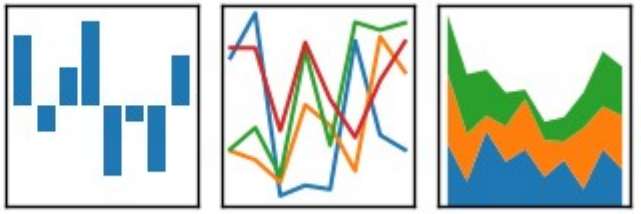
pandas

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



https://pandas.pydata.org/

Panel of Data

Part 2/2

Resum de la sessió 1

- DataFrame: taules 2D de dades. Series: taules: 1D

- Saber buscar mètodes en el manual d'usuari. Estàn agrupats per funcionalitat.

- Els índexos (tant en Series com en DataFrames) poden ser de diferents tipus i poden estar repetits

- Distingir entre la **posició** (0, 1, 2, …) d'una fila o una columna i **l'índex** (nom, índex) d'una fila o columna

- Diferenciar si treballem sobre una **vista d'un DataFrame** o sobre una **còpia** parcial o total d'un DataFrame

- Series: indexades per files `s[nom_fil], s[llista_nom_files], s[idxmin:idxmax]`

- DataFrames: indexats pel columnes `df[nom_col], df[llista_noms_col]`

- DataFrames: acés a files `df.loc[index_fila], df.iloc[posició_fila,` suporten slicing per files: `df.loc[idxmin: idxmax]`

- Accés a elements: `df.loc[fila, col], df.at[fila, col]` o
`df.iloc[pos_fila, pos_col] df.iat[pos_fila, pos_col]`

Selecció avançada

Les files en les que una o més columnes cumpleixen una condició

```
>>> df[df['Edat'] == 23]
      Nom Cognom1    Cognom2  Edat  Punts
0     Pol   Pérez    Juanola    23      5
2     Kim  Nguyen        NaN    23     12
4   Lluna  Picart  Hernández    23      8
6  Alicia   Cugat    Bunyola    23      9
7  Albert  Jordan      López    23     11

>>> df[(df['Edat'] == 23) & (df['Punts']> 10)]
      Nom Cognom1 Cognom2  Edat  Punts
2     Kim  Nguyen     NaN    23     12
7  Albert  Jordan   López    23     11

>>> df[~(df['Edat'] == 23) | (df['Punts']== 11)]
      Nom   Cognom1    Cognom2  Edat  Punts
1  Ricard  El Karim  Ullestres    24     10
3    Joan      Poll       Mató    22     10
5   Joana   Ranglas      Wayne    24      9
7  Albert    Jordan      López    23     11
8   Saïda    Blaize     Yussef    22     13
```

df[selecció op_relació valor]
amb op_relació: {>, <, >=, <=, ==, !=}

df[(selecció) op_bool (selecció)]
amb op_bool = {& (and), | (or), ~(not}

Alerta!
Cada expressió entre ()

DataFrame

Selecció avançada

Condicions sobre columnes de tipus str: utilizar els mètodes **Series.str.mètode**

```
>>> df[df['Nom'].str.startswith('A')]
      Nom Cognom1  Cognom2  Edat  Punts
0    Alba   Pérez  Juanola    23      5
6  Alicia   Cugat  Bunyola    23      9
7  Albert  Jordan    López    23     11

>>> df[(df['Nom'].str.contains('a'))
    | (df['Nom'].str.contains('A'))]
      Nom   Cognom1    Cognom2  Edat  Punts
0    Alba     Pérez    Juanola    23      5
1  Ricard  El Karim  Ullestres    24     10
3    Joan      Poll       Mató    22     10
4   Lluna    Picart  Hernández    23      8
5   Joana   Ranglas      Wayne    24      9
6  Alicia     Cugat    Bunyola    23      9
7  Albert    Jordan      López    23     11
8   Saïda    Blaize     Yussef    22     13
```

