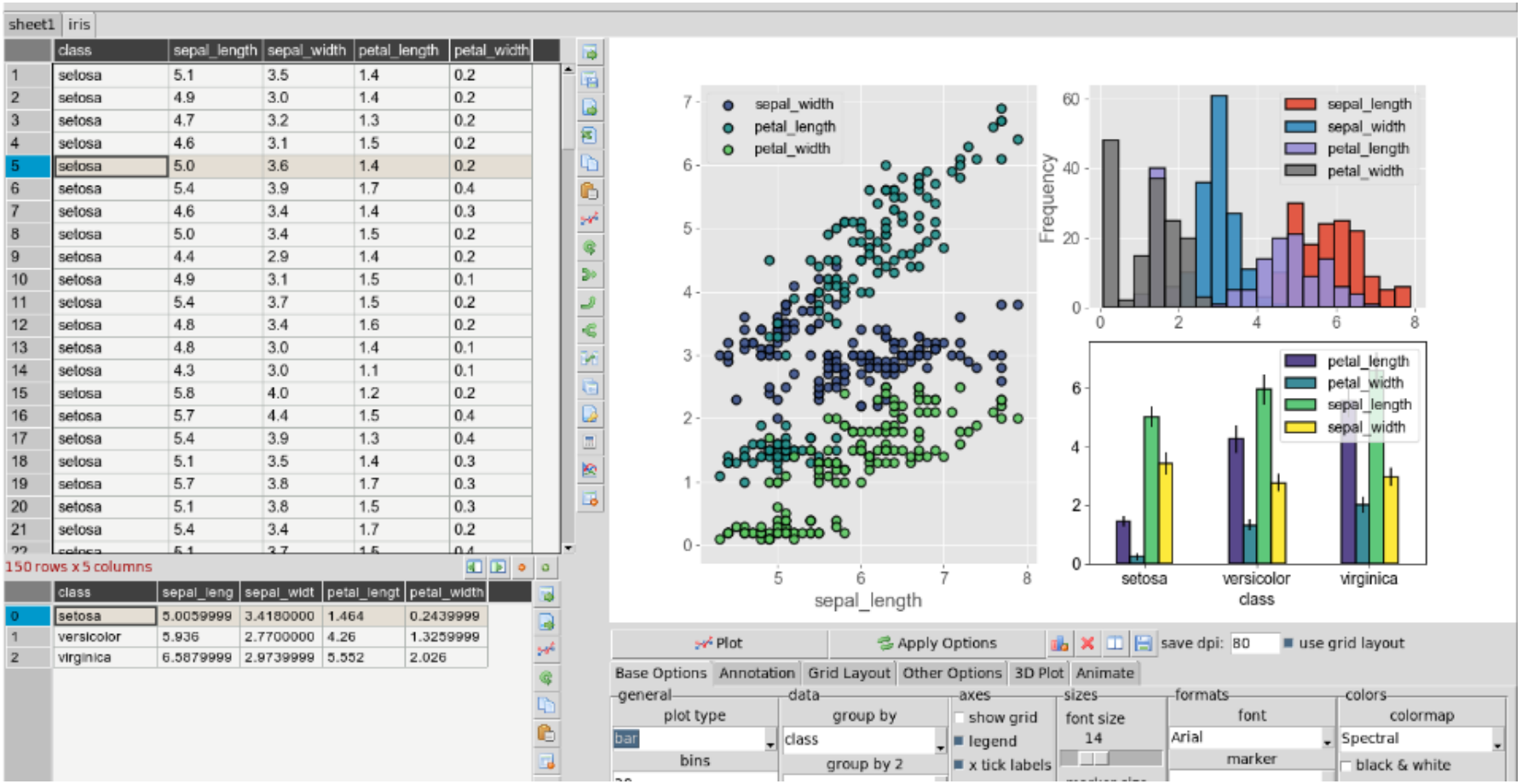
Programació amb Python

Unió Europea
Fons social europeu

Pandas : Análisis de datos



Python  Pandas

Librería para el análisis de datos
Tiene estructuras de datos que permite :

- limpiar los datos en bruto
- alinear datos para su comparación,
- fusionar conjuntos de datos,
- gestión de datos perdidos,
- cálculos estadísticos

Pandas fue diseñada originalmente para gestionar datos financieros, y como alternativo al uso de hojas de cálculo (es decir, Microsoft Excel).

