro     3       5      6
    Paula     6       8      8
    Miguel    4       5      2
```

Ejercicio 69_02. Definición de tabla en MATLAB

```
% Creamos una Tabla en MATLAB
datos = [10, 3, 6; 10, 5, 8; 8, 6, 8; 5, 4, 2];
nombres_filas = {'Alex', 'Pedro', 'Paula', 'Miguel'};
nombres_columnas = {'ex1', 'ex2', 'ex3'};
df = array2table(datos, 'RowNames', nombres_filas, 'VariableNames',
nombres_columnas);
```

```
% Mostramos la Tabla
disp('Tabla matriz:');
disp(df);

Tabla matriz:
              ex1    ex2    ex3
              ___    ___    ___

    Alex      10      3      6
    Pedro     10      5      8
    Paula      8      6      8
    Miguel     5      4      2
```

Ejercicio 70_01. ***Definición partiendo de matriz de numpy***

```
# Matriz con numpy
matriz = np.array([[1, 2, 3], [4, 5, 6]])
# Nombramos columnas y damos etiquetas a filas
filas = ['2018', '2019']
col = ['valor1', 'valor2', 'valor3']
# Creamos dataframe
df2 = pd.DataFrame(matriz, index=filas, columns=col)
print("Imprimimos el dataframe: ")
print(df2)

Imprimimos el dataframe:
            valor1    valor2    valor3

    2018      1         2         3
    2019      4         5         6
```

Ejercicio 70_02. ***Definición partiendo de matriz***

```
matriz = [1, 2, 3; 4, 5, 6];
% Nombramos columnas y damos etiquetas a filas
filas = {'2018', '2019'};
columnas = {'valor1', 'valor2', 'valor3'};
% Creamos una tabla
df2 = array2table(matriz, 'RowNames', filas, 'VariableNames', columnas);
% Mostramos el DataFrame
disp('Matriz tabla 2:');
disp(df2);

Matriz tabla 2:
            valor1    valor2    valor3
            ______    ______    ______

    2018      1         2         3
    2019      4         5         6
```

Autores: María Inmaculada Rodríguez García, María Gema Carrasco García, Javier González Enrique, Juan Jesús Ruiz Aguilar, Ignacio J. Turias Domínguez. Universidad de Cádiz

Ejercicio 71_01. Acceso a los elementos del *dataframe* en Python

```
print("Obtenemos columna de df: ")
print(df.ex1)
print("Obtenemos fila 3 de la columna ex1: ")
print(df.ex1[2])
print("Obtenemos fila Paula de la columna ex3: ")
print(df.ex3["Paula"])
```

```
Obtenemos columna de df:
Alex     10
Pedro     3
Paula     6
Miguel    4
Name: ex1, dtype: int64
Obtenemos fila 3 de la columna ex1:
6
Obtenemos fila Paula de la columna ex3:
8
```

Ejercicio 71_02. Acceso a los elementos en MATLAB

```
% Acceso a los elementos
disp('Obtenemos columna de df:');
disp(df.ex1');
disp('Obtenemos fila 3 de la columna ex1:');
disp(df.ex1(3));
disp('Obtenemos fila Paula de la columna ex3:');
disp(df.ex3('Paula'));
```

```
Obtenemos columna de df:
    10    10     8     5
Obtenemos fila 3 de la columna ex1:
     8
Obtenemos fila Paula de la columna ex3:
     8
```

Ejercicio 72_01. Dataframes similares a diccionarios

```
import numpy as np
import pandas as pd

df = pd.DataFrame({
#Valores reales
"A": [1.0, 3.5, 3.4, 1.4],
# Fechas
"B": pd.date_range("20221129"),
# 4días
"C": pd.Series([1, 2, 3, np.NaN]),
# Valores enteros
"D": np.array([3, 4, 5, 6]),
# Valores categóricos
"E": pd.Categorical(["test", "train", "test", "train"]),
```

```
 })

# Añadimos las etiquetas de filas
filas = [["Lunes", "Martes", "Miércoles", "Jueves"]]
df.index = filas;
print(df)
print("Mostramos el tipo de dato de cada columna")
print(df.dtypes)
```

```
                A          B          C     D    E      F
    Lunes       1.0    29-Nov-2022    1.0    3    2    test
    Martes      3.5    30-Nov-2022    2.0    4    5    train
    Miércoles   3.4    01-Dec-2022    3.0    5    8    test
    Jueves      1.4    02-Dec-2022    NaN    6    1    train

Mostramos el tipo de dato de cada columna:
A      float64
B      datetime64[ns]
C      float64
D      int32
E      category
dtype: object
```

```
dfVentas = pd.DataFrame({
'meses': [1, 2, 3, 4],
'años': [2018, 2019, 2020, 2021],
'ventas': [125, 130, 232, 134]
})
print("Imprimimos el dataframe original dfVentas: ")
print(dfVentas)
dfVentas2 = dfVentas.set_index("años")
print("\n Mostramos el nuevo dataframe dfVentas2: ") # Usamos \n para salto de línea
print(dfVentas2)
```