pandas.Series.str.casefold
pandas.Series.str.cat
pandas.Series.str.center
pandas.Series.str.contains
pandas.Series.str.count
pandas.Series.str.decode
pandas.Series.str.encode
pandas.Series.str.endswith
pandas.Series.str.extract
pandas.Series.str.extractall
pandas.Series.str.find
pandas.Series.str.findall
pandas.Series.str.get
pandas.Series.str.index
pandas.Series.str.join
pandas.Series.str.len
pandas.Series.str.ljust
pandas.Series.str.lower
pandas.Series.str.lstrip
pandas.Series.str.match
pandas.Series.str.normalize
pandas.Series.str.pad
pandas.Series.str.partition
pandas.Series.str.repeat
pandas.Series.str.replace
pandas.Series.str.rfind
pandas.Series.str.rindex
pandas.Series.str.rjust
pandas.Series.str.rpartition
pandas.Series.str.rstrip
pandas.Series.str.slice
pandas.Series.str.slice_replace
pandas.Series.str.split
pandas.Series.str.rsplit
pandas.Series.str.startswith
pandas.Series.str.strip
pandas.Series.str.swapcase
pandas.Series.str.title
pandas.Series.str.translate
pandas.Series.str.upper
pandas.Series.str.wrap
pandas.Series.str.zfill
pandas.Series.str.isalnum
pandas.Series.str.isalpha
pandas.Series.str.isdigit
pandas.Series.str.isspace
pandas.Series.str.islower
pandas.Series.str.isupper
pandas.Series.str.istitle
pandas.Series.str.isnumeric
pandas.Series.str.isdecimal
pandas.Series.str.get_dummies

Operacions sobre series

```
>>> df['P1']+df['P2']
0     8.0
1    12.0
2    16.0
3     NaN
4    20.0
5     NaN
6     NaN
7    12.0
8    16.0
dtype: float64

>>> df['P1'].add(df['P2'], fill_value=0)
0     8.0
1    12.0
2    16.0
3    10.0
4    20.0
5     9.0
6     9.0
7    12.0
8    16.0
dtype: float64
```

```
>>> df
     Nom   Cognom1    Cognom2  Edat  P1    P2    P3
0   Alba     Pérez    Juanola    23   5   3.0   7.0
1  Ricard  El Karim  Ullestres    24  10   2.0   NaN
2    Kim    Nguyen        NaN    23  12   4.0  12.0
3   Joan      Poll       Mató    22  10   NaN  12.0
4  Lluna    Picart  Hernández    23   8  12.0   NaN
5  Joana   Ranglas      Wayne    24   9   NaN   6.0
6  Alicia    Cugat    Bunyola    23   9   NaN   7.0
7  Albert   Jordan      López    23  11   1.0  12.0
8   Saïda    Blaize     Yussef    22  13   3.0   NaN
```

`Series.add(other, level=None, fill_value=None, axis=0)` [source]

Return Addition of series and other, element-wise (binary operator *add*).

Equivalent to `series + other`, but with support to substitute a fill_value for missing data in either one of the inputs.

Parameters: **other** : *Series or scalar value*

fill_value : *None or float value, default None (NaN)*

Fill existing missing (NaN) values, and any new element needed for successful Series alignment, with this value before computation. If data in both corresponding Series locations is missing the result of filling (at that location) will be missing.

level : *int or name*

Broadcast across a level, matching Index values on the passed MultiIndex level.

Returns: **Series**

The result of the operation.

See also

`Series.radd`

Reverse of the Addition operator, see Python documentation for more details.

Operacions sobre series

pandas.Series.add
pandas.Series.sub
pandas.Series.mul
pandas.Series.div
pandas.Series.truediv
pandas.Series.floordiv
pandas.Series.mod
pandas.Series.pow
pandas.Series.radd
pandas.Series.rsub
pandas.Series.rmul
pandas.Series.rdiv
pandas.Series.rtruediv
pandas.Series.rfloordiv
pandas.Series.rmod
pandas.Series.rpow
pandas.Series.combine
pandas.Series.combine_first
pandas.Series.round
pandas.Series.product
pandas.Series.dot

```
>>> df.loc[2:4, 'Edat']
2    23
3    22
4    23
Name: Edat, dtype: int64
>>> df.loc[2:4, 'Edat'].product()
11638
>>> df['Pr'] = df['P1']*df['P2']
>>> df['PrT'] = df['P1']*df['P2']*df['P3']
>>> df
      Nom Cognom1 Cognom2       Edat P1  P2    P3    Pr    PrT
0    Alba Pérez     Juanola     23   5   3.0   7.0   15.0   105.0
1  Ricard El Karim  Ullestres   24  10  2.0   NaN   20.0    NaN
2     Kim Nguyen    NaN         23   12  4.0  12.0   48.0   576.0
3    Joan Poll      Mató        22   10  NaN  12.0    NaN   NaN
4   Lluna Picart    Hernández   23   8   12.0  NaN   96.0   NaN
5   Joana Ranglas   Wayne       24   9   NaN   6.0    NaN   NaN
6  Alicia Cugat     Bunyola     23   9   NaN   7.0    NaN   NaN
7  Albert Jordan    López       23   11  1.0  12.0   11.0   132.0
8  Saïda  Blaize    Yussef      22   13  3.0   NaN   39.0   NaN
```