La librería Pandas ofrece dos de las estructuras más usadas en Data Science: la estructura **Series** y el **DataFrame**

```
import pandas as pd
```

Series

	apples
0	3
1	2
2	0
3	1

+

Series

	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Agafa del servidor : s11_pandas.ipynb

Series

```
import pandas as pd

pd.Series([7,5,3], index = ["Ene", "Feb", "Mar"])
```

Un vector numérico. Puede tener índices numéricos o alfabéticos.

En el siguiente ejemplo, estamos creando una serie simplemente a partir de una lista:

```
In [17]: s = pd.Series([7, 5, 3])
         s

Out[17]: 0    7
         1    5
         2    3
         dtype: int64
```

Al no haberse especificado un índice, se asigna uno automáticamente con los valores 0, 1 y 2.

Si repetimos esta instrucción especificando un índice:

```
In [18]: s = pd.Series([7, 5, 3], index = ["Ene", "Feb", "Mar"])
         s

Out[18]: Ene    7
         Feb    5
```

DataFrame

El DataFrame es una estructura de datos tabular que se compone de columnas y filas ordenadas. Para que todo sea más sencillo vamos a ver un ejemplo de creación de un DataFrame (tabla) de un diccionario de listas.

El siguiente ejemplo muestra un diccionario que consta de 3 columnas o claves: Nom, Dept, DiesV (dies vacances).

```
import pandas as pd
dades = {'Nom': ['Sònia', 'Laura', 'David', 'Rosa', 'Sam'],
         'Dept' : ['PROD', 'ADMIN', 'MANT', 'ADMIN', 'PROD'],
         'DiesV' : [32, 55, 20, 43, 30]}

df = pd.DataFrame(dades)
print (df)
```

```
     Nom   Dept  DiesV
0  Sònia   PROD     32
1  Laura  ADMIN     55
2  David   MANT     20
3   Rosa  ADMIN     43
4    Sam   PROD     30
```

El método loc se puede usar de dos formas diferentes: seleccionar filas o columnas en base a una etiqueta o seleccionar filas o columnas en base a una condición.

```
df.loc[df['Nom'] == 'Rosa']
```

	Nom	Dept	DiesV
3	Rosa	ADMIN	43

```
df['Dept']
```

```
0     PROD
1    ADMIN
2     MANT
3    ADMIN
4     PROD
```

```
    Nom   Dept  DiesV
0  Sònia   PROD     32
1  Laura  ADMIN     55
2  David   MANT     20
3   Rosa  ADMIN     43
4    Sam   PROD     30
```

Pero también se puede acceder con índices numéricos que empiezan en 1.

```
print(df.loc[1])
```

```
Nom       Laura
Dept      ADMIN
DiesV        55
Name: 1, dtype: object
```

```
df[1:3]
```

	Nom	Dept	DiesV
1	Laura	ADMIN	55
2	David	MANT	20

El método iloc se utiliza en los DataFrames para seleccionar los elementos únicamente en base a su ubicación física, que empieza en cero.
Su sintaxis es data.iloc[<filas>, <columnas>]

```
df.iloc[0:5] # Primeras cinco filas
df.iloc[:, 0:5] # Primeras cinco columnas
df.iloc[[0,2,1]] # Primera, tercera y segunda filas
df.iloc[:, [0,2,1]] # Primera, tercera y segunda columnas
```

Es importante tener en cuenta que iloc devuelve una Serie Pandas cuando se selecciona una fila y un DataFrame cuando se selecciona varias. En el caso que sea necesario seleccionar un DataFrame con una única columna es necesario pasar una lista con la columna, no un escalar.

```
df[df['Dept'] == "ADMIN"]
```

	Nom	Dept	DiesV
1	Laura	ADMIN	55
3	Rosa	ADMIN	43

```
df.sort_values(by=['DiesV'], ascending=False)
```

	Nom	Dept	DiesV
1	Laura	ADMIN	55
3	Rosa	ADMIN	43
0	Sònia	PROD	32
4	Sam	PROD	30
2	David	MANT	20

```
import pandas as pd
ventas = pd.DataFrame({"A":[41,32,56,18],
                       "B":[17,54,6,78],
                       "C":[12,13,16,18] })
print (ventas)
```

```
    A   B   C
0  41  17  12
1  32  54  13
2  56   6  16
3  18  78  18
```

```
import pandas as pd
ventas = pd.DataFrame({"A":[41,32,56,18],
                       "B":[17,54,6,78],
                       "C":[12,13,16,18] },
                      index = ["Gen", "Feb", "Mar", "Abr"])
print (ventas)
```

```
     A   B   C
Gen  41  17  12
Feb  32  54  13
Mar  56   6  16
Abr  18  78  18
```

Un ejemplo de fichero para ejemplos personalizado

Queremos crear un fichero de gran tamaño que combine datos aleatorios, pero dentro de rangos lógicos.

