Herramientas de programación en Python

3. Tipos estructurados para manejo de datos

3.2 **Series y DataFrame de Pandas**

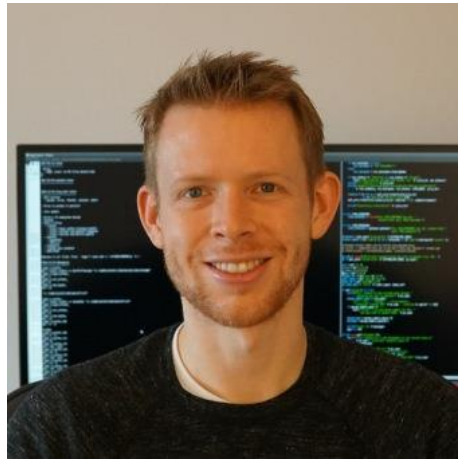
Pedro Gomis

pedro.gomis@upc.edu

Pandas

- **Pandas**[1] es un módulo (biblioteca o *library*) de Python que se usa para trabajar con conjuntos de datos. Tiene funciones para analizar, filtrar, explorar y manipular datos.

- El nombre **Pandas** hace referencia tanto a "***Data Panel*** (panel de datos)" como a "***Python Data Analysis*** (Análisis de datos de Python)" y fue creado por Wes McKinney[2, 3] en 2008



- Pandas se desarrolló a partir de los *arrays* de NumPy, pero con nuevas funcionalidades para representar datos tabulados etiquetados, similares a las hojas de cálculo.

- Algunos tutoriales de Pandas en **4** a **7**

1. https://pandas.pydata.org/
2. McKinney, Wes. (2011). "pandas: a Foundational Python Library for Data Analysis and Statistics" (PDF). recuperado de https://www.dlr.de/sc/Portaldata/15/Resources/dokumente/pyhpc2011/submissions/pyhpc2011_submission_9.pdf
3. https://wesmckinney.com/
4. https://aprendeconalf.es/docencia/python/manual/pandas/
5. https://www.python-course.eu/pandas.php
6. https://pandas.pydata.org/pandas-docs/stable/user_guide/dsintro.html
7. https://pandas.pydata.org/pandas-docs/stable/getting_started/tutorials.html

Pandas: tipos de datos

- Entre las principales características del módulo Pandas están:
  - Permite leer y escribir ficheros en formato CSV, MS Excel y bases de datos SQL
  - Se puede seleccionar y filtrar de manera sencilla tablas de datos en función de posición, valor o etiquetas
  - Ofrece métodos para reordenar, dividir y combinar conjuntos de datos.
  - Permite trabajar con series temporales

- Los tipos de datos en Pandas son:
  - **Series**: estructura de una dimensión (1D)
  - **DataFrame**: estructura de dos dimensiones (tablas, 2D) y
  - **Panels**: Estructura de tres dimensiones (tablas 3D). Este tipo de objeto **está obsoleto** y se ha **eliminado** en las versiones recientes. La forma recomendada de representar estos tipos de datos tridimensionales es con un MultiIndex en un DataFrame, a través del método Panel.to_frame ().

Pandas: Series

- Para comenzar, importaremos los módulos NumPy y Pandas

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

In [2]: import pandas as pd
```

Series

- Una serie (Series) es un objeto similar a una matriz etiquetada unidimensionalmente, capaz de contener cualquier tipo de datos (enteros, strings, reales –float-, objetos Python, etc.). Las etiquetas de los elementos se denominan índices (index). Por omisión, las etiquetas se numeran a partir del 0.

```
In [3]: S = pd.Series([11, 28, 72, 3, 5, 8])
In [4]: S
Out[4]:
0    11
1    28
2    72
3     3
4     5
5     8
dtype: int64
```

Pandas: Series

- En general, las Series se crean con la siguiente instrucción, donde se pueden definir las etiquetas de los índices:

```
S = pd.Series(data, index=indices)
```

- Por ejemplo, las etiquetas de los índices pueden ser de tipo string:

```
In [5]: frutas = ['mangos', 'naranjas', 'fresas', 'peras']
In [6]: cantidades = [20, 33, 52, 10]
In [7]: S = pd.Series(cantidades, index = frutas)
In [8]: S
Out[8]:
mangos      20
naranjas    33
fresas      52
peras       10
dtype: int64
```

Pandas: Series

- Es muy útil crear Series a partir de diccionarios. Los índices serán las claves:

```
In [9]: dicc = {'mangos': 20, 'naranjas': 33, 'fresas': 52, 'peras': 20}
In [10]: S = pd.Series(dicc)
In [11]: S
Out[11]:
mangos      20
naranjas    33
fresas      52
peras       10
dtype: int64
```

- Si agregamos una etiqueta en los índices que no esté en el diccionario, tendrá un valor perdido que se representa por **NaN** (Not-a-Number)

```
In [12]: S = pd.Series(dicc, index = ['mangos', 'naranjas', 'fresas', 'peras', 'kiwis'])
In [13]: S
Out[13]:
mangos      20.0
naranjas    33.0
fresas      52.0
peras       20.0
kiwis        NaN
dtype: float64
```

Pandas: Series

- **Indexación** o **accesos** a los elementos de las Series. Si se accede a un solo elemento se hace de forma similar a un diccionario, pero si quiere acceder a varios elementos, se **empaquetan en una lista**:

```
In [14]: S['mangos']
Out[14]: 20.0
In [15]: S[['mangos', 'fresas']]  # empaquetados em uma lista
Out[15]:
mangos    20.0
fresas    52.0
dtype: float64
```

- Se pueden acceder a los elementos también por su posición [0], [1], [1:3], etc

```
In [16]: S[0]
Out[16]: 20.0
In [17]: S[1:3]
Out[17]:
naranjas    33.0
fresas      52.0
dtype: float64
In [18]: S[2:]
Out[18]:
fresas    52.0
peras     20.0
kiwis      NaN
```

Pandas: Series

- **Atributos de las Series.** Se pueden **agregar** o **acceder** atributos al objeto Series.
  - s.name: se añade nombre a la Series
  - s.size : devuelve el número de elementos de la Series.
  - s.index : devuelve una lista con los nombres de las filas de la Series.
  - s.dtype : devuelve el tipo de datos de los elementos de la Series.