El producte de la columna Edat per les files 2 a 4 (incloses)

El producte de les columnes

Observeu com creem una nova columna

Operacions sobre DataFrames

```
>>> df2
     Nom   Cognom1   Cognom2  Edat
0  Ricard       Pla    Marset  23.0
1   Ahmed  El Karim  Alwassid  24.0
2   Júlia      Pons    Perera   NaN
3   Neson  Seelbach      Tort  25.0
4   Grace  Seelbach      Tort  23.0
>>> df+df2
         Cognom1         Cognom2   Edat        Nom   P1   P2   P3
0        PérezPla     JuanolaMarset  46.0   AlbaRicard  NaN  NaN  NaN
1  El KarimEl Karim  UllestresAlwassid  48.0  RicardAhmed  NaN  NaN  NaN
2      NguyenPons             NaN   NaN    KimJúlia  NaN  NaN  NaN
3     PollSeelbach        MatóTort  47.0   JoanNeson  NaN  NaN  NaN
4   PicartSeelbach    HernándezTort  46.0   LlunaGrace  NaN  NaN  NaN
5              NaN             NaN   NaN         NaN  NaN  NaN  NaN
6              NaN             NaN   NaN         NaN  NaN  NaN  NaN
7              NaN             NaN   NaN         NaN  NaN  NaN  NaN
8              NaN             NaN   NaN         NaN  NaN  NaN  NaN
```

Operacions de relació

pandas.DataFrame.lt

pandas.DataFrame.gt

pandas.DataFrame.le

pandas.DataFrame.ge

pandas.DataFrame.ne

pandas.DataFrame.eq

(Idem series)

```
>>> df['P1']> df['P2']
0     True
1     True
2     True
3    False
4    False
5    False
6    False
7     True
8     True
dtype: bool
```

```
>>> df['P1'].gt(df['P2'],
fill_value=0)
0     True
1     True
2     True
3     True
4    False
5     True
6     True
7     True
8     True
dtype: bool
```

Consulta avançada basada en operacions de relació.

També: df[df.P1>df.P2]

```
>>> df[df['P1'] >df['P2']]
      Nom   Cognom1    Cognom2  Edat  P1   P2    P3
0    Alba     Pérez    Juanola    23   5  3.0   7.0
1  Ricard  El Karim  Ullestres    24  10  2.0   NaN
2     Kim    Nguyen        NaN    23  12  4.0  12.0
7  Albert    Jordan      López    23  11  1.0  12.0
8   Saïda    Blaize     Yussef    22  13  3.0   NaN
```

DataFrame

- Mètodes que modifiquen o creen un nou dataframe

  Alguns mètodes tenen el paràmetre **inplace**

- Quan aquest paràmetre té valor True, el dataframe es modifica. Altrament, es crea un nou DataFrame

pandas.DataFrame.sort_values

`DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')` [source]

Sort by the values along either axis

Parameters:

by : *str or list of str*
Name or list of names to sort by.
- if *axis* is 0 or *'index'* then *by* may contain index levels and/or column labels
- if *axis* is 1 or *'columns'* then *by* may contain column levels and/or index labels

Changed in version 0.23.0: Allow specifying index or column level names.

axis : *{0 or 'index', 1 or 'columns'}, default 0*
Axis to be sorted

ascending : *bool or list of bool, default True*
Sort ascending vs. descending. Specify list for multiple sort orders. If this is a list of bools, must match the length of the by.

inplace : *bool, default False*
if True, perform operation in-place

kind : *{'quicksort', 'mergesort', 'heapsort'}, default 'quicksort'*
Choice of sorting algorithm. See also ndarray.np.sort for more information. *mergesort* is the only stable algorithm. For DataFrames, this option is only applied when sorting on a single column or label.

na_position : *{'first', 'last'}, default 'last'*
first puts NaNs at the beginning, *last* puts NaNs at the end