Suponemos que son lecturas de bases metereológicas de distintos observatorios. Recogemos

- La fecha en 3 campos int : año, mes, dia
- El observatorio : que es un código pre-definido.
- El viento en kms/hora : un dato entero y su componente (código pre-definido) N,S,E,O,…
- El volumen de lluvia en litros/m2
- Y las temperaturas máxima, mínima y media de tipo float.

```
import random    as r

#
#---------------------------------------------- Definicions
#
# Els observatoris i les components del vent s'agafen aleatòriament entre
    aquests valors
obsers = ["MASNOU", "TIBIDABO", "BERGA", "VIC", "LLEIDA", "GIRONA",
    "BCN","HOSPI", "STACOL", "TARRACO"]
compos = ["N","S","E","O","NE", "NO","SE","SO"]

#---------------------------------------------- Procés principal
#



#------------------------------ 1. Obre arxiu
fich = open("observatoris.csv","w")
#------------------------------ 2. Grava capçalera
linia= "anyy;mes;dia;obser;ventkh;component;plujamm;tmax;tmin;tmitjana\n"
fich.write(linia)
```

```
#----------------------------- 3. Genera linies de dades
for i in range (1, 5000) :

 #------------------------------ anyy; mes; dia;
   linia = "{};{};{};".format(r.randint(1950,2018),
                              r.randint(1,12),
                              r.randint(1,30))
   #
   #------------------------------ obser; vent; component; plujamm;
   #
   linia += '"{}";{};"{}";{};'.format(observatoris[r.randint(0,9)],
                                      r.randint(0,200),
                                      component[r.randint(0,7)],
                                      r.randint(0,10))
   #
   #----------------------------- tmax; tmin; tmit
   #
   tempMin = round(r.uniform(-10, 20),2)
   tempMax = round(r.uniform(tempMin, 45),2)
   tempMitjana = round((tempMax-tempMin)/2,2)
   linia +="{};{};{}".format(tempMax, tempMin, tempMitjana) + "\n"
#----------------------------- 4. Tanca l'arxiu
fich.close()
```

Como puedo tratar los datos del fichero generado ?

Prueba desde Jupyter Notebook

```
import pandas as pd

df = pd.read_csv("observatoris.csv", sep=";")
df
```

Ya tenemos un dataframe de tipo pandas. Vamos a hacer consultas

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmit
0	1971	2	18	STACOL	38	NO	9	11.29	8.33	1.48
1	1968	1	18	MASNOU	48	S	6	38.41	17.46	10.47
2	2003	4	9	LLEIDA	5	S	7	20.21	-4.42	12.32
3	1961	10	21	HOSPI	200	S	3	24.31	-3.57	13.94
4	1968	2	17	BCN	144	N	1	24.75	-8.13	16.44
5	1992	10	8	BERGA	0	N	5	16.79	5.88	5.46

Mostrar el tipo de datos de cada columna

```
df.dtypes
```

head() muestra las primeras lineas del fichero

```
df.head()
```

Out[5]:

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmit
0	1971	2	18	STACOL	38	NO	9	11.29	8.33	1.48
1	1968	1	18	MASNOU	48	S	6	38.41	17.46	10.47
2	2003	4	9	LLEIDA	5	S	7	20.21	-4.42	12.32
3	1961	10	21	HOSPI	200	S	3	24.31	-3.57	13.94
4	1968	2	17	BCN	144	N	1	24.75	-8.13	16.44

```
df.tail()
```

ut[6]:

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmit
4994	2008	3	12	LLEIDA	114	N	6	7.54	-9.49	8.52
4995	1973	5	14	BCN	124	E	1	33.52	10.77	11.38
4996	1973	12	11	HOSPI	41	SO	10	20.49	2.91	8.79
4997	2014	5	24	HOSPI	143	SE	9	9.22	3.46	2.88
4998	1992	1	25	STACOL	29	NO	1	42.36	-5.30	23.83

```
In [7]: df.describe()
```