```
Imprimimos el dataframe original dfVentas:
     meses    años    ventas
  0    1      2018     125
  1    2      2019     130
  2    3      2020     232
  3    4      2021     134

Mostramos el nuevo dataframe dfVentas2:
    años   meses     ventas
    2018      1        125
    2019      2        130
    2020      3        232
    2021      4        134
```

```
# Importante: al fijar el índice de esta forma, debemos observar que la columna
elimina el conjunto de características. Esto podemos controlarlo con "drop".
dfVentas2 = dfVentas.set_index("años", drop = False)
print(dfVentas2)
```

```
Mostramos el nuevo dataframe dfVentas2:

    años    meses    ventas
    ____________________
    2018      1        125
    2019      2        130
    2020      3        232
    2021      4        134

# Mostramos el dataframe original
print("DataFrame dfVentas: ")
print(dfVentas)
# Cambiamos el índice con set_index haciendo esta modificiación en dfVentas
dfVentas.set_index("años", inplace = True, drop = False)
print("\n DataFrame dfVentas tras modificar con set_index: ") # usamos \n para salto
de línea
print(dfVentas)
```

Ejercicio 72_02. Trabajar con registros en MATLAB

```
datos = {
    [1.0; 3.5; 3.4; 1.4],                                % Valores reales
    [datetime('2022-11-29'); datetime('2022-11-30'); …
     datetime('2022-12-01');datetime('2022-12-02')], % Fechas
    [1; 2; 3; NaN],                                      % 4 días
    [3; 4; 5; 6],
    [2; 5; 8; 1],                                         % Valores enteros
     categorical({'test', 'train', 'test', 'train'}') % Valores categóricos
 };
nombres_filas = {'Lunes', 'Martes', 'Miércoles', 'Jueves'};
nombres_columnas = {'A', 'B', 'C', 'D', 'E', 'F'};
df = table(datos{:}, 'RowNames', nombres_filas, 'VariableNames', nombres_columnas);
% Mostramos la tabla
disp('DataTable:');
disp(df);
% Mostramos el tipo de dato de cada columna
disp('El tipo de datos es:');
disp(class(df));

DataTable:
                A          B          C     D    E      F
               ___    ___________    ___    _    _    _____
    Lunes        1    29-Nov-2022      1    3    2    test
    Martes     3.5    30-Nov-2022      2    4    5    train
    Miércoles  3.4    01-Dec-2022      3    5    8    test
    Jueves     1.4    02-Dec-2022    NaN    6    1    train

El tipo de datos es:
table

% Crear un array con los datos
datos = [1, 2, 3, 4; 2018, 2019, 2020, 2021; 125, 130, 232, 134]';
% Crear un dataframe en MATLAB
dfVentas = array2table(datos, 'VariableNames', {'meses', 'anios', 'ventas'});
% Mostrar el dataframe original
```

Autores: María Inmaculada Rodríguez García, María Gema Carrasco García, Javier González Enrique, Juan Jesús Ruiz Aguilar, Ignacio J. Turias Domínguez. Universidad de Cádiz

```
disp('Imprimimos el dataframe original dfVentas:');
disp(dfVentas);
% Establecer "años" como índice
dfVentas2 = table2timetable(dfVentas, 'RowTimes', datetime(dfVentas.anios, 1, 1));
% Mostrar el nuevo dataframe con "años" como índice
disp('Mostramos el nuevo dataframe dfVentas2:');
disp(dfVentas2);
```

```
Imprimimos el dataframe original dfVentas:
    meses    anios    ventas
    _____    _____    ______
      1      2018      125
      2      2019      130
      3      2020      232
      4      2021      134

Mostramos el nuevo dataframe dfVentas2:
       Time         meses    anios    ventas
    ____________    _____    _____    ______
    01-Jan-2018       1      2018      125
    01-Jan-2019       2      2019      130
    01-Jan-2020       3      2020      232
    01-Jan-2021       4      2021      134
```

```
% Otra forma:
% Crear un array con los datos
datos = [1, 2, 3, 4; 2018, 2019, 2020, 2021; 125, 130, 232, 134]';
% Crear un dataframe en MATLAB
dfVentas = array2table(datos, 'VariableNames', {'meses', 'anios', 'ventas'});
% Mostrar el dataframe original
disp('Imprimimos el dataframe original dfVentas:');
disp(dfVentas);
% Establecer "años" como índice sin eliminar la columna
dfVentas2 = dfVentas;
dfVentas2.Properties.RowNames = cellstr(num2str(dfVentas2.anios));
dfVentas2.anios = [];
disp('Mostramos el nuevo dataframe dfVentas2:');
disp(dfVentas2);
```

```
Imprimimos el dataframe original dfVentas:
    meses    anios    ventas
    _____    _____    ______
      1      2018      125
      2      2019      130
      3      2020      232
      4      2021      134

Mostramos el nuevo dataframe dfVentas2:
             meses    ventas
             _____    ______
    2018       1       125
    2019       2       130
    2020       3       232
    2021       4       134
```