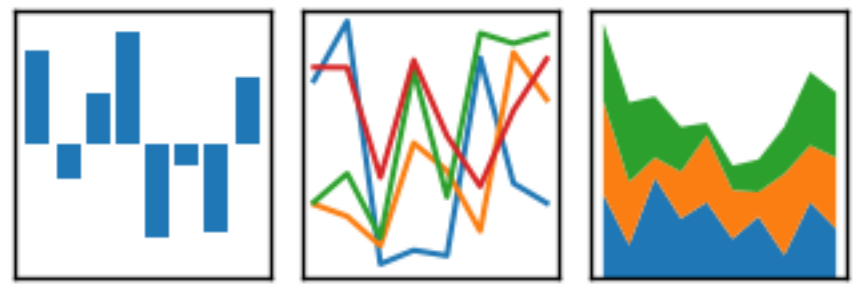
pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- Pandas (https://pandas.pydata.org)

Biblioteca de Python para manipulación y análisis de datos.

Entre sus puntos fuertes podríamos destacar los siguientes:

- Incorpora estructuras de datos muy potentes y flexibles para el manejo de datos.
- Incorpora multitud de funciones o herramientas para el análisis de datos de forma fácil e intuitiva.
- Es software libre multiplataforma.
- Pyplot está integrado en Pandas, lo que permite la representación de la información de forma directa.

- Instalación de Pandas

Hay diferentes métodos para instalar Pandas, pero quizás el más sencillo sea a través del gestor de paquetes de Python denominado *PIP3* (https://piplt.pypa.io/en/stable/).

```
cc — -bash — 167×23
MacBook-Pro-de-cc:~ cc$ pip3 install pandas
Collecting pandas
  Using cached https://files.pythonhosted.org/packages/39/73/99aa822ee88cef5829607217c11bf24ecc1171ae5d49d5f780085f5da518/pandas-0.25.1-cp37-cp37m-macosx_10_9_x86_64.m
acosx_10_10_x86_64.whl
Requirement already satisfied: pytz>=2017.2 in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from pandas) (2018.9)
Requirement already satisfied: python-dateutil>=2.6.1 in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from pandas) (2.8.0)
Requirement already satisfied: numpy>=1.13.3 in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from pandas) (1.17.2)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from python-dateutil>=2.6.1->pandas) (1.12.0)
Installing collected packages: pandas
Successfully installed pandas-0.25.1
MacBook-Pro-de-cc:~ cc$
```

- Importar pandas en un intérprete de Python 3.

```
>>> import pandas as pd
>>>
```

- Podemos ver la versión de pandas:

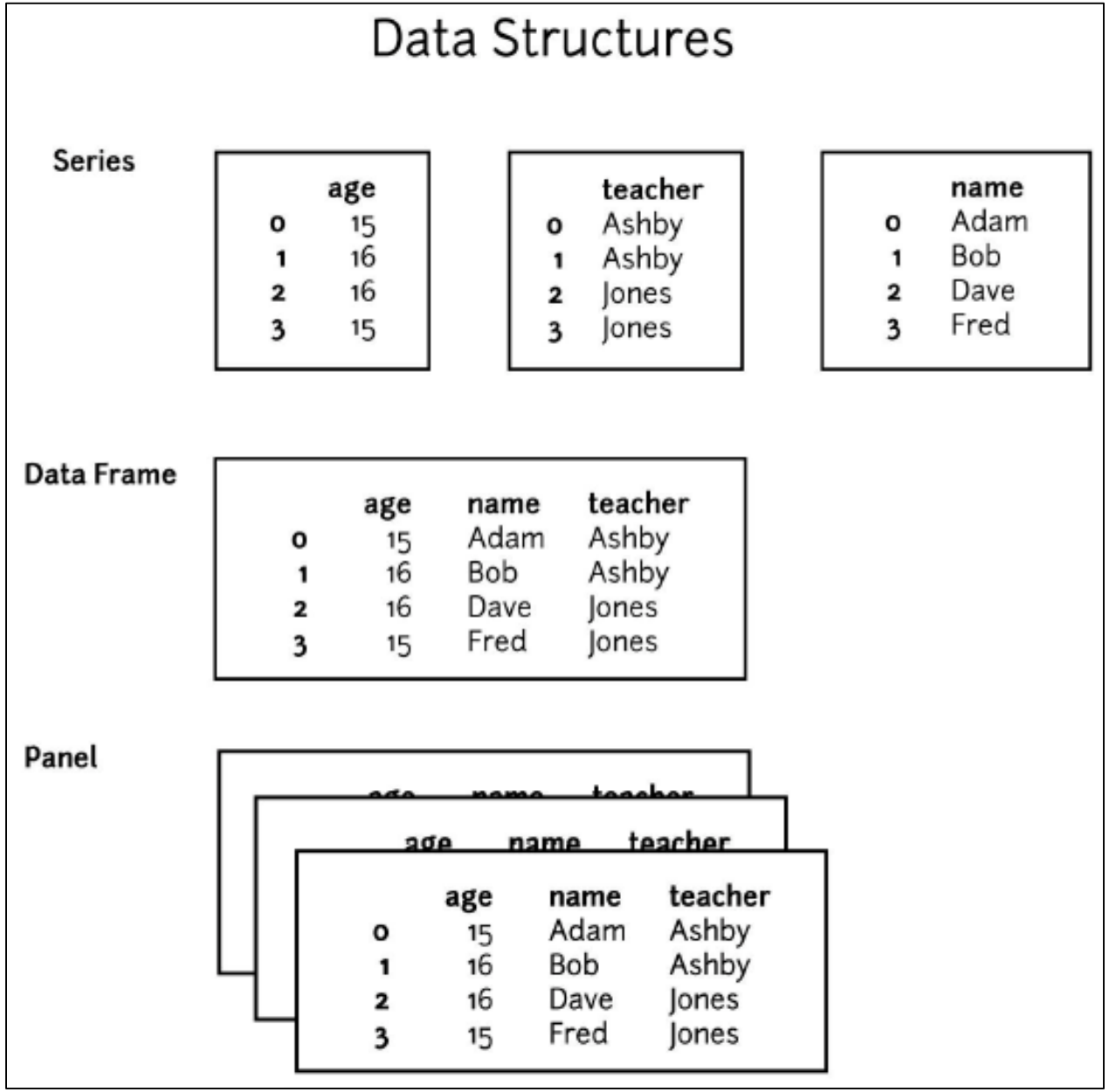
```
>>> import pandas
>>> pandas.__version__
'0.25.1'
```

En caso de que Matplotlib no esté correctamente instalada aparecería este mensaje:

ModuleNotFoundError: No module named 'pandas'

- Estructuras de datos en Pandas

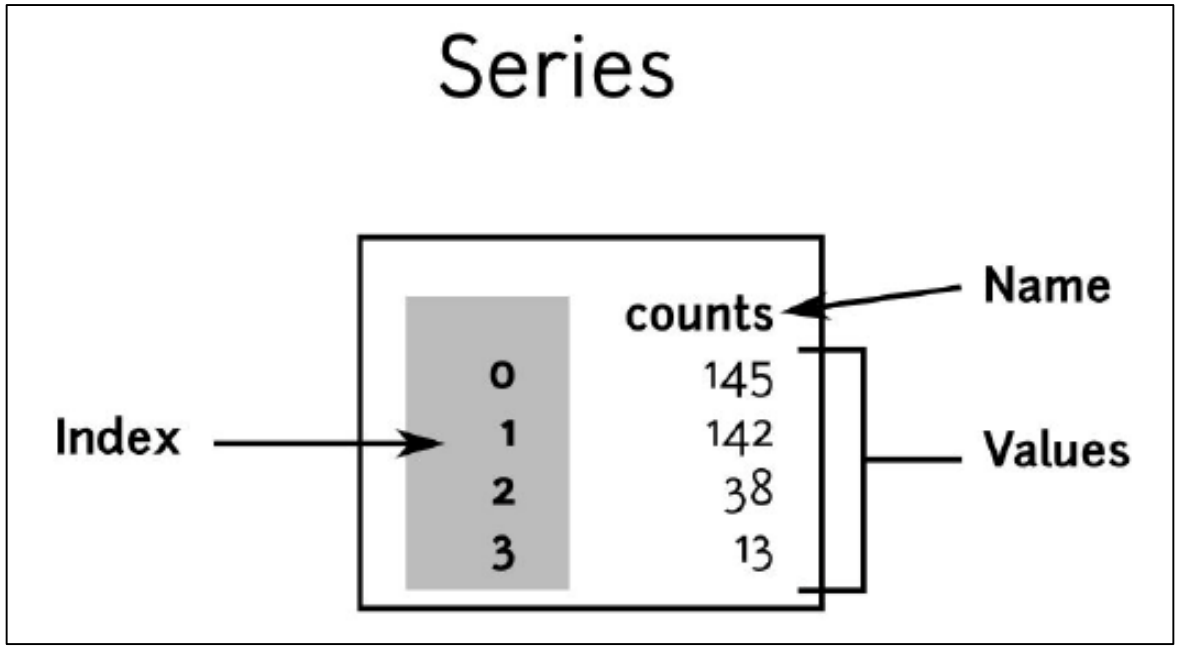
Data Structure	Dimensionality	Spreadsheet Analog
`Series`	1D	Column
`DataFrame`	2D	Single Sheet
`Panel`	3D	Multiple Sheets



- La estructura de datos *Series* (i)

Una serie es un objeto que sirve para modelar datos de una sola dimensión. Un objeto del tipo Series consta de un conjunto de datos unidimensional y de un nombre y un índice asociados a dichos datos.

