In [19]:
```
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
```

In [20]:
```
df[['Nom','DiesV']]
```

Out[20]:

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

In [21]:
```
df [1:3]
```

Out[21]:

	Nom	Dept	DiesV
1	Laura	ADMIN	55
2	David	MANT	20

In [22]:
```
print(df[1:3][['Nom', 'Dept']])
print("------------------------")
print(df[1:5]) #linees de 1 a 5-1
```

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

In [23]:
```
df[df['Nom'] == 'Rosa']
```

Out[23]:

	Nom	Dept	DiesV
3	Rosa	ADMIN	43

In [61]:
```
print(df[['Nom','Dept']])
```

```
     Nom   Dept
0  Sònia   PROD
1  Laura  ADMIN
2  David   MANT
3   Rosa  ADMIN
4    Sam   PROD
```

In [34]:
```
df.loc[df['Nom'] == 'Rosa']
```

Out[34]:

	Nom	Dept	DiesV
3	Rosa	ADMIN	43

In [27]:
```
df['Nom'] == 'Rosa'
#print(df)
```

Out[27]:
```
0    False
1    False
2    False
3     True
4    False
Name: Nom, dtype: bool
```

In [17]:
```
#amb això li dono Rosa a tots
# df['Nom'] = 'Rosa'
# print(df)
```

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

In [30]:
```
df[df['Dept'] == "ADMIN"]
```

Out[30]:

	Nom	Dept	DiesV
1	Laura	ADMIN	55
3	Rosa	ADMIN	43

In [31]:
```
df.sort_values(by=[ 'Dept', 'DiesV',], ascending=False)
```

Out[31]:

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

In [32]:
```
df['PreuDia'] = [60, 80, 90, 100, 85]
print (df)
```

```
     Nom   Dept  DiesV  PreuDia
0  Sònia   PROD     32       60
1  Laura  ADMIN     55       80
2  David   MANT     20       90
3   Rosa  ADMIN     43      100
4    Sam   PROD     30       85
```

In [33]:
```
df.loc[5] = ['Manel', 'MANT', 5, 90]
print (df)
```

```
     Nom   Dept  DiesV  PreuDia
0  Sònia   PROD     32       60
1  Laura  ADMIN     55       80
2  David   MANT     20       90
3   Rosa  ADMIN     43      100
4    Sam   PROD     30       85
5  Manel   MANT      5       90
```

In [34]:
```
df=df.append({'Nom':'Sara' , 'Dept':'VENDES', 'DiesV':8, 'PreuDia':44} , ignore_ind
```

```
C:\Users\Alumne_mati1\AppData\Local\Temp\ipykernel_9796\3603459778.py:1: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  df=df.append({'Nom':'Sara' , 'Dept':'VENDES', 'DiesV':8, 'PreuDia':44} , ignore_i
ndex=True)
```

In [35]:
```
print(df)
```

```
     Nom    Dept  DiesV  PreuDia
0  Sònia    PROD     32       60
1  Laura   ADMIN     55       80
2  David    MANT     20       90
3   Rosa   ADMIN     43      100
4    Sam    PROD     30       85
5  Manel    MANT      5       90
6   Sara  VENDES      8       44
```

In [36]:
```
df.describe()
```

Out[36]:

	DiesV	PreuDia
count	7.000000	7.000000
mean	27.571429	78.428571
std	18.100250	19.594703
min	5.000000	44.000000
25%	14.000000	70.000000
50%	30.000000	85.000000
75%	37.500000	90.000000
max	55.000000	100.000000

In [40]:
```
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"])
#si no especifiquem index doncs fa numeros incrementals
print (ventas)
print("SUMA")
print(ventas.sum(axis=1))
#axis=0 canvia el resultat
print(ventas.describe())
```

```
      A   B   C
Gen  41  17  12
Feb  32  54  13
Mar  56   6  16
Abr  18  78  18
SUMA
Gen     70
Feb     99
Mar     78
Abr    114
dtype: int64
                A          B          C
count   4.000000   4.000000   4.000000
mean   36.750000  38.750000  14.750000
std    15.945219  33.260337   2.753785
min    18.000000   6.000000  12.000000
25%    28.500000  14.250000  12.750000
50%    36.500000  35.500000  14.500000
75%    44.750000  60.000000  16.500000
max    56.000000  78.000000  18.000000
```