Returns:

sorted_obj : *DataFrame*

```
>>> df
   edat     nom  nota  tot
0    34  albert     4   38
1    67   joana     5   72
2    12   lluis     3   15
3    45     eva     2   47
>>> df.sort_values(by='edat')
   edat     nom  nota  tot
2    12   lluis     3   15
0    34  albert     4   38
3    45     eva     2   47
1    67   joana     5   72
```

DataFrame

where/mask

pandas.DataFrame.where

`DataFrame.where(cond, other=nan, inplace=False, axis=None, level=None, errors='raise', try_cast=False)` [source]

Replace values where the condition is False.

Parameters:

cond : *bool Series/DataFrame, array-like, or callable*

Where *cond* is True, keep the original value. Where False, replace with corresponding value from *other*. If *cond* is callable, it is computed on the Series/DataFrame and should return boolean Series/DataFrame or array. The callable must not change input Series/DataFrame (though pandas doesn't check it).

other : *scalar, Series/DataFrame, or callable*

Entries where *cond* is False are replaced with corresponding value from *other*. If other is callable, it is computed on the Series/DataFrame and should return scalar or Series/DataFrame. The callable must not change input Series/DataFrame (though pandas doesn't check it).

inplace : *bool, default False*

Whether to perform the operation in place on the data.

axis : *int, default None*

Alignment axis if needed.

level : *int, default None*

Alignment level if needed.

errors : *str, {'raise', 'ignore'}, default 'raise'*

Note that currently this parameter won't affect the results and will always coerce to a suitable dtype.

- 'raise' : allow exceptions to be raised.
- 'ignore' : suppress exceptions. On error return original object.

try_cast : *bool, default False*

Try to cast the result back to the input type (if possible).

Returns:

Same type as caller

```
>>> df
   edat     nom  nota  tot
0    34  albert     4   38
1    67   joana     5   72
2    12   lluis     3   15
3    45     eva     2   47
```

Tenen l'atribut inplace

Posar inplace=True per modificar df

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.where.html#pandas.DataFrame.where

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.mask.html#pandas.DataFrame.mask

DataFrame

where/mask

Modificació/Creació d'un DataFrame segons esquema:

```
if  condició:
    Mantenir valor
else:
    Modificar valor
```

Modificació/creació d'un DataFrame segons esquema:

```
if  not condició:
    Mantenir valor
else:
    Modificar valor
```

```
>>> df
   edat     nom nota  tot
0    34  albert    4   38
1    67   joana    5   72
2    12   lluis    3   15
3    45     eva    2   47
>>> df['nota'] = df['nota'].where(df.nota >3, 'SP')
>>> df
   edat     nom nota  tot
0    34  albert    4   38
1    67   joana    5   72
2    12   lluis   SP   15
3    45     eva   SP   47
>>> df['nota'] = df['nota'].mask(df.nota!='SP', 'Pass')
>>> df
   edat     nom  nota  tot
0    34  albert  Pass   38
1    67   joana  Pass   72
2    12   lluis    SP   15
3    45     eva    SP   47
```

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.where.html#pandas.DataFrame.where

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.mask.html#pandas.DataFrame.mask

Apply

El mètode apply permet d'aplicar una funció a una columna (o més)

```
>>> df['Edat'].apply(lambda x: 2*x-10)
0    36
1    38
2    36
3    34
4    36
5    38
6    36
7    36
8    34
Name: Edat, dtype: int64
```

També: `>>> df['Edat'].apply(f)` on `f` és una funció

```
>>> df['Nova'] = df.loc[3:5, 'Edat'].apply(lambda x: x-5)
>>> df
      Nom   Cognom1    Cognom2  Edat  P1    P2    P3  Nova
0    Alba     Pérez    Juanola    23   5   3.0   7.0   NaN
1  Ricard  El Karim  Ullestres    24  10   2.0   NaN   NaN
2     Kim    Nguyen        NaN    23  12   4.0  12.0   NaN
3    Joan      Poll       Mató    22  10   NaN  12.0  17.0
4   Lluna    Picart  Hernández    23   8  12.0   NaN  18.0
5   Joana   Ranglas      Wayne    24   9   NaN   6.0  19.0
6  Alicia     Cugat    Bunyola    23   9   NaN   7.0   NaN
7  Albert    Jordan      López    23  11   1.0  12.0   NaN
8   Saïda    Blaize     Yussef    22  13   3.0   NaN   NaN
```

Observeu com es crea una nova columna!