Pueden generarse a partir de arrays, listas o diccionarios.



- La estructura de datos *Series* (ii): creando una serie

```
 9 import pandas as pd
10
11 h = [178, 159, 165, 198, 169, 167, 178, 177]
12 alturas = pd.Series(h, name = 'altura')
13 print(alturas)
14
15 nombres = pd.Series(['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'], name = 'nombre')
16 print(nombres)
```

```
In [12]: print(alturas)
0    178
1    159
2    165
3    198
4    169
5    167
6    178
7    177
Name: altura, dtype: int64

In [13]: print(nombres)
0      Pedro
1        Ana
2      Paula
3    Antonio
4      Julia
5     Isabel
6       Paco
7     Manuel
Name: nombre, dtype: object
```

Si no se especifica un índice, se asigna una secuencia de números enteros por defecto.

- La estructura de datos *Series* (iii): asignando un índice

```
 9 import pandas as pd
10
11 h = [178, 159, 165, 198, 169, 167, 178, 177]
12 n = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
13
14 datos = pd.Series(h, name = 'datos', index = n)
15 print(datos)
```

```
Pedro      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Name: datos, dtype: int64
```

Interesante: el índice no tiene por qué ser numérico ni único (puede tener valores repetidos).

- La estructura de datos *Series* (iv): accediendo a elementos de forma individual

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                   index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 print(datos[2])
15 print(datos[-3])
16
17 print(datos['Paco'])
18 ana = datos['Ana']
19 print(ana)
20 print(type(ana))
```

```
165
167
178
159
<class 'numpy.int64'>
```

Cuando accedemos a datos de forma individual obtenemos un elemento del tipo correspondiente a los incluidos en la creación de la lista. En el ejemplo anterior se obtiene como resultados datos del tipo *numpy.int64*.

- La estructura de datos *Series* (v): accediendo a subconjuntos de datos de la serie

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                   index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 print(datos[3 : 6], '\n')
15
16 print(datos[[3,4,5]], '\n')
17
18 datos_ap = datos['Ana' : 'Paco']
19 print(datos_ap, '\n')
20
21 print(type(datos_ap), '\n')
```

```
Antonio    198
Julia      169
Isabel     167
Name: datos, dtype: int64

Antonio    198
Julia      169
Isabel     167
Name: datos, dtype: int64

Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Name: datos, dtype: int64

<class 'pandas.core.series.Series'>
```

Cuando accedemos a rangos de datos obtenemos un nuevo objeto de tipo *Seires* con el subconjunto de datos indicado. En el ejemplo anterior se obtiene como resultados datos del tipo *pandas.core.series.Series*.

- La estructura de datos *Series* (vi): creando listas a partir de la serie

- Creando una lista con los índices de la serie

indices = serie.index.tolist()

- Creando una lista con los valores de la serie

valores = serie.values.tolist()

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 indices = datos.index.tolist()
15 print(indices, "\n")
16
17 valores = datos.values.tolist()
18 print(valores)
```

```
['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']

[178, 159, 165, 198, 169, 167, 178, 177]
```

- La estructura de datos *Series* (vii): creando arrays a partir de la serie

- Creando un array con los índices de la serie

indices = serie.index.values

- Creando un array con los valores de la serie

valores = serie.values

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 indices = datos.index.values
15 print(indices, "\n")
16
17 valores = datos.values
18 print(valores)
```

```
['Pedro' 'Ana' 'Paula' 'Antonio' 'Julia' 'Isabel' 'Paco' 'Manuel']

[178 159 165 198 169 167 178 177]
```

- La estructura de datos *Series* (viii): comprobando la pertenencia de elementos a la serie

Para comprobar la pertenencia de elementos a la serie podemos hacerlo a través del índice:

valor_indice in serie

Podemos consultar también por valor. En este caso, la sintaxis sería la siguiente:

valor in serie

En ambos casos los valores devueltos serían True o False dependiendo de si el elemento consultado pertenece o no a la serie.

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 print(169 in datos.values)
15 print(170 in datos.values,'\n')
16
17 print('Paula' in datos)
18 print('Francisco' in datos)
```

```
True
False

True
False
```

- La estructura de datos *Series* (ix): iterando en una estructura *Series*:

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14
15 for item in datos:
16     print(item)
17
18 for item in datos.index:
19     print(item)
20
21 for item in datos.iteritems():
22     print(item)
```

```
178
159
165
198
169
167
178
177
Pedro
Ana
Paula
Antonio
Julia
Isabel
Paco
Manuel
('Pedro', 178)
('Ana', 159)
('Paula', 165)
('Antonio', 198)
('Julia', 169)
('Isabel', 167)
('Paco', 178)
('Manuel', 177)
```

- La estructura de datos *Series* (x): actualizando valores

Para actualizar el valor de un elemento de la serie podemos hacerlo a través del índice con la notación de arrays:

serie[valor_indice] = nuevo_valor

También a través de la posición del elemento en la serie:

serie[posicion] = nuevo_valor

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                   index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 datos['Pedro'] = 168
15 datos[3] = 202
16 print(datos, '\n')
```

```
Pedro      168
Ana        159
Paula      165
Antonio    202
Julia      169
Isabel     167
Paco       178
Manuel     177
Name: datos, dt
```

- La estructura de datos *Series* (xi): actualizaciones de elementos repetidos

Podemos cambiar todos los valores para una misma etiqueta o valor de índice con una simple Asignación. No obstante, si queremos asignar diferentes valores a elementos repetidos, tendremos que hacerlo a través de su posición.

La siguiente instrucción obtiene todos las posiciones de un determinado valor de índice:

pos = [i for i, x in enumerate(datos.iteritems()) if (x[0] == 'Pedro')]

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14
15 datos = datos.append(pd.Series([169], index = ['Julia']))
16 datos = datos.append(pd.Series([209], index = ['Pedro']))
17 print(datos, '\n')
18
19 datos['Pedro'] = 200
20 print(datos, '\n')
21
22 pos = [i for i, x in enumerate(datos.iteritems()) if (x[0] == 'Pedro')]
23
24 datos[pos[0]] = 210
25 datos[pos[1]] = 150
26 print(datos, '\n')
```

```
Pedro      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Julia      169
Pedro      209
dtype: int64

Pedro      200
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Julia      169
Pedro      200
dtype: int64

Pedro      210
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Julia      169
Pedro      150
dtype: int64
```

- La estructura de datos *Series* (xii): añadiendo elementos a una serie

serie.append(serie2)

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 datos = datos.append(pd.Series([169], index = ['Rosa']))
15 datos = datos.append(pd.Series([209], index = ['Pedro']))
16 print(datos, '\n')
17 print(datos['Pedro'])
```

```
Pedro      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Rosa       169
Pedro      209
dtype: int64

Pedro    178
Pedro    209
dtype: int64
```

*Nota: rename devuelve una nueva serie con el resultado, pero **no** almacena los cambios en la serie que realiza la llamada, por lo que si queremos que los cambios se vean reflejados hay que asignar la operación a otra serie o a sí misma.*

- La estructura de datos *Series* (xiii): modificando un índice

serie.rename({valor_indice : nuevo_valor, parametros_opcionales})

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 datos = datos.append(pd.Series([169], index = ['Rosa']))
15 datos = datos.append(pd.Series([209], index = ['Pedro']))
16
17 datos.rename({'Pedro':'Lucas'})
18 print(datos, '\n')
19
20 datos2 = datos.rename({'Pedro' : 'Lucas'})
21 print(datos2, '\n')
22
23 datos.rename({'Pedro':'Lucas'}, inplace = True)
24 print(datos)
```

```
Pedro      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Rosa       169
Pedro      209
dtype: int64

Lucas      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Rosa       169
Lucas      209
dtype: int64

Lucas      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Rosa       169
Lucas      209
dtype: int64
```

*Nota: rename devuelve una nueva serie con el resultado, por lo que los cambios **no** se producen en la serie que llama a la función, a no ser que se establezca el parámetro inplace = True*, o bien, se asigne a sí misma.

- La estructura de datos *Series* (xiv): estableciendo un nuevo índice para la serie

serie.index = nuevo:indice

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 new_index = ['Sergio', 'Jose', 'Gregorio', 'Josefina', 'Maria', 'Luis', 'Sofia', 'Guadalupe']
15 datos.index = new_index
16
```

```
Sergio      178
Jose        159
Gregorio    165
Josefina    198
Maria       169
Luis        167
Sofia       178
Guadalupe   177
```

- La estructura de datos *Series* (xv): borrando elementos de una serie

serie.drop(elemento, parametros_opcionales)

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 datos.drop('Antonio', inplace = True)
15
16 print(datos)
```

```
Pedro      178
Ana        159
Paula      165
Julia      169
Isabel     167
Paco       178
Manuel     177
```

Nota: los cambios en la serie no se reflejan en la misma, a no ser que se especifique con el parámetro *inplace = True*, o bien, se asigne a sí misma.