```
In [19]: S.name = 'Frutas de la compra'
In [20]: S.size
Out[20]: 20.0
In [21]: S.size
Out[21]: 5
In [22]: S.index
Out[22]: Index(['mangos', 'naranjas', 'fresas', 'peras', 'kiwis'], dtype='object')
In [23]: S.dtype
Out[23]: dtype('float64')
In [24]: S.name
Out[24]: 'Frutas de la compra
```

Pandas: Series

Métodos (funciones internas) de las Series. Operaciones sobre los elementos de las Series. Los métodos ignoran los valores NaN

- s.count() : Devuelve el número de elementos que no son nulos ni NaN en la serie s.
- s.sum() : Devuelve la suma de los datos de la serie s cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena str.
- s.cumsum() : Devuelve una serie con la suma acumulada de los datos de la serie s cuando los datos son de un tipo numérico.
- s.value_counts() : Devuelve la serie con la frecuencia (número de repeticiones) de cada valor de la serie s.
- s.min() : Devuelve el menor de los datos de la serie s.
- s.max() : Devuelve el mayor de los datos de la serie s.
- s.mean() : Devuelve la media de los datos de la serie s cuando los datos son de un tipo numérico.
- s.std() : Devuelve la desviación típica de los datos de la serie s cuando los datos son de un tipo numérico.
- s.describe(): Devuelve una serie con un resumen descriptivo que incluye el número de datos, su suma, el mínimo, el máximo, la media, la desviación típica y los cuartiles.

Pandas: Series

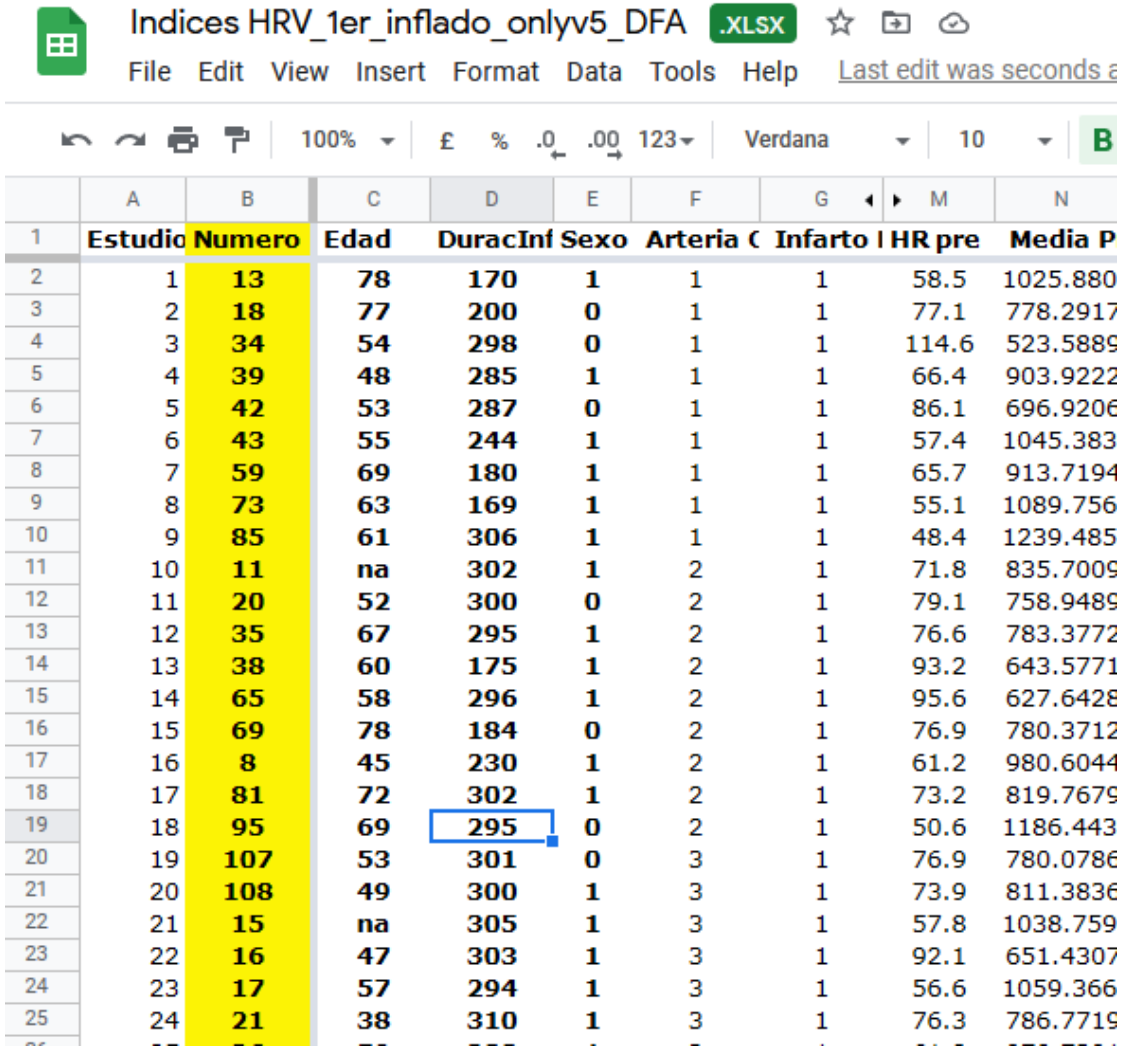
Métodos (funciones internas) de las Series. Operaciones sobre los elementos de las Series. Los métodos ignoran los valores NaN. Ejemplos sobre la Series 'Frutas de la compra'