Out[7]:

	anyy	mes	dia	ventkh	plujamm	tmax	tmin	tmit
count	4999.000000	4999.000000	4999.000000	4999.000000	4999.000000	4999.000000	4999.000000	4999.000000
mean	1984.137828	6.563313	15.300660	100.344669	4.981396	25.296983	4.943527	10.176759
std	19.975886	3.471767	8.678153	57.680564	3.126686	12.398262	8.579557	6.380774
min	1950.000000	1.000000	1.000000	0.000000	0.000000	-9.920000	-10.000000	0.000000
25%	1967.000000	4.000000	8.000000	50.000000	2.000000	16.430000	-2.500000	4.905000
50%	1984.000000	7.000000	15.000000	101.000000	5.000000	26.180000	4.870000	9.570000
75%	2002.000000	10.000000	23.000000	149.000000	8.000000	35.805000	12.305000	14.690000
max	2018.000000	12.000000	30.000000	200.000000	10.000000	45.000000	20.000000	27.270000

Calcula los parámetros básicos de la muestra : recuento, media, std = desviación standard, mínimo, percentiles, máximo.

df.std(axis = 0, skipna = True)

In [8]:
```
df.T
```

Matriz traspuesta

Out[8]:

	0	1	2	3	4	5	6	7
anyy	1971	1968	2003	1961	1968	1992	1986	2013
mes	2	1	4	10	2	10	4	7
dia	18	18	9	21	17	8	22	16
obser	STACOL	MASNOU	LLEIDA	HOSPI	BCN	BERGA	TIBIDABO	BERGA
ventkh	38	48	5	200	144	0	176	66
component	NO	S	S	S	N	N	NE	SO
plujamm	9	6	7	3	1	5	10	5
tmax	11.29	38.41	20.21	24.31	24.75	16.79	31.58	21.73
tmin	8.33	17.46	-4.42	-3.57	-8.13	5.88	7.76	18.28
tmit	1.48	10.47	12.32	13.94	16.44	5.46	11.91	1.72

10 rows × 4999 columns

Pandas comandos básicos

https://www.youtube.com/watch?v=PvNKKrPE0AI

`df.shape`	Muestra la dimensión del dataframe
`df.columns`	Muestra el nombre de las columnas
`df.index`	Muestra rango de los índices

`df.head(n)`	Muestra las primeras líneas (def 5)
`df.tail(n)`	Muestra las últimas líneas (def 5)
`df.describe()`	Muestra estadísticas básicas
`df.sort_values(by='tmin')`	Clasifica por una columna
`df.T`	Transponer la matriz

Lista y tipos de columnas del dataframe `df.columns.tolist()`

Ver algunas columnas `df.get(["anyy", "mes", "tmax"])`

Ver la columna año del dataframe `df['anyy']`

La temperatura máxima, máxima del dataframe

`df['tmax'].max()`

```
df['obser'].value_counts()
```

```
TARRACO     532
MASNOU      523
TIBIDABO    517
BCN         510
STACOL      502
VIC         500
LLEIDA      490
GIRONA      483
BERGA       479
HOSPI       463
```

Selecciona y cuenta por columna

Filtrar por observatorio BCN

```
df2 = df[df['obser'] == "BCN"]
print (df2)
```

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmitjana
1	2011	10	24	BCN	138	N	6	39.77	16.68	11.55
5	1974	7	2	BCN	151	E	4	32.82	19.00	6.91
10	1954	7	8	BCN	15	SE	0	22.21	10.92	5.65
11	2006	7	15	BCN	171	S	8	14.92	4.29	5.31
35	1967	9	26	BCN	141	N	10	14.23	-8.29	11.26
37	1994	4	11	BCN	18	N	8	18.04	13.07	2.48

Concatenar comandos sobre dataframes suele generar nuevos dataframes, si se trata se selecciones de elementos.

```
resultado = df[['anyy','mes','tmax']][df['obser'] == "BCN"]
print (resultado)
```

	anyy	mes	tmax
1	2011	10	39.77
5	1974	7	32.82
10	1954	7	22.21
11	2006	7	14.92
35	1967	9	14.23
37	1994	4	18.04
42	2010	8	15.52

Si estamos realizando cálculos, el resultado puede ser un valor.

```
resultado = df['anyy'][df['obser'] == "BCN"].max()
print (resultado)
```