- La estructura de datos *Series* (xvi): borrando elementos duplicados de una serie

serie.drop_duplicates()

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 d2 = datos.drop_duplicates()
15
16 print(datos, '\n')
17 print(d2)
```

```
Pedro      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Name: datos, dt

Pedro      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Manuel     177
```

Nota: los cambios en la serie no se reflejan en la misma, a no ser que se especifique con el parámetro *inplace = True*, o bien, se asigne a sí misma.

- La estructura de datos *Series* (xvii): otras operaciones (i)

Operation	Result
`+`	Adds scalar (or series with matching index values) returns `Series`
`-`	Subtracts scalar (or series with matching index values) returns `Series`
`/`	Divides scalar (or series with matching index values) returns `Series`
`//`	"Floor" Divides scalar (or series with matching index values) returns `Series`
`*`	Multiplies scalar (or series with matching index values) returns `Series`
`%`	Modulus scalar (or series with matching index values) returns `Series`
`==, !=`	Equality scalar (or series with matching index values) returns `Series`
`>, <`	Greater/less than scalar (or series with matching index values) returns `Series`
`>=, <=`	Greater/less than or equal scalar (or series with matching index values) returns `Series`
`^`	Binary XOR returns `Series`
`|`	Binary OR returns `Series`
`&`	Binary AND returns `Series`

- La estructura de datos *Series* (xviii): otras operaciones (ii)

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                    index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13 print(datos, "\n")
14
15 datos = datos * 1.1
16 print(datos, "\n")
17
18 consulta_altura = [nombre for nombre, valor in datos.iteritems() if (valor > 190)]
19 print("Mas altos de 190cm: ", consulta_altura, "\n")
20
21 consulta_nombres = [[nombre, valor] for nombre, valor in datos.iteritems() if (nombre in ['Ana', 'Pedro', 'Juanito'])]
22 print("Alturas Subconjunto: ", consulta_nombres, "\n")
```

```
Pedro      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Name: datos, dtype: int64
```

```
Pedro      195.8
Ana        174.9
Paula      181.5
Antonio    217.8
Julia      185.9
Isabel     183.7
Paco       195.8
Manuel     194.7
Name: datos, dtype: float64
```

```
Mas altos de 190cm:  ['Pedro', 'Antonio', 'Paco', 'Manuel']

Alturas Subconjunto:  [['Pedro', 195.8], ['Ana', 174.9]]
```

- La estructura de datos *Series* (xix): mostrando cabecera y cola de la serie

serie.head(num_elementos)	Devuelve una serie con los primeros *num_elementos* de la serie.
serie.tail(num_elementos)	Devuelve una serie con los últimos *num_elementos* de la serie.

```
 9 import pandas as pd
10 import random
11
12 aleatorios = [random.randint(0,1000) for i in range(101)]
13 datos = pd.Series(aleatorios)
14 print(datos.head(), '\n')
15 print(datos.tail(10))
```

```
0     546
1     382
2     906
3     309
4     747
dtype: int64

91      454
92      200
93      410
94      912
95      755
96     1000
97      174
98      722
99       52
100     756
dtype: int64
```

Nota: si se omite el parámetro *num_elementos* se tomará el valor *5* por defecto

- La estructura de datos *Series* (xx): ordenación de los datos

Podemos ordenar atendiendo al índice de la serie o a los valores de la misma.

serie.sort_index(ascending, inplace, otros_parametros)
serie.sort_values(ascending, inplace, otros_parametros)

- ascending: para ordenar de forma ascendente (por defecto = True)
- inplace: para que los cambios se hagan definitivos en la propia serie (por defecto = False)

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                   index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13
14 # muestro los datos de la serie ordenada por nombre ascendentemente
15 datos_ordenados_indice = datos.sort_index()
16 print(datos_ordenados_indice, '\n')
17
18 # muestro los datos de la serie de mayor a menor altura
19 datos_ordenados_valor = datos.sort_values(ascending = False)
20 print(datos_ordenados_valor)
```

```
Ana         159
Antonio     198
Isabel      167
Julia       169
Manuel      177
Paco        178
Paula       165
Pedro       178
Name: datos, dtype: int64

Antonio     198
Paco        178
Pedro       178
Manuel      177
Julia       169
Isabel      167
Paula       165
Ana         159
Name: datos, dtype: int64
```

- La estructura de datos *Series* (xxi): contando elementos

serie.count()	Devuelve el número de elementos de la serie.
serie.value_counts()	Devuelve una serie con cada valor y su número de ocurrencias.
serie.nunique()	Devuelve el número de datos sin repetir que hay
serie.unique()	Devuelve un array con los valores de la serie sin repeticiones

```
 9 import pandas as pd
10
11 datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
12                   index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
13 print(datos, "\n")
14
15 num_elems = datos.count()
16 print("num_elementos", num_elems , "\n")
17
18 frecs = datos.value_counts()
19 print("Frecuencia de los elementos:\n", frecs, "\n")
20
21 num_elem_sin_repes = datos.nunique()
22 print("Hay", num_elem_sin_repes, "datos sin repetir \n")
23
24 unicos = datos.unique()
25 print("Elementos sin repeticiones: ", unicos)
```

```
Pedro      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Name: datos, dtype: int64

num_elementos 8

Frecuencia de los elementos:
 178    2
159     1
169     1
167     1
198     1
165     1
177     1
Name: datos, dtype: int64

Hay 7  datos sin repetir

Elementos sin repeticiones:  [178 159 165 198 169 167 177]
```

- La estructura de datos *Series* (xxii): algunas funciones estadísticas

serie.sum()	Devuelve la suma de los elementos de la serie
serie.mean()	Devuelve la media de los valores de la serie
serie.median()	Devuelve la mediana de los valores de la serie
serie.quantile(valor)	Devuelve el percentil especificado entre 0 y 1
serie.min()	Devuelve el valor mínimo de la serie
serie.idxmin()	Devuelve la etiqueta correspondiente al valor mínimo de la serie
serie.max()	Devuelve el valor máximo de la serie
serie.idxmax()	Devuelve la etiqueta correspondiente al valor máximo de la serie
serie.var()	Devuelve la varianza de los datos de la serie
serie.std()	Devuelve la desviación estándar
serie.describe()	Devuelve una serie de datos de estadística descriptiva

Nota: hay más funciones estadísticas que se pueden consultar en la bibliografía.

```python
import pandas as pd

datos = pd.Series([178, 159, 165, 198, 169, 167, 178, 177] , name = 'datos',
                  index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia',
                           'Isabel', 'Paco', 'Manuel'])
print(datos, "\n")

suma = datos.sum()
media = datos.mean()
mediana = datos.median()
p50 = datos.quantile()
p10 = datos.quantile(.10)
p75 = datos.quantile(.75)
minimo = datos.min()
idmin = datos.idxmin()
maximo = datos.max()
idmax = datos.idxmax()
varianza = datos.var()
desviacion = datos.std()
estad = datos.describe()

print("suma:", suma, "media:", media, "mediana:", mediana)
print("minimo:", minimo, idmin, "maximo:", maximo, idmax)
print("p10:", p10, "p50:", p50, "p75:", p75, "\n")
print(estad)
```

```
Pedro      178
Ana        159
Paula      165
Antonio    198
Julia      169
Isabel     167
Paco       178
Manuel     177
Name: datos, dtype: int64

suma: 1391 media: 173.875 mediana: 173.0
minimo: 159 Ana maximo: 198 Antonio
p10: 163.2 p50: 173.0 p75: 178.0