```
In [25]: S.name = 'Frutas de la compra'
Out[25]: 'Frutas de la compra'
In [26]: S.count()
Out[26]: 4
In [27]: S.sum()
Out[27]: 125.0
In [28]: S.min()
Out[28]: 20.0
In [29]: S.mean()
Out[29]: 31.25
In [30]: S.describe()
Out[30]:
count     4.000000
mean     31.250000
std      15.129992
min      20.000000
25%      20.000000
50%      26.500000
75%      37.750000
max      52.000000
Name: Frutas de la compra, dtype: float64
```

Pandas: DataFrames

- En Python-Pandas un **DataFrame** es un tipo de objeto que define una estructura de datos etiquetada bidimensional con columnas, de tipos que pueden ser diferentes, en forma de **tabla.**

- Se puede relacionar a una hoja de cálculo o una tabla SQL, o un diccionario de objetos Series. Están inspiradas también en los Data.frame del lenguaje R.
- Generalmente, es el objeto **pandas** más utilizado.
- Cada columna es de tipo **Series**, por lo que los datos de cada columna son del mismo tipo. Se puede considerar como una concatenación de Series, donde cada Series tiene a su vez un índice (colums).
- En los DataFrame se puede especificar tanto el **index** (el nombre de las filas) como **columns** (el nombre de las columnas).



	A	B	C	D	E	F	G	M	N
1	Estudio	Numero	Edad	DuracInf	Sexo	Arteria C	Infarto I	HR pre	Media P
2	1	13	78	170	1	1	1	58.5	1025.880
3	2	18	77	200	0	1	1	77.1	778.2917
4	3	34	54	298	0	1	1	114.6	523.5889
5	4	39	48	285	1	1	1	66.4	903.9222
6	5	42	53	287	0	1	1	86.1	696.9206
7	6	43	55	244	1	1	1	57.4	1045.383
8	7	59	69	180	1	1	1	65.7	913.7194
9	8	73	63	169	1	1	1	55.1	1089.756
10	9	85	61	306	1	1	1	48.4	1239.485
11	10	11	na	302	1	2	1	71.8	835.7009
12	11	20	52	300	0	2	1	79.1	758.9489
13	12	35	67	295	1	2	1	76.6	783.3772
14	13	38	60	175	1	2	1	93.2	643.5771
15	14	65	58	296	1	2	1	95.6	627.6428
16	15	69	78	184	0	2	1	76.9	780.3712
17	16	8	45	230	1	2	1	61.2	980.6044
18	17	81	72	302	1	2	1	73.2	819.7679
19	18	95	69	295	0	2	1	50.6	1186.443
20	19	107	53	301	0	3	1	76.9	780.0786
21	20	108	49	300	1	3	1	73.9	811.3836
22	21	15	na	305	1	3	1	57.8	1038.759
23	22	16	47	303	1	3	1	92.1	651.4307
24	23	17	57	294	1	3	1	56.6	1059.366
25	24	21	38	310	1	3	1	76.3	786.7719

Pandas: DataFrames

- **DataFrame a partir de un diccionario de listas.** Las claves son los nombres de las columnas y los valores son listas todas del mismo tamaño, que formarán los datos de las columnas. Ejemplo:

```
In [1]: d = {'nombre':['Juan','Carla','Judith'], 'edad':[23, 28, 25], 'peso': [70.5, 56.7, 51.2]}
In [2]: df = pd.DataFrame(d)
In [3]: df
Out[3]:
   nombre  edad  peso
0    Juan    23  70.5
1   Carla    28  56.7
2  Judith    25  51.2
```

- Se pueden agregar etiquetas a los índices (filas) del DataFrame

```
In [4]: df = pd.DataFrame(d, index = ['id1', 'id2', 'id3'])
In [5]: df
Out[5]:
      nombre  edad  peso
id1    Juan    23  70.5
id2   Carla    28  56.7
id3  Judith    25  51.2
```

Pandas: DataFrames

- **DataFrame a partir listas de listas**. Sintaxis:

```
DataFrame(data =listas, index = filas, columns = columnas, dtype=tipos)
```

```
In [4]: lst = [['Juan',23,70.5],['Carla',28,56.7],['Judith',25,51.2]]
In [5]: df = pd.DataFrame(lst, columns=['nombre','edad','peso'],index = ['id1', 'id2', 'id3'])
In [6]: df
Out[6]:
     nombre  edad  peso
id1    Juan    23  70.5
id2   Carla    28  56.7
id3  Judith    25  51.2
```

- **DataFrame a partir de un diccionario de Series**.