DataFrame

groupby

Subdividir dades en grups per poder fer-ne un tractament grupal

Grup = dataframe.groupby(llista de columnes)

```
>>> grups = df.groupby(df['grup'])

>>> g2.groups
{'A': Int64Index([0, 4],
dtype='int64'), 'C': Int64Index([2,
3], dtype='int64'), 'B':
Int64Index([1, 5], dtype='int64')}

>>> for name, g in grups:
...     print (name, g)
...
A   noms  edats  nota grup
0  Pol     23   2.3    A
4  Nil     35   3.5    A
B      noms  edats  nota grup
1  Andreu     34   3.4    B
5    Nora     21   7.9    B
C     noms  edats  nota grup
2  Júlia     25   5.5    C
3   Loli     12   6.2    C
```

```
>>> df
     noms  edats  nota grup
0     Pol     23   2.3    A
1  Andreu     34   3.4    B
2   Júlia     25   5.5    C
3    Loli     12   6.2    C
4     Nil     35   3.5    A
5    Nora     21   7.9    B
```

Per obtenir un grup concret:

grup.get_group(clau del grup)

```
>>> gA = g.get_group('A')
>>> gA
  noms  edats  nota grup
0  Pol     23   2.3    A
4  Nil     35   3.5    A
```

https://pandas.pydata.org/pandas-docs/stable/groupby.html

DataFrame

groupby

Calcular la mitjana, el seu valor màxim o altre d'una o més columna del grup

```
>>> grups = df.groupby(df['grup'])
>>> grups.mean()
       edats  nota
grup
A       29.0  2.90
B       27.5  5.65
C       18.5  5.85
>>> grups['edats'].max()
grup
A    35
B    34
C    25
Name: edats, dtype: int64
```

```
>>> df
     noms  edats  nota grup
0     Pol     23   2.3    A
1  Andreu     34   3.4    B
2   Júlia     25   5.5    C
3    Loli     12   6.2    C
4     Nil     35   3.5    A
5    Nora     21   7.9    B
```

La edat màxima de les persones de cada grup

https://pandas.pydata.org/pandas-docs/stable/groupby.html

DataFrame

groupby

Subdividir dades en grups per poder fer-ne un tractament grupal

Grup = dataframe.groupby(llista de columnes)

```
>>> g2 = df.groupby('edats')
>>> len(g2)
2
>>> g2.get_group(23)
    noms  edats  nota grup
0    Pol     23   2.3    A
2  Jília     23   5.5    C
3   Loli     23   6.2    C
5   Nora     23   7.0    B
>>> g2.get_group(34)
     noms  edats  nota grup
1  Andreu     34   3.4    B
4     Nil     34   3.5    A
>>> g2.mean()
       nota
edats
23     5.25
34     3.45
```

```
>>> df
     noms  edats  nota grup
0     Pol     23   2.3    A
1  Andreu     34   3.4    B
2   Júlia     23   5.5    C
3    Loli     23   6.2    C
4     Nil     34   3.5    A
5    Nora     23   7.9    B
```

La nota mitja de les persones de 23 anys és 5.25
i la de les persones de 34 anys és 3.45

https://pandas.pydata.org/pandas-docs/stable/groupby.html

DataFrame

aggregate

```
>>> grups= df.groupby('grup')
>>> df2 = grups.agg ('min')
>>> df2
        noms  edats  nota
grup
A        Nil     23   2.3
B     Andreu     23   3.4
C      Júlia     23   5.5
```

(sinònim de aggregate)

```
>>> df
     noms  edats  nota grup
0     Pol     23   2.3    A
1  Andreu     34   3.4    B
2   Júlia     23   5.5    C
3    Loli     23   6.2    C
4     Nil     34   3.5    A
5    Nora     23   7.0    B
```

df2 és un `DataFrame` (alerta grups és un `Group` no un `DataFrame`!)
Està format per la fila "mínima" de cada grup de grups, és a dir la fila corresponent als mínims de cada columna en cada grup. Per exemple en el grup A, hi ha en Pol i en Nil, el mínim dels noms és Nil, el mínim d'edat 23 i el de nota 2.3.

https://pandas.pydata.org/pandas-docs/stable/groupby.html

Groupby i Aggregations

grup.agg(diccionari amb clau=columnes i valor=operacions)

```
>>> grup.agg({'edats':max, 'nota':min})
nota  edats
grup
A      2.3     35
B      3.4     34
C      5.5     25
>>> g2.agg({'edats':[max, min], 'nota':min})
     nota edats
      min   max min
grup
A     2.3    35  23
B     3.4    34  21
C     5.5    25  12
```

Aquí, en cada grup, seleccionem el max de la columna edats i el min en la columna nota

Aquí, creem 3 columnes en el dataframe agregat: min i max d'edat de cada grup i nota mínima de cada grup.