2018

df.sort_values(by='tmax')

```
df.sort_values(by='plujamm', ascending=False)
```

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmit
1251	1995	1	30	MASNOU	136	SO	10	29.81	3.67	13.07
424	2002	4	30	GIRONA	198	SE	10	30.83	2.72	14.05
420	2001	5	6	LLEIDA	130	S	10	34.02	17.60	8.21

```
df[:3]                    # Muestra las columnas especificadas
df[['anyy','mes']]        # Muestra las columnas especificadas
df[['anyy','mes']][:3] # cols y lins especificadas
df[df['tmin'] > 15]     # muestra registros con tmin > 15
df.groupby(['anyy']).sum()    # Agrupa por año y suma
df.groupby('anyy')['tmin'].mean() #Agrupa por año y saca
                                  #la media de la columna tmin

 # Agrupar por un criterio y obtener la media
df.groupby(['anyy', 'mes']).mean()
```

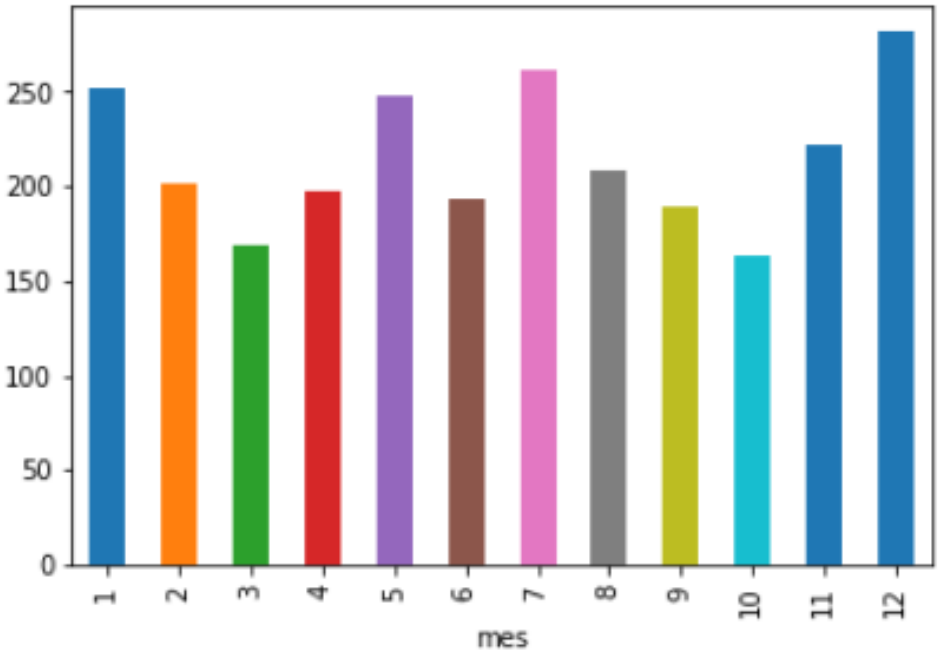
```
df['tmin'].mean()
df['tmin'].median()
df['tmin'].apply (lambda x: -x)
df['tmin'].apply (lambda x: x**2)
```

```
import matplotlib.pyplot as plt

plot_data = df[df['obser'] == 'BCN']
plot_data = plot_data.groupby('mes')['plujamm'].sum()
plot_data.plot(kind='bar')
```

Agrupar los datos del observatorio de Barcelona y agruparlo por mes, sumando el volumen de lluvia

Realizar un gráfico de barras del DataFrame.



https://pandas.pydata.org/docs/user_guide/10min.html

Práctica P01

Ejercicio 1: Dado este diccionario crea una serie de pandas (pd.Series)

```
notas={'Juan':9.0,'María':6.5,'Pablo':4.0,'Carmen':8.5,'Luis':5.0}
```

Usando la Serie calcula la nota mínima, la máxima, la media y la desviación típica.

Ejercicio 2: añade 3 personas al diccionario notas y pásalo a DataFrame.

a- Ordena las notas de menor a mayor
b- Muestra solamente la nota de Juan.
c- Haz una selección de las personas suspendidas

Práctica P02

Ejercicio 1 : Carga en tu cuaderno python el siguiente fichero : obser2021.csv

Ejercicio 2: Que hace esta programación ?

```
df2 = df[df['anyy'] > 2016]
df2 = df2[['anyy', 'mes', 'plujamm']]
df2.groupby(['anyy']).sum()
```

Ejercicio 3: Analiza los datos de los 5 últimos años con datos.

a. Localiza el observatorio con la temperatura máxima
b. Localiza el observatorio con la temperatura mínima en Febrero
c. Observatorio, día mes y año con el viento más fuerte

Ejercicio 4: Queremos saber las temperaturas máximas de cada mes en los distintos observatorios y en los distintos años.