count      8.000000
mean     173.875000
std       11.933596
min      159.000000
25%      166.500000
50%      173.000000
75%      178.000000
max      198.000000
Name: datos, dtype: float64
```

- La estructura de datos *Series* (xxiii): importar datos desde un fichero en formato csv

Para leer o importar un fichero de datos csv usaremos:

pd.read_csv(nombre_fichero, sep, index_col, squeeze, otras_opciones)

- *nombre_fichero* = nombre del fichero a importar.
- *sep* = separador de datos (por defecto ‘,’)
- *index_col* = columna que se usará como índice (si la hay)
- *squeeze* = si queremos que se importe a una serie (por defecto a un dataframe)

Ejemplo:

data = pd.read_csv(‘datos.csv', sep = “;“ , index_col = 'indice', squeeze = True)

https://pandas.pydata.org/pandas-docs/version/0.25/reference/api/pandas.read_csv.html?highlight=read_csv

- La estructura de datos *Series* (xxiv): exportar datos a un fichero en formato csv

Para guardar los datos de una serie en un fichero csv:

series.to_csv(nombre_fichero, sep, otras_opciones)

- *nombre_fichero* = nombre del fichero a importar.
- *sep* = separador de datos (por defecto ‘,’)

Ejemplo:

datos.to_csv(‘aturas.csv', sep = “;“)

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.to_csv.html

- La estructura de datos *Series* (xxv): la función plot() de una serie

Las series incorporan el método plot() que nos permite realizar de forma directa los siguientes tipos de gráficas:

PLOT METHODS	RESULT
`plot.area`	Creates an area plot for numeric columns
`plot.bar`	Creates a bar plot for numeric columns
`plot.barh`	Creates a horizonal bar plot for numeric columns
`plot.box`	Creates a box plot for numeric columns
`plot.density`	Creates a kernel density estimation plot for numeric columns (also `plot.kde`)
`plot.hist`	Creates a histogram for numeric columns
`plot.line`	Create a line plot. Plots index on x column, and numeric column values for y
`plot.pie`	Create a pie plot.

plot(kind, otros_parámetros)

kind = tipo de gráfico a realizar.

Por ejemplo: datos.plot(kind = ‘bar’, label = ‘grafico_barras’)

Ejercicio 1. Cargar los datos del fichero edades.csv en una serie de pandas y realizar las siguientes operaciones:

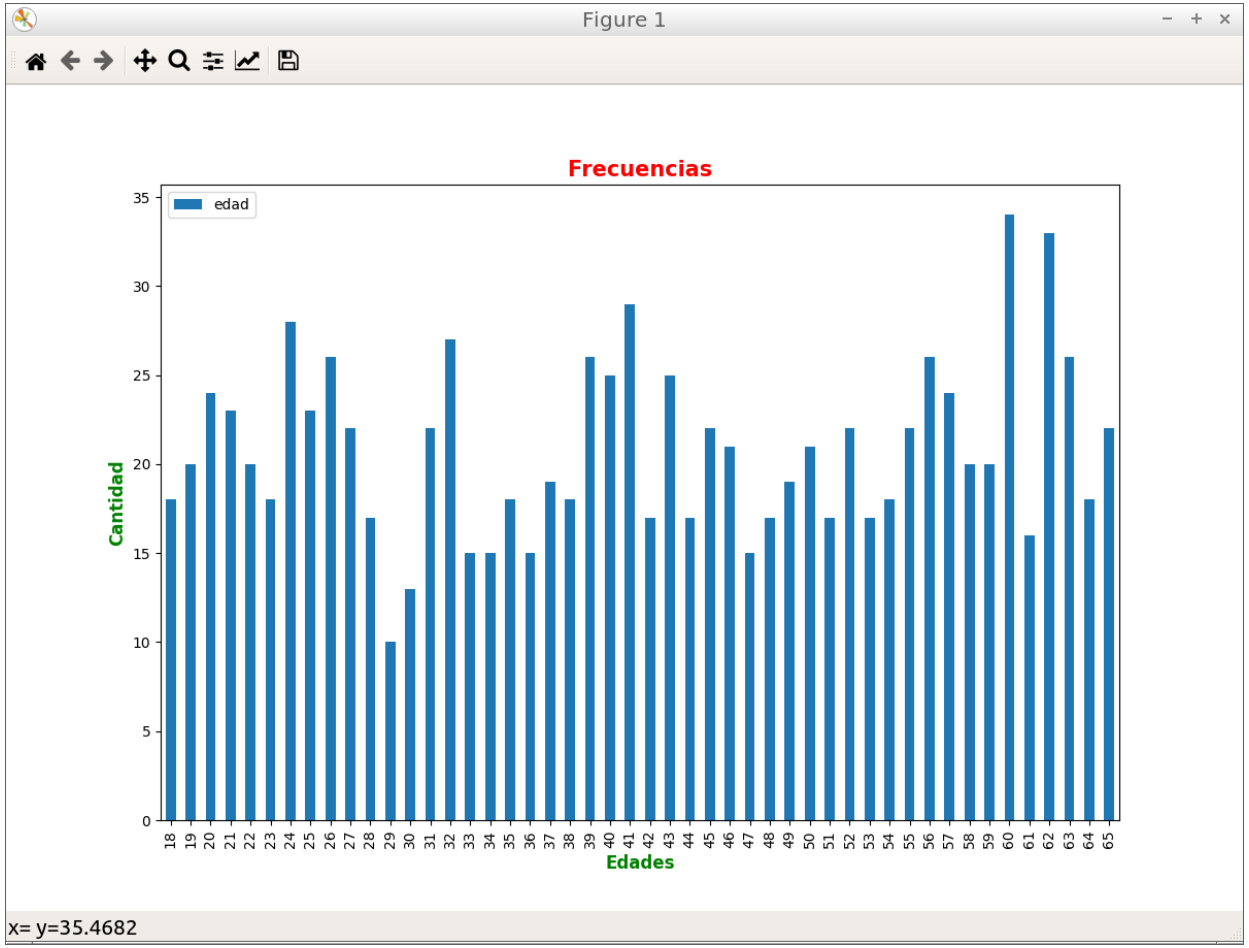
a) Indicar el número de datos que contiene el fichero.
b) Obtener una serie con cada valor y su número de ocurrencias.
c) Obtener la media de edad
d) Obtener la edad de la persona más joven y de la mayor.
e) Calcular el percentil 80
f) ¿Cuántas personas hay con más de 64 años?
g) ¿Cuántas personas no superan la edad media?

```python
import pandas as pd

datos = pd.read_csv('edades.csv', sep = ",", index_col = 'index', squeeze = True)
print("Numero de datos:", datos.count())

print(datos.value_counts().sort_index())
#datos2 = datos.value_counts()
#datos2 = datos2.sort_index()
#print(datos2)

media = datos.mean()
print("Edad Media:", media)

menor = datos.min()
mayor = datos.max()
print("Más joven:", menor, "Más Viejo:", mayor)

p80 = datos.quantile(0.8)
print("Percentil 80:", p80)

mas64 = len([x for x in datos.values if x > 64])
print("Personas con mas de 64 años:", mas64)

no_superan_media = len([x for x in datos.values if x <= media])
print("Personas que no superan la edad media:", no_superan_media)
```

```
Numero de datos: 1000
18    18
19    20
20    24
21    23
22    20
23    18
24    28
25    23
26    26
27    22
28    17
29    10
30    13
31    22
32    27
33    15
34    15
35    18
36    15
37    19
38    18
39    26
40    25
41    29
42    17
43    25
44    17
45    22
46    21
47    15
48    17
49    19
50    21
51    17
52    22
53    17
54    18
55    22
56    26
57    24
58    20
59    20
60    34
61    16
62    33
63    26
64    18
65    22
Name: edad, dtype: int64
Edad Media: 42.04
Más joven: 18 Más Viejo: 65
Percentil 80: 57.0
Personas con mas de 64 años: 22
Personas que no superan la edad media: 508
```

Ejercicio 2. Cargar los datos del fichero edades.csv en una serie de pandas y realizar un histograma con las edades. Añadir a la gráfica un título, etiquetas para ambos ejes y una leyenda.

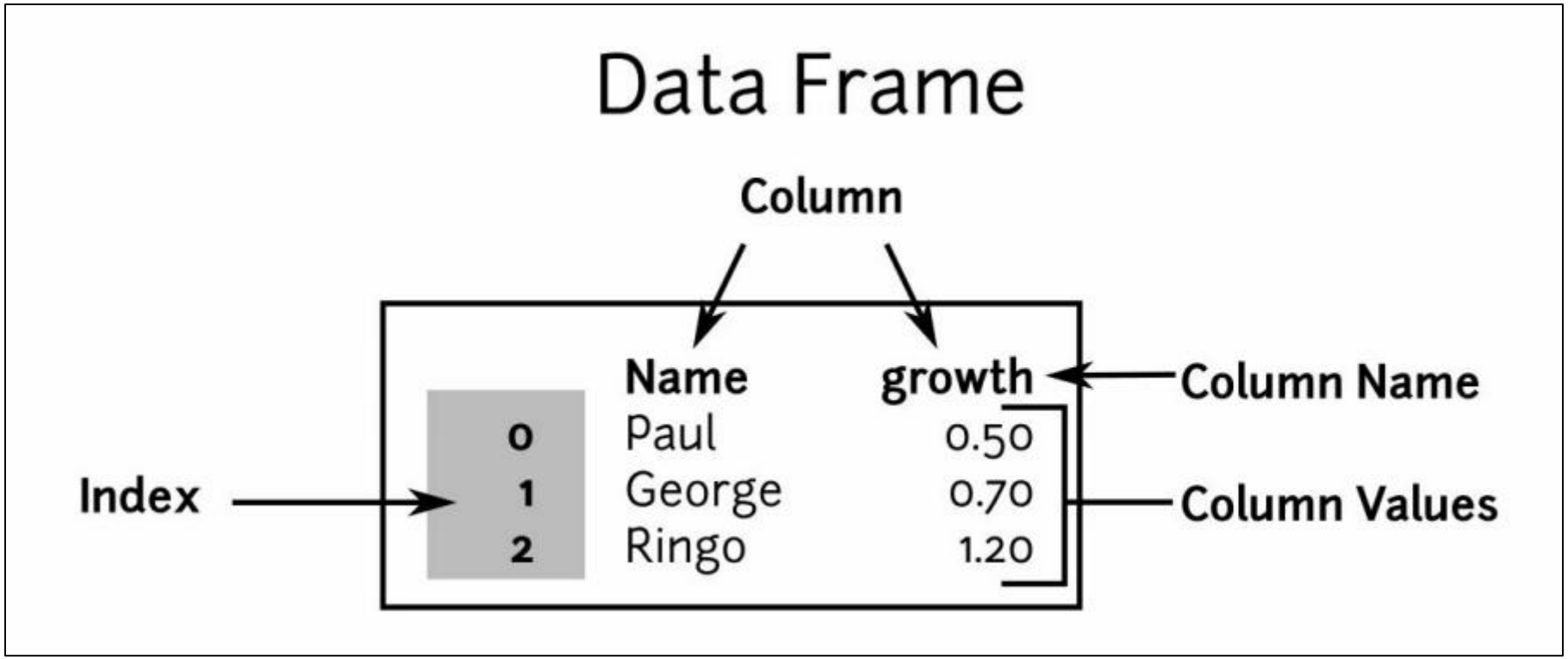
```
 9 import pandas as pd
10 import matplotlib.pyplot as plt
11
12
13
14 datos = pd.read_csv('edades.csv', sep = ",", index_col = 'index', squeeze = True)
15 datos_frecuencias = datos.value_counts().sort_index()
16
17 fig = plt.figure(figsize = (12,8))
18 ax = plt.subplot(111)
19
20 ax.set_title("Frecuencias", size = 15, color = 'r', weight = 'bold')
21 ax.set_xlabel('Edades', color = 'g', size = 12, weight = 'bold')
22 ax.set_ylabel('Cantidad', color = 'g', size = 12, weight = 'bold')
23 datos_frecuencias.plot(kind='bar', label = 'edad')
24 ax.legend(loc = 2)
```



- La estructura de datos *DataFrame* (i)

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

Un DataFrame es la extensión del objeto Series para poder gestionar datos organizados en una estructura bidimensional (tabla con varias columnas, matrices de datos, …)



- La estructura de datos *DataFrame* (ii) : creación de un DataFrame