DataFrame

merge

Es poden fusionar DataFrames indicant el criteri de fusió (columna)

```
>>> df2
      Nom    Beguda     Menjar
0  Albert  cocacola  butifarra
1  Bernat  gintonic      pizza
2   Jordi     aigua    verdura
3   Enric        vi         pa
>>> df3
      Nom  edat
0  Albert    23
1  Bernat    45
2   Jordi    89
3   Enric    67
>>> pd.merge(df2, df3, on='Nom')
      Nom    Beguda     Menjar  edat
0  Albert  cocacola  butifarra    23
1  Bernat  gintonic      pizza    45
2   Jordi     aigua    verdura    89
3   Enric        vi         pa    67
```

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.merge.html

DataFrame

reset_index

En un DataFrame multidimensional, es pot crear un nou índex numèric i fer que l'index actual esdevingui una columna. Es pot esborrar la columna o canviar-li el nom

```
>>> df = pd.DataFrame([('INFO', 'A5.0'), ('INFO', 'A5.10'), ('MEC', 'A8.1'),
('MEC', 'LAB3')], index = ['Sol', 'Coru', 'Vilaplata', 'Possal'], columns
=('Assig', 'Aula'))
>>> df
          Assig   Aula
Sol        INFO   A5.0
Coru       INFO  A5.10
Villaplata MEC   A8.1
Possal     MEC   LAB3
>>> df2 = df.reset_index()
>>> df2
       index Assig   Aula
0        Sol  INFO   A5.0
1       Coru  INFO  A5.10
2  Villaplata MEC   A8.1
3     Possal  MEC   LAB3
>>> df2.rename(columns={'index':'profe'})
       profe Assig   Aula
0        Sol  INFO   A5.0
1       Coru  INFO  A5.10
2  Villaplata MEC   A8.1
3     Possal  MEC   LAB3
```

```
>>> df3 = df2.drop(columns='Assig')
>>> df3
       index     Aula
0     Sol        A5.0
1     Coru       A5.10
2     Villaplata A8.1
3     Possal     LAB3
```

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.reset_index.html

DataFrame

merge

Es poden fusionar DataFrames indicant el criteri de fusió (columna)

```
>>> df2
      Nom    Beguda     Menjar
0  Albert  cocacola  butifarra
1  Bernat  gintonic      pizza
2   Jordi     aigua    verdura
3   Enric        vi         pa
>>> df3
      Nom  edat
0  Albert    23
1  Bernat    45
2   Jordi    89
3   Enric    67
>>> pd.merge(df2, df3, on='Nom')
      Nom    Beguda     Menjar  edat
0  Albert  cocacola  butifarra    23
1  Bernat  gintonic      pizza    45
2   Jordi     aigua    verdura    89
3   Enric        vi         pa    67
```

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.merge.html

DataFrame

stack , unstack

Es pot canviar l'estructura d'un DataFrame, pivotar-lo i canviar-ne la jerarquia.

```
>>> df
     noms  edat  nota
0  albert    12    10
1   ancil    13     3
2   pedro    14     4
3    chen    12     2
4  hassan    13     8
>>> df2 = df.stack()
>>> df2
0  noms    albert
   edat        12
   nota        10
1  noms     ancil
   edat        13
   nota         3
2  noms     pedro
   edat        14
   nota         4
3  noms      chen
   edat        12
   nota         2
4  noms    hassan
   edat        13
   nota         8
dtype: object
```

```
>>> df2.axes
[MultiIndex(levels=[[0, 1, 2, 3, 4], ['noms', 'edat',
'nota']],
           labels=[[0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3,
4, 4, 4], [0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1,
2]])]
```

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.stack.html#pandas.DataFrame.stack

Gràcies!