```
In [7]: S1 = pd.Series(['Juan','Carla','Judith'], index=['id1','id2','id3'])
In [8]: S2 = pd.Series([23, 28, 25], index=['id1','id2','id3'])
In [9]: S3 = pd.Series([70.5, 56.7, 51.2], index=['id1','id2','id3'])
In [10]: d1 = {'nombre': S1, 'edad': S2, 'peso': S3}
In [11]: df1 = pd.DataFrame(d1)
In [12]: df1
Out[12]:
     nombre  edad  peso
id1    Juan    23  70.5
id2   Carla    28  56.7
id3  Judith    25  51.2
```

Pandas: DataFrames

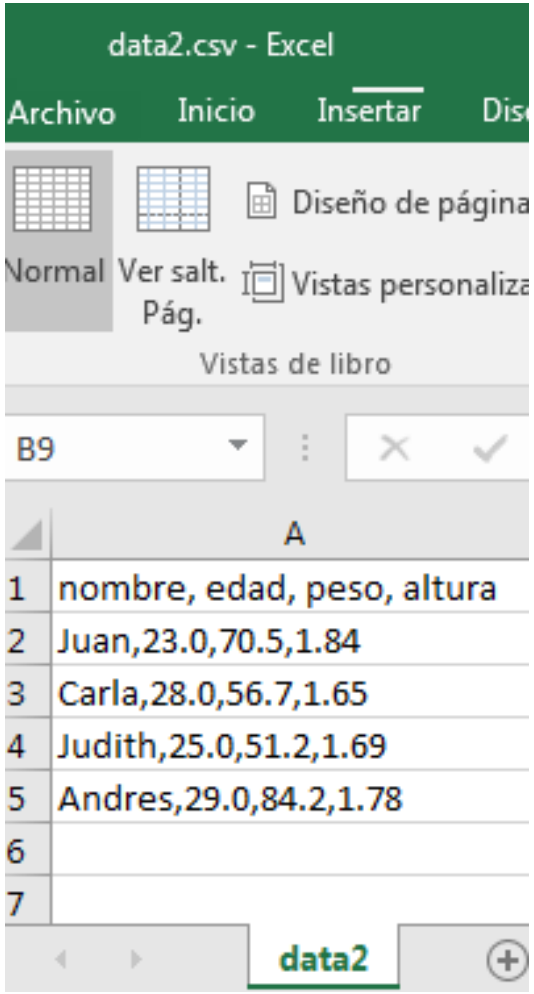
- **DataFrame a partir de un diccionario de Series**. Incluye otra Series, S4, con una fila adicional por lo que las filas de las otras columnas aparecerán con NaN

```
In [13]: S4 = pd.Series([1.84, 1.65, 1.69, 1.78], index=['id1','id2','id3', 'id4'])
In [14]: d2 = {'nombre': S1, 'edad': S2, 'peso': S3, 'altura': S4}
In [15]: df2 = pd.DataFrame(d2)
In [16]: df2
Out[16]:
     nombre  edad  peso  altura
id1    Juan  23.0  70.5    1.84
id2   Carla  28.0  56.7    1.65
id3  Judith  25.0  51.2    1.69
id4     NaN   NaN   NaN    1.78
```

Pandas: DataFrames

- **DataFrame a partir de un fichero CSV* o Excel**.

```
In [17]: df3 = pd.read_csv('data2.csv')
In [18]: df3
Out[18]:
   nombre   edad   peso   altura
0    Juan   23.0   70.5     1.84
1   Carla   28.0   56.7     1.65
2  Judith   25.0   51.2     1.69
3  Andres   29.0   84.2     1.78
```



* Fichero data2.csv disponible en el aula. Debe estar en el directorio de trabajo. Alternativamente, incluir la ruta del fichero (path), de la forma

```
In [17]: df3 = pd.read_csv('F:/VIU/MasterBig_data_Master_IA/16MBID/data2.csv')
```

Pandas: DataFrames

- **DataFrame: acceso a los elementos mediante las posiciones**.
  - `df.iloc[i, j]` : Devuelve el elemento que se encuentra en la fila i y la columna j del DataFrame df Pueden indicarse secuencias de índices para obtener partes del DataFrame.
  - `df.iloc[i]` : Devuelve una serie con los elementos de la fila i del DataFrame df

```
In [19]: df3
Out[19]:
   nombre   edad   peso   altura
0    Juan   23.0   70.5     1.84
1   Carla   28.0   56.7     1.65
2  Judith   25.0   51.2     1.69
3  Andres   29.0   84.2     1.78
In [20]: df3.iloc[0, 0]
Out[20]: 'Juan'
In [21]: df3.iloc[1]
Out[21]:
nombre     Carla
 edad         28
 peso       56.7
 altura     1.65
Name: 1, dtype: object
```

Pandas: DataFrames

- **DataFrame: acceso a los elementos mediante nombres**.
  - `df.loc[fila, columna]` : Devuelve el elemento que se encuentra en la fila con nombre fila y la columna de con nombre columna del DataFrame df
  - `df[columna]` : Devuelve una serie con los elementos de la columna de nombre columna del DataFrame df

```
In [22]: df
Out[22]:
     nombre  edad  peso
id1    Juan    23  70.5
id2   Carla    28  56.7
id3  Judith    25  51.2
In [23]: df.loc['id1','nombre']
Out[23]: 'Juan'
In [24]: df.loc['id1']
Out[24]:
nombre    Juan
edad        23
peso      70.5
Name: id1, dtype: object
```

```
In [25]: df.loc['id1',['nombre','peso']]
Out[25]:
nombre    Juan
peso      70.5
Name: id1, dtype: object
```

Pandas: DataFrames

- **DataFrame: operaciones con columnas**.
  - **Añadir** una nueva columna a un DataFrame es similar al de añadir un nuevo par clave-valor a un diccionario, pero poniendo los valores de la columna en una lista o Series.
  - Se puede operar sobre las columnas, como hallar el ***índice de masa muscular***, IMC = peso/altura**2.