- Cómo presentarías los datos ?
- Que criterio utilizarías para decidir cual es el año más frío ?

Vamos a estudiar los valores nulos

```
import numpy as np
#Creamos 2 valores nulos

#df['tmax'][:2] = np.nan

#y vemos como queda el dataframe
#df.isna().sum()

#df.isna().sum().sum()

#df['tmax'].fillna(2)

#df['tmax'] = df['tmax'].fillna(2)
#df.isna().sum()
```

Práctica P03. Avanzado

Cuidado con el copiar y pegar de esta web. Si te dan problemas los espacios dan problemas, pégalo a otro editor (notepad o notepad++)

Realiza algunos ejercicios de esta web.

https://aprendeconalf.es/docencia/python/ejercicios/pandas/

Best Python libraries for Machine Learning

https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/

Pandas

https://www.youtube.com/watch?v=CmorAWRsCAw

Tutorial con ejercicios

https://aprendeconalf.es/docencia/python/manual/pandas/

Pandas en 14 minutos

https://www.youtube.com/watch?v=PvNKKrPE0AI

```
# SOLUCIONS P02
# Ejercicio 1 : Carga en tu cuaderno python el siguiente
fichero : obser2021.csv

import numpy as np
import pandas as pd

df = pd.read_csv("obser2021.csv", sep=";")

# Ejercicio 2: Que hace esta programación ?

df2 = df[df['anyy'] > 2016]
df2 = df2[['anyy', 'mes', 'plujamm']]
df2.groupby(['anyy']).sum()
```

anyy	mes	plujamm
2017	519	358
2018	390	344

```
# Ejercicio 3: a) Analiza los datos de los 5 últimos años
con datos.

df3 = df.groupby(['anyy']).mean()
df3.sort_values(by=['anyy'], ascending=False)[0:5]
```

anyy	mes	dia	ventkh	plujamm	tmax	tmin	tmitjana
2018	6.393443	14.065574	105.081967	5.639344	25.508197	5.065574	10.220820
2017	6.740260	15.116883	103.753247	4.649351	23.831169	4.403766	9.714286
2016	6.352941	15.220588	100.485294	5.661765	24.116765	5.325441	9.395441
2015	6.056338	13.408451	97.000000	4.478873	24.252958	5.954930	9.149014
2014	6.454545	15.688312	97.285714	4.454545	25.570390	5.092208	10.238701

```
# Ejercicio 3: b)
# Localiza el observatorio con la temperatura máxima

df5a = df[df['anyy'] > 2013]
max = df5a ['tmax'].max()

df5a.loc[df5a['tmax'] == max]
```

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmitjana
2576	2014	11	27	TARRACO	8	N	0	44.99	17.59	13.7

```
# Ejercicio 3: c)
# Localiza el observatorio con la temperatura mínima en Febrero

df5aFeb = df5a[df5a['mes'] == 2]
min = df5aFeb ['tmin'].min()

df5aFeb.loc[df5aFeb['tmin'] == min]
```

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmitjana
2735	2017	2	26	TARRACO	89	SO	4	14.97	-9.23	12.1

```
# Ejercicio 3: d)
# Observatorio, día mes y año con el viento más fuerte

vent_max = df5a ['ventkh'].max()
df5a.loc[df5a['ventkh'] == vent_max].iloc [:,[3,2,1,0,4]]
```

	obser	dia	mes	anyy	ventkh
1266	LLEIDA	5	5	2014	200
2406	BERGA	26	8	2014	200

```
# Ejercicio 4: Queremos saber las temperaturas máximas de cada mes en los distintos observatorios y en los distintos años.
# Cómo presentarías los datos ?
# Que criterio utilizarías para decidir cual es el año más frío ?

df4a = df.groupby(['obser','anyy', 'mes']).agg({'tmax': ['max']})
df4a.tail()
```

			tmax
			max
obser	anyy	mes	
VIC	2017	4	16.37
		10	29.60
	2018	1	25.73
		2	42.04
		4	25.47

```
df4b = df.groupby(['anyy']).mean()

df4b.sort_values(by=['tmin'])[0:1]
```

anyy	mes	dia	ventkh	plujamm	tmax	tmin	tmitjana
1957	7.0	15.493333	101.68	4.4	20.940533	1.889333	9.525867