```
 9 import pandas as pd
10
11 datos = pd.DataFrame(data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177] ,
12                       'edad': [18, 25, 56, 43, 65, 21, 37, 51]},
13                       index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
14 print(datos, "\n")
15
16 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
17 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
18 datos2 = pd.DataFrame(data, indice)
19 print(datos2, '\n')
20
21 mis_datos = [[178,18], [159,25], [165,56], [198,43], [169,65], [167,21], [178,37], [177,51]]
22 mis_columnas = ['altura', 'edad']
23 mi_indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
24
25 datos3 = pd.DataFrame(mis_datos, columns = mis_columnas, index = mi_indice)
26 print(datos3)
```

```
         altura  edad
Pedro       178    18
Ana         159    25
Paula       165    56
Antonio     198    43
Julia       169    65
Isabel      167    21
Paco        178    37
Manuel      177    51

         altura  edad
Pedro       178    18
Ana         159    25
Paula       165    56
Antonio     198    43
Julia       169    65
Isabel      167    21
Paco        178    37
Manuel      177    51

         altura  edad
Pedro       178    18
Ana         159    25
Paula       165    56
Antonio     198    43
Julia       169    65
Isabel      167    21
Paco        178    37
Manuel      177    51
```

Nota: Si no se especifica un índice, se asigna una secuencia de números enteros por defecto.

Ejercicio 3. Crear un dataframe con 3 columnas y 100 filas de datos generados aleatoriamente con la función random.randn() de Numpy. Dejar que el índice se genere automáticamente. Poner nombre a cada columna.

```
10 import pandas as pd
11 import numpy as np
12
13 matriz_datos = np.random.randn(100,3)
14 columnas = ['c1', 'c2', 'c3']
15 datos = pd.DataFrame(matriz_datos, columns = columnas)
16 print(datos)
```

```
          c1        c2        c3
0  -0.730023 -1.189768 -1.204057
1   1.669250  0.161371  0.600202
2  -2.443048  0.490735  0.075504
3  -0.884704 -0.497010  1.237328
4   1.345314 -1.073356 -0.261498
..       ...       ...       ...
95 -1.843294  0.663235 -0.194245
96  0.797250 -0.461782  0.193592
97 -0.522637  1.942518  1.564181
98  0.269190 -0.623755  0.676413
99  1.002628  0.781224 -1.536800

[100 rows x 3 columns]
```

- La estructura de datos *DataFrame* (iii): importando y exportando en ficheros csv

pd.read_csv()

```
 9 import pandas as pd
10
11 datos = pd.read_csv('zip.csv')
12 print(datos)
```

```
                 name  age    zip
0      Herbert Garza   48  91932
1     Mitchell Gomez   45  87050
2        Jacob Allen   33  12973
3      Henry Salazar   49  49466
4         Ricky Webb   53  17650
..                ...  ...    ...
995     Andre Reeves   22   9917
996   Lucas Anderson   47  16955
997    Ophelia Bryan   44  59141
998      Glenn Floyd   46  43498
999       Chris Cobb   32  15996

[1000 rows x 3 columns]
```

dataframe.to_csv()

```
 9 import pandas as pd
10
11 datos = pd.DataFrame(data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177] ,
12                             'edad': [18, 25, 56, 43, 65, 21, 37, 51]},
13                            index = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel'])
14
15 datos.to_csv('mis_datos.csv', sep  = '\t')
```

```
        altura  edad
Pedro      178 18
Ana 159 25
Paula      165 56
Antonio 198 43
Julia      169 65
Isabel     167 21
Paco       178 37
Manuel     177 51
```

- La estructura de datos *DataFrame* (iv): operaciones de acceso a valores y al índice del DataFrame

```
 9 import pandas as pd
10
11
12 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
13 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
14 datos = pd.DataFrame(data, indice)
15
16 datos.columns.values         # devuelve un array con los nombres de las columnas del dataframe
17 datos.values                 # devuelve una matriz del num_filas x num_columnas con los datos del DataFrame
18 datos ['altura']             # devuelve una serie con el indice y los valores de la columna altura
19 datos['altura'].values       # devuelve un array con los datos de la columna altura
20 datos.loc['Ana']             # devuelve una serie indexada por los nombres de las columnas con los valores del elemento cuyo indice = 'Ana'
21 datos.loc['Ana' : 'Julia']   # devuelve un dataframe con los datos de las filas comprendidas entre Ana y Julia (ambas incluidas)
22 datos.iloc[3]                # devuelve una serie indexada por los nombres de las columnas con los valores de la fila cuya posición es la 3
23 datos.iloc[3:5]              # devuelve un dataframe con las filas desde la posición 3 a la 5 (la 5 no se incluye)
24 datos.iloc[[1, 3, 5]]        # devuelve un dataframe con las filas cuyas posiciones son 1, 3, 5
25 datos.iloc[[1, 3, 5], 1]     # devuelve una serie con los datos de la columna 1 (edad) para las filas 1, 3 y 5
26 datos['edad'].values.tolist() # devuelve una lista con los datos de la columna c3
27 datos.index                  # devuelve los datos del indice en una estructura Index
28 datos.index.values           # devuelve un array con los datos del indice
29 datos.index.values.tolist()  # devuelve una lista con los datos del indice
30 datos.values[3]              # devuelve un array con los datos de la fila cuya posición es 3
31 datos[3:5]                   # devuelve un dataframe con los datos de las filas 3 a 5 (esta última no incluida)
```

- La estructura de datos *DataFrame* (v): iterando sobre filas

Podemos iterar sobre filas de muchas formas. Dos de las más usadas son las siguientes:

- Usando iterrows(). Devuelve para cada fila el valor del índice y una serie para cada dato
- Usando itertuples(). Devuelve una tupla por cada fila incluyendo el índice y los datos de las columnas

```
 9 import pandas as pd
10
11 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
12 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
13 datos = pd.DataFrame(data, indice)
14
15
16 # iterando sobre filas con iterrows()
17 for index, row in datos.iterrows():
18     print(index, row)
19 print()
20
21 # iterando sobre filas con itertuples()
22 for i in datos.itertuples():
23     print(i)
```

```
Pedro altura    178
edad         18
Name: Pedro, dtype: int64
Ana altura    159
edad         25
Name: Ana, dtype: int64
Paula altura    165
edad         56
Name: Paula, dtype: int64
Antonio altura    198
edad         43
Name: Antonio, dtype: int64
Julia altura    169
edad         65
Name: Julia, dtype: int64
Isabel altura    167
edad         21
Name: Isabel, dtype: int64
Paco altura    178
edad         37
Name: Paco, dtype: int64
Manuel altura    177
edad         51
Name: Manuel, dtype: int64

Pandas(Index='Pedro', altura=178, edad=18)
Pandas(Index='Ana', altura=159, edad=25)
Pandas(Index='Paula', altura=165, edad=56)
Pandas(Index='Antonio', altura=198, edad=43)
Pandas(Index='Julia', altura=169, edad=65)
Pandas(Index='Isabel', altura=167, edad=21)
Pandas(Index='Paco', altura=178, edad=37)
Pandas(Index='Manuel', altura=177, edad=51)
```

- La estructura de datos *DataFrame* (vi): iterando sobre columnas

Podemos iterar sobre columnas usando el atributo columns del dataframe

```
 9 import pandas as pd
10
11 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
12 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
13 datos = pd.DataFrame(data, indice)
14
15 for c in datos.columns:
16     print(c)
17
18 print()
19
20 for c in datos.columns:
21     print(datos[c])
```