```
In [26]: df
Out[26]:
     nombre  edad  peso
id1    Juan    23  70.5
id2   Carla    28  56.7
id3  Judith    25  51.2
In [27]: df['altura'] = [1.84, 1.65, 1.69]   # se añade la columna 'altura'
Out[27]:
     nombre  edad  peso  altura
id1    Juan    23  70.5    1.84
id2   Carla    28  56.7    1.65
id3  Judith    25  51.2    1.69
In [28]: df['IMC'] = df['peso']/df['altura']**2  # se calcula y añade columna 'IMC'
In [29]: df
Out[29]:
     nombre  edad  peso  altura        IMC
id1    Juan    23  70.5    1.84  20.823488
id2   Carla    28  56.7    1.65  20.826446
id3  Judith    25  51.2    1.69  17.926543
```

Pandas: DataFrames

- **Atributos de los DataFrame.** Se pueden **agregar** o **acceder** atributos al objeto DataFrame. Algunos son:
  - df.name: se añade nombre al DataFrame
  - df.shape : devuelve una tupla con el número de filas y columnas del DataFrame df
  - df.size : devuelve el número de elementos de la Series.
  - df.columns : devuelve una lista con los nombres de las columnas del DataFrame df
  - df.index : devuelve una lista con los nombres de las filas del DataFrame df
  - df.dtypes : devuelve el tipo de datos de las columnas del DataFrame df
  - df.head(n) : devuelve las n primeras filas del DataFrame df
  - df.tail(n) : devuelve las n últimas filas del DataFrame df

```
In [30]: df.name = ' Datos clínicos '
In [31]: df.size
Out[31]: 15
In [32]: df.shape
Out[32]: (3, 5)
In [33]: df.columns
Out[33]: Index(['nombre', 'edad', 'peso', 'altura', 'IMC'], dtype='object')
In [34]: df.index
Out[34]: Index(['id1', 'id2', 'id3'], dtype='object')
In [35]: df.head(2)
Out[35]:
    nombre  edad  peso  altura        IMC
id1   Juan    23  70.5    1.84  20.823488
id2  Carla    28  56.7    1.65  20.826446
```

```
In [36]: df.dtypes
Out[36]:
nombre     object
edad        int64
peso      float64
altura    float64
IMC       float64
dtype: object
```

Pandas: DataFrames

Métodos (funciones internas) de los DataFrames. Operaciones sobre los elementos de las tablas. Los métodos ignoran los valores NaN. Algunos de los métodos son:

- df.count() : Devuelve el número de elementos que no son nulos ni NaN en la serie s.
- df.sum() : Devuelve la suma de los datos de la serie s cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena str.
- df.cumsum() : Devuelve una serie con la suma acumulada de los datos de la serie s cuando los datos son de un tipo numérico.
- df.value_counts() : Devuelve la serie con la frecuencia (número de repeticiones) de cada valor de la serie s.
- df.min() : Devuelve el menor de los datos de la serie s.
- df.max() : Devuelve el mayor de los datos de la serie s.
- df.mean() : Devuelve la media de los datos de la serie s cuando los datos son de un tipo numérico.
- df.std() : Devuelve la desviación típica de los datos de la serie s cuando los datos son de un tipo numérico.
- df.describe(): Devuelve una serie con un resumen descriptivo que incluye el número de datos, su suma, el mínimo, el máximo, la media, la desviación típica y los cuartiles.

Pandas: DataFrames

- **DataFrame: métodos**. Ejemplos con df3

```
In [37]: df3.min()
Out[37]:
nombre     Andres
 edad          23
 peso        51.2
 altura      1.65
In [38]: df3.mean()
Out[38]:
 edad       26.25
 peso       65.65
 altura      1.74
In [39]: df3.describe()
Out[39]:
             edad       peso    altura
count    4.000000   4.000000  4.000000
mean    26.250000  65.650000  1.740000
std      2.753785  14.793354  0.086023
min     23.000000  51.200000  1.650000
25%     24.500000  55.325000  1.680000
50%     26.500000  63.600000  1.735000
75%     28.250000  73.925000  1.795000
max     29.000000  84.200000  1.840000
```