```
altura
edad

Pedro       178
Ana         159
Paula       165
Antonio     198
Julia       169
Isabel      167
Paco        178
Manuel      177
Name: altura, dtype: int64
Pedro       18
Ana         25
Paula       56
Antonio     43
Julia       65
Isabel      21
Paco        37
Manuel      51
Name: edad, dtype: int64
```

- La estructura de datos *DataFrame* (vii): iterando sobre el índice

Podemos iterar sobre el índice del DataFrame usando el atributo index del DataFrame

```
 9 import pandas as pd
10
11 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
12 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
13 datos = pd.DataFrame(data, indice)
14
15 for ind in datos.index:
16     print(ind)
17
18 print()
19
20 for ind in datos.index:
21     print(datos.loc[ind])
```

```
Pedro
Ana
Paula
Antonio
Julia
Isabel
Paco
Manuel

altura    178
edad       18
Name: Pedro, dtype: int64
altura    159
edad       25
Name: Ana, dtype: int64
altura    165
edad       56
Name: Paula, dtype: int64
altura    198
edad       43
Name: Antonio, dtype: int64
altura    169
edad       65
Name: Julia, dtype: int64
altura    167
edad       21
Name: Isabel, dtype: int64
altura    178
edad       37
Name: Paco, dtype: int64
altura    177
edad       51
Name: Manuel, dtype: int64
```

- La estructura de datos *DataFrame* (viii): filtrando o seleccionando valores con condiciones

Podemos aplicar operadores lógicos a columnas y filas para seleccionar datos.

```
 9 import pandas as pd
10
11 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
12 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
13 datos = pd.DataFrame(data, indice)
14
15 datos['edad'] > 40          # devuelve una serie con el resultado booleano de la operación para la columna edad
16 datos[datos['edad'] > 40]   # devuelve un dataframe con las filas de datos para las que edad > 40
17 datos[datos.edad < 25]      # devuelve un dataframe con las filas de datos para las que edad < 25
18 datos[(datos.edad > 40) & (datos.altura > 170)] # devuelve un dataframe con las filas que cumplan ambas condiciones
19 datos[((datos.edad < 30) | (datos.edad > 50)) & (datos.altura > 170)] # menores de 30 o mayores de 50 que midan mas de 170 cm.
```

Lo más intuitivo y cómodo es utilizar la notación ***dataframe.columna*** a la hora de realizar selecciones de datos.

Ejercicio 4. Importar el fichero zip.csv a un DataFrame y realizar las siguientes operaciones:

- Seleccionar los datos de aquellas personas cuyo código postal (zip) sea superior al 98000
- Seleccionar los datos de aquellas personas cuya edad sea de al menos 60 años
- Seleccionar los datos de aquellas personas con menos de 30 años y que tengan un código postal inferior a 1000

```
 9 import pandas as pd
10
11 datos = pd.read_csv('zip.csv')
12
13 mayor98000 = datos[datos.zip > 98000]
14 print(mayor98000)
15
16 sesenta = datos[datos.age >= 60]
17 print(sesenta)
18
19 menos30y1000 = datos[(datos.age < 30) & (datos.zip < 1000)]
20 print(menos30y1000)
```

```
                name  age    zip
70     Lucinda Sherman   40  98544
92       Joel Flowers   57  98461
193    Frances Keller   43  99152
207        Brian Wade   64  99859
278  Nettie Robertson   44  99025
340    Nellie Stanley   53  99344
390   Alejandro Klein   31  98002
434    Fannie Robbins   32  99532
440       Viola Lewis   33  98729
442        Ruby Adams   62  98647
505      Hunter Klein   25  99896
524       Luis Walton   59  98423
549          Tom Pena   65  99883
603       Susie Hogan   47  99270
639        Emma Simon   54  99409
656    Blanche Greene   32  99708
770         Sue Nunez   43  99110
784     Jesus Summers   28  99361
892    Susie Phillips   61  98800
907    Christian Ryan   46  99892
940    Derrick Moreno   58  99976
970    Jimmy Martinez   53  98328
                name  age    zip
17       Linnie Francis   61  55618
18           Manuel Cox   60  88355
19        Matilda Pratt   63  17962
27    Cornelia Davidson   64  12933
39         Clara Butler   60  92193
..                  ...  ...    ...
961        Jeffrey Boyd   65   3947
968         Tillie Tran   60  75732
973  Stephen McLaughlin   60  76236
983         Jon Harvey   65  31261
989   Christian Robbins   61  19738

[141 rows x 3 columns]
               name  age  zip
251  Virginia Payne   25  696
747   Clyde Webster   21  336
```

- La estructura de datos *DataFrame* (ix): modificando valores

Una vez sabemos acceder a los datos, modificar su valor será tan sencillo como asignar el nuevo.

```
 9 import pandas as pd
10
11 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
12 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
13 datos = pd.DataFrame(data, indice)
14
15 datos.edad = datos.edad + 5
16 datos[datos['edad'] > 40] += 5
17 datos.loc[datos.edad > 50, 'altura'] = datos.altura * 1.2
18 datos.loc[datos.edad > 50, 'altura'] *= 1.2
19 datos.loc[datos.edad > 50, 'edad'] = datos.edad ** 2
```

- La estructura de datos *DataFrame* (x): borrado de filas y columnas

Podemos borrar filas y columnas con el siguiente método del dataframe:

drop(nombre, axis, inplace)

nombre: cadena de texto o una lista con los nombres de las filas o columnas a borrar.
axis: valor asignado al nombre, 0 borra filas (opción por defecto), 1 borra columnas.
inplace: True si queremos que los cambios los realice en el propio DataFrame

```
In [334]: datos
Out[334]:
         altura  edad
Pedro       178    18
Ana         159    25
Paula       165    56
Antonio     198    43
Julia       169    65
Isabel      167    21
Paco        178    37
Manuel      177    51

In [335]: datos.drop('Ana')
Out[335]:
         altura  edad
Pedro       178    18
Paula       165    56
Antonio     198    43
Julia       169    65
Isabel      167    21
Paco        178    37
Manuel      177    51
```

```
In [340]: datos
Out[340]:
         altura  edad
Pedro       178    18
Ana         159    25
Paula       165    56
Antonio     198    43
Julia       169    65
Isabel      167    21
Paco        178    37
Manuel      177    51

In [341]: datos.drop('altura', axis = 1)
Out[341]:
         edad
Pedro      18
Ana        25
Paula      56
Antonio    43
Julia      65
Isabel     21
Paco       37
Manuel     51
```

- La estructura de datos *DataFrame* (xi): añadir filas y columnas

Para añadir una fila a un dataframe tendremos que ponerla en el mismo formato dataframe y usar el método append(), indicando los datos y el índice a añadir.

Para insertar una columna en un dataframe usaremos el método insert(posición, 'nombre_columna', datos_columna)

```
 9 import pandas as pd
10 import numpy as np
11
12 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
13 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
14 datos = pd.DataFrame(data, indice)
15 print(datos,'\n')
16
17 pesos = np.random.randint(50,100,datos.shape[0])
18 datos.insert(1,'peso',pesos)
19 print(datos,'\n')
20
21 dato_anadir = {'altura' : [200] , 'peso' : [95], 'edad' : [50]}
22 indice_anadir = ['Lucas']
23 datos = datos.append(pd.DataFrame(dato_anadir,indice_anadir))
24 print(datos)
```

- La estructura de datos *DataFrame* (xii): tratamiento de valores nulos

```
import pandas as pd
import numpy as np

data = {'altura': [178, 159, np.NaN, 198, 169, 167, np.NaN, 177], 'edad': [18, np.NaN, 56, 43, 65, 21, 37, np.NaN]}
indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
datos = pd.DataFrame(data, indice)
print(datos,'\n')

datos.isnull()  # devuelve True o False para cada dato del dataframe
datos.dropna()  # devuelve un dataframe con todas las filas que no tienen ningún valor nulo
datos.fillna(1, inplace = True) # pone el vlor 1 en todos los valores nulos en el propio dataframe
datos.fillna({'altura' : datos.altura.mean(), 'edad' : datos.edad.median()}) # rellena los nulos en altura con la media y los de edad con la mediana
datos.interpolate(method = 'linear') # rellena los nulos con una interpolación, en este caso lineal
```

- La estructura de datos *DataFrame* (xiii): tratamiento de outliers

Para el tratamiento de los outliers podemos utilizar la función:

clip(lower = valor_minimo, upper = valor_maximo)

que pone a valor_minimo todos los datos por debajo de él y a valor_máximo los que están por encima.

```
In [428]: datos
Out[428]:
         altura  edad
Pedro     178.0  30.0
Ana       159.0   NaN
Paula       NaN  56.0
Antonio   198.0  43.0
Julia     169.0  65.0
Isabel    167.0  30.0
Paco        NaN  37.0
Manuel    177.0   NaN

In [429]: datos.edad = datos.edad.clip(lower = 30, upper = 60)

In [430]: datos
Out[430]:
         altura  edad
Pedro     178.0  30.0
Ana       159.0   NaN
Paula       NaN  56.0
Antonio   198.0  43.0
Julia     169.0  60.0
Isabel    167.0  30.0
Paco        NaN  37.0
Manuel    177.0   NaN
```

Ejercicio 5. Importar el fichero zip.csv a un DataFrame y realizar las siguiente operación con la función *clip()*.

- Vamos a considerar que los datos inferiores a 1000 y los superiores a 90000 son outliers.
  Por tanto, vamos a hacer que los inferiores a 1000 tomen el valor 1000 y los superiores a 90000 el valor 90000.

```
 9 import pandas as pd
10
11 datos = pd.read_csv('zip.csv')
12
13 datos.zip = datos.zip.clip(lower = 1000, upper = 90000)
14
15 print(datos[datos.zip > 90000])
16 print(datos[datos.zip < 1000])
```

Ejercicio 6. Importar el fichero test.csv a un DataFrame y realizar las siguientes operaciones:

- Eliminar la columna estado.
- Incrementar la edad de todas las personas en 3 años.
- Guardar el resultado en el fichero mitest.csv utilizando como separador el carácter '\t' (tabulador)

```
 9 import pandas as pd
10
11 datos = pd.read_csv('test.csv')
12 print(datos)
13 datos.drop('estado', axis = 1, inplace = True)
14 datos.edad += 5
15 print(datos)
16 datos.to_csv('mitest.csv', sep= '\t')
```

```
                nombre  edad     ciudad estado    zip    sexo
0       Ernest Watson    28     Abkefe     IN  83089  Female
1          Roy Clarke    44   Sutighes     IL  21455  Female
2       Beulah Torres    39   Sekirjam     GA  55780  Female
3            Ola Hill    43     Gelati     ID   7994  Female
4          Lina Walsh    25    Onimeof     MS  70221  Female
..                ...   ...        ...    ...    ...     ...
995  Verna Armstrong    41  Sigholtuk     NM  20686    Male
996    Gilbert Gross    26    Meorive     VT  35943    Male
997   Franklin Henry    47    Soihvev     NH  17773    Male
998     Celia Fuller    43   Ledjaifu     AL  91441  Female
999   Shawn Martinez    56   Ruknopif     SC  72527    Male

[1000 rows x 6 columns]
                nombre  edad     ciudad    zip    sexo
0       Ernest Watson    33     Abkefe  83089  Female
1          Roy Clarke    49   Sutighes  21455  Female
2       Beulah Torres    44   Sekirjam  55780  Female
3            Ola Hill    48     Gelati   7994  Female
4          Lina Walsh    30    Onimeof  70221  Female
..                ...   ...        ...    ...     ...
995  Verna Armstrong    46  Sigholtuk  20686    Male
996    Gilbert Gross    31    Meorive  35943    Male
997   Franklin Henry    52    Soihvev  17773    Male
998     Celia Fuller    48   Ledjaifu  91441  Female
999   Shawn Martinez    61   Ruknopif  72527    Male

[1000 rows x 5 columns]
```

- La estructura de datos *DataFrame* (xiv): ordenación de los datos

Podemos ordenar atendiendo a los valores:

dataframe.sort_values()

Podemos ordenar atendiendo al índice:

dataframe.sort_index()

```
import pandas as pd

data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
datos = pd.DataFrame(data, indice)

datos.sort_values(by = 'altura')    # ordena por altura ascendente
datos.sort_values(by = 'edad', ascending = False)   # ordena por edad descendente
datos.sort_values(by = ['altura','edad'], ascending = False)    # ordena por altura y en caso de empate por edad, en ambos casos descendente
datos.sort_values(by = ['altura','edad'], ascending = [True, False], inplace = True)    # ordena por altura ascendente, en caso de empate edad descendente
                                                                                        # inplace hace los cambios definitivos en el propio dataframe
datos.sort_index()  # ordena según el índice
```

- La estructura de datos *DataFrame* (xv): un poco de estadística descriptiva

```
 9 import pandas as pd
10
11 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
12 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
13 datos = pd.DataFrame(data, indice)
14
15 datos.count()                  # cuenta los valores de cada columna
16 datos.sum()                    # suma de los valores de cada columna
17 datos.corr()                   # coeficiente de correlación de Pearson
18 datos.corr('spearman')         # coeficiente de correlación de Spearman
19 datos.corr('kendall')          # coeficiente de correlación de Kendall
20 datos.var()                    # varianza
21 datos.cov()                    # covarianza
22 datos.prod()                   # producto de los valores de cada columna
23 datos.mean()                   # media
24 datos.max()                    # máximo
25 datos.min()                    # mínimo
26 datos.idxmax()                 # valor del indice correspondiente al máximo de cada columna
27 datos.idxmin()                 # valor del indice correspondiente al mínimo de cada columna
28 datos.quantile(0.3)            # valor del percentil indicado
29 datos.describe()               # dataframe con datos estadísticos que describen el dataframe
```

Ejercicio 7. Importar el fichero mitest.csv a un DataFrame y realizar las siguientes operaciones:

- Ordenar el dataframe por edad ascendente, por sexo primero mujeres y por zip descendente.
- Calcular la suma de las edades
- Calcular la edad media
- Sacar una estadística descriptiva del dataframe

```
 9 import pandas as pd
10
11 datos = pd.read_csv('mitest.csv', sep = '\t')
12 print(datos)
13
14 datos.sort_values(by = ['edad', 'sexo', 'zip'], ascending = [True, True, False], inplace = True)
15 print(datos)
16
17 suma_edad = datos.edad.sum()
18 edad_media = datos.edad.mean()
19
20 print("Suma de edades:", suma_edad, "Edad Media:", edad_media)
21
22 print(datos.describe())
```

```
     Unnamed: 0           nombre  edad     ciudad    zip    sexo
0             0    Ernest Watson    33     Abkefe  83089  Female
1             1       Roy Clarke    49   Sutighes  21455  Female
2             2    Beulah Torres    44   Sekirjam  55780  Female
3             3        Ola Hill    48    Gelati    7994  Female
4             4       Lina Walsh    30   Onimeof  70221  Female
..          ...              ...   ...        ...    ...     ...
995         995  Verna Armstrong    46  Sigholtuk  20686    Male
996         996    Gilbert Gross    31    Meorive  35943    Male
997         997   Franklin Henry    52    Soihvev  17773    Male
998         998     Celia Fuller    48   Ledjaifu  91441  Female
999         999   Shawn Martinez    61   Ruknopif  72527    Male

[1000 rows x 6 columns]
     Unnamed: 0            nombre  edad     ciudad    zip    sexo
704         704      Steve French    23    Kekjiuk  99937  Female
66           66     Harold Abbott    23  Ekemolime  86412  Female
7             7  Clifford Ferguson    23    Vairiha  66419  Female
57           57         Cody Ward    23   Kosadkig  62080  Female
137         137    Lucille Francis    23  Dikeluhek  48798  Female
..          ...               ...   ...        ...    ...     ...
762         762      Etta Gonzales    70   Tobokhaw  10592    Male
741         741     Mabelle Warner    70   Hovihsar   8779    Male
252         252      Bessie Berry    70     Gaikze   8553    Male
772         772     Mabelle Brooks    70   Fobefmuk   5072    Male
234         234       Marian Moody    70    Vikeeje    925    Male

[1000 rows x 6 columns]
Suma de edades: 45756 Edad Media: 45.756
       Unnamed: 0         edad           zip
count  1000.000000  1000.000000   1000.000000
mean    499.500000    45.756000  49804.367000
std     288.819436    13.922295  29221.777991
min       0.000000    23.000000    133.000000
25%     249.750000    33.000000  24783.000000
50%     499.500000    46.000000  50723.500000
75%     749.250000    58.000000  75167.750000
max     999.000000    70.000000  99949.000000
```

- La estructura de datos *DataFrame* (xvi): agrupamiendo de datos

A veces necesitamos obtener información de grupos de datos. El agrupamiento se realiza con la función

groupby(campo o campos a agrupar)

Y sobre el grupo creado podemos realizar una de las siguientes operaciones:

Nombre de la función	Descripción
`count()`	Número de valores no-NaN en el grupo
`sum()`	Suma de valores no-NaN
`mean()`	Media de valores no-NaN
`median()`	Mediana de valores no-NaN
`std(), var()`	Desviación estándar y varianza
`min(), max()`	Valores mínimo y máximo del grupo
`prod()`	Producto de valores no-NaN
`fist(), last()`	Primer y último valor no-NaN

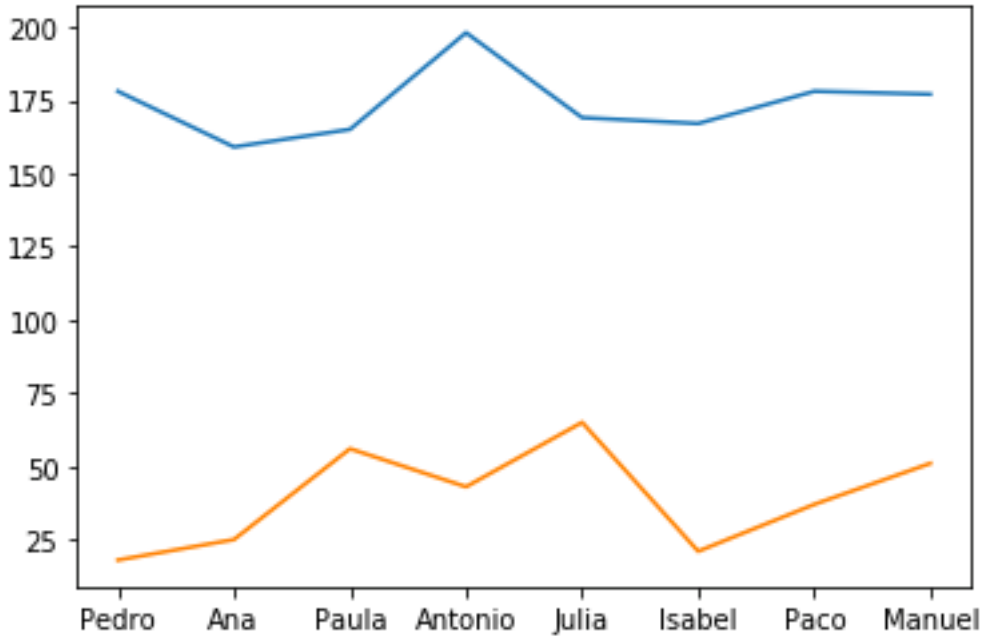
- La estructura de datos *DataFrame* (xvii): agrupamiendo de datos

```
11 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177, 156, 165, 189], 'edad': [18, 25, 56, 43, 65, 21, 37, 51, 18, 25, 18]}
12 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel', 'Pepe', 'Julio', 'Andres']
13 datos = pd.DataFrame(data, indice)
14
15 print(datos.groupby('edad').mean())
16 print(datos.groupby('edad').count())
```

```
            altura
edad
18      174.333333
21      167.000000
25      162.000000
37      178.000000
43      198.000000
51      177.000000
56      165.000000
65      169.000000
        altura
edad
18           3
21           1
25           2
37           1
43           1
51           1
56           1
65           1
```

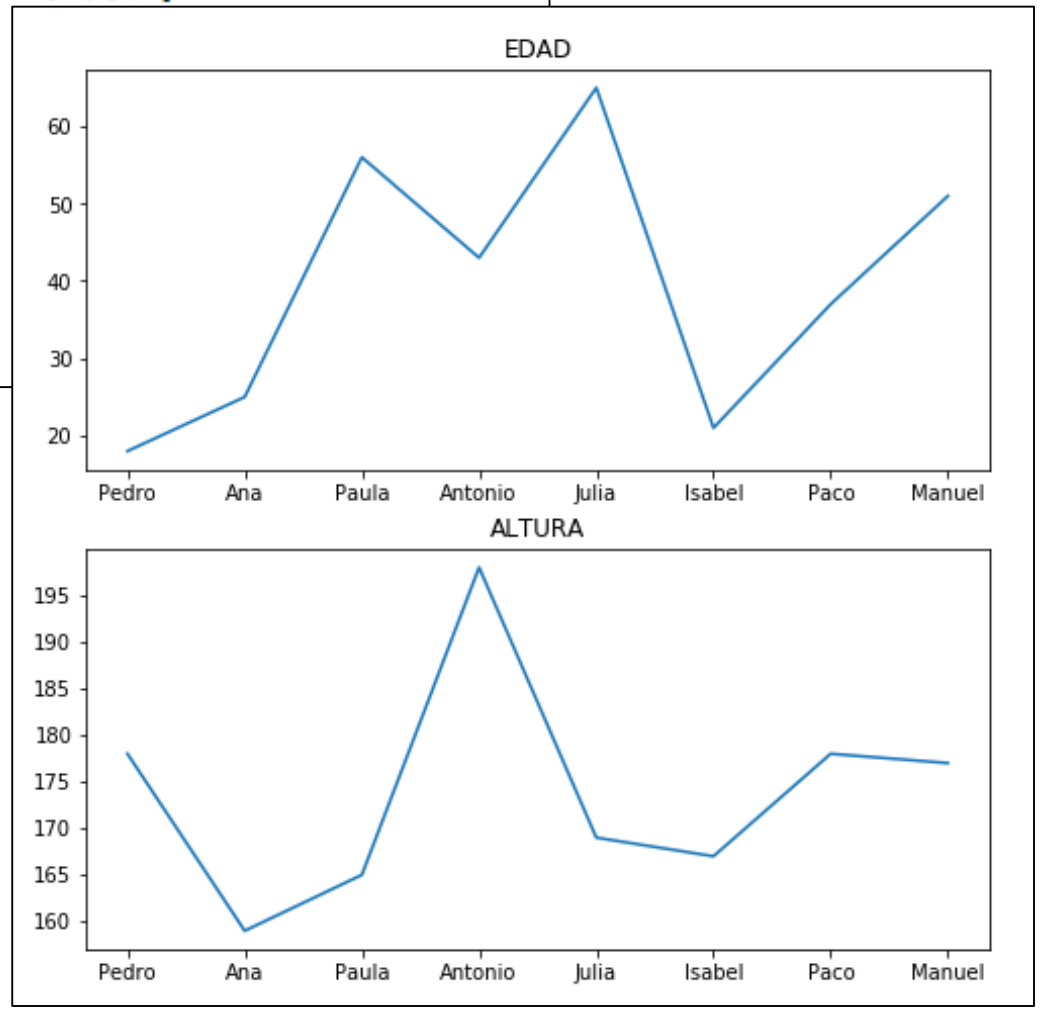
- La estructura de datos *DataFrame* (xviii): ploteado

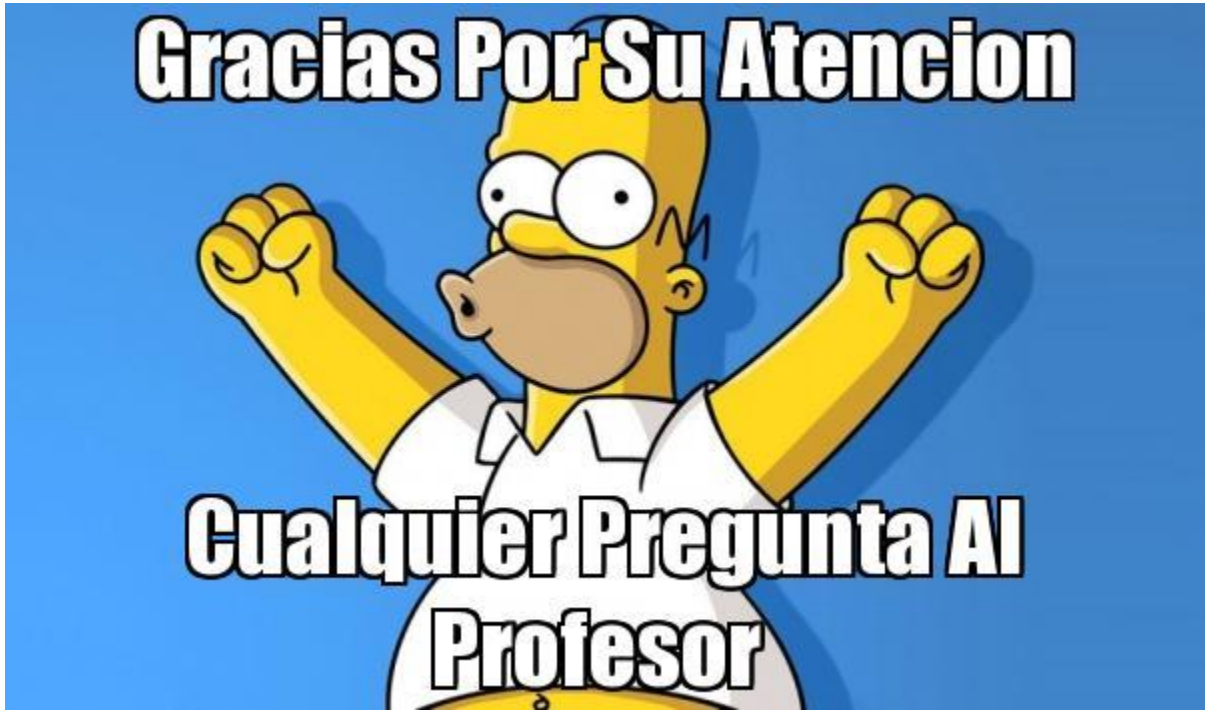
```
 9 import pandas as pd
10 import matplotlib.pyplot as plt
11
12 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
13 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
14 datos = pd.DataFrame(data, indice)
15
16 fig = plt.figure()
17 ax = plt.subplot(111)
18 ax.plot(datos)
```



- La estructura de datos *DataFrame* (xix): ploteado

```
 9 import pandas as pd
10 import matplotlib.pyplot as plt
11
12 data = {'altura': [178, 159, 165, 198, 169, 167, 178, 177], 'edad': [18, 25, 56, 43, 65, 21, 37, 51]}
13 indice = ['Pedro', 'Ana', 'Paula', 'Antonio', 'Julia', 'Isabel', 'Paco', 'Manuel']
14 datos = pd.DataFrame(data, indice)
15
16 fig = plt.figure(figsize = (8,8))
17 ax1 = plt.subplot(211)
18 ax1.plot(datos['edad'])
19 ax1.set_title('EDAD')
20
21 ax2 = plt.subplot(212)
22 ax2.plot(datos['altura'])
23 ax2.set_title('ALTURA')
```




Gracias Por Su Atencion
Cualquier Pregunta Al Profesor