In [60]:
```
# PRACTICA 01
# Ejercicio 1: Añade al dataframe df : 1 persona para cada departamento
# con 5, 7 y 9 días de vacaciones.
# Muestra la lista de empleados ordenados por Dept, DiesV.
# Ejercicio 2: Dado este diccionario crea un DataFrame :
# notas={'Juan':9.0,'María':6.5,'Pablo':4.0,'Carmen':8.5,'Luis':5.0}
# Calcula la nota mínima, la máxima, la media y la desviación típica.
# Ejercicio 3: añade 3 personas al diccionario notas.
# a- Ordena las notas de menor a mayor
# b- Muestra solamente la nota de Juan.
# c- Haz una selección de las personas suspendidas
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
```

In [61]:
```
#df.loc[5] = ['Manel', 'MANT', 5, 90]

# df=df.append({'Nom':'Pere' , 'Dept':'PROD', 'DiesV':5},  ignore_index=True)
# df=df.append({'Nom':'Marta' , 'Dept':'MANT', 'DiesV':7} , ignore_index=True)
# df=df.append({'Nom':'Joana' , 'Dept':'ADMIN', 'DiesV':9} , ignore_index=True)

#print(df)

#amb aquest evitem l'error tot hi que amb error inclos el de dalt al fer print tb h

df3 = pd.DataFrame([['Pere', 'PROD', '5'], ['Marta', 'MANT', '7'], ['Joana', 'ADMIN

print(pd.concat([df,df3], ignore_index=True))
```

```
     Nom   Dept DiesV
0  Sònia   PROD    32
1  Laura  ADMIN    55
2  David   MANT    20
3   Rosa  ADMIN    43
4    Sam   PROD    30
5   Pere   PROD     5
6  Marta   MANT     7
7  Joana  ADMIN     9
```

In [62]:
```
# Muestra la lista de empleados ordenados por Dept, DiesV.
# Ejercicio 2: Dado este diccionario crea un DataFrame :
# notas={'Juan':9.0,'María':6.5,'Pablo':4.0,'Carmen':8.5,'Luis':5.0}
df.sort_values(by=['DiesV','Dept'], ascending=False)
```

Out[62]:

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

In [63]:
```
notas = {'Juan':9.0,'María':6.5,'Pablo':4.0,'Carmen':8.5,'Luis':5.0}
df3 = pd.DataFrame([[key, notas[key]] for key in notas.keys()], columns=['Nom', 'Di
df3
```

Out[63]:

	Nom	DiesV
0	Juan	9.0
1	María	6.5
2	Pablo	4.0
3	Carmen	8.5
4	Luis	5.0

In [64]:
```
# Calcula la nota mínima, la máxima, la media y la desviación típica.
df.describe()
```

Out[64]:

	DiesV
count	5.000000
mean	36.000000
std	13.397761
min	20.000000
25%	30.000000
50%	32.000000
75%	43.000000
max	55.000000

In [ ]:

In [ ]:

In [65]:
```
import numpy as np
import pandas as pd

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

Out[65]:

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmitjana
0	1973	12	9	TARRACO	191	NO	10	21.22	-2.45	11.83
1	2011	10	24	BCN	138	N	6	39.77	16.68	11.55
2	1998	6	6	VIC	192	NE	0	19.26	15.06	2.10
3	1987	1	12	MASNOU	36	NE	10	29.64	13.40	8.12
4	1954	2	10	STACOL	114	S	0	30.56	1.87	14.34
...	...	...	...	...	...	...	...	...	...	...
4994	1980	3	9	BCN	18	N	3	-0.79	-8.28	3.74
4995	1969	11	26	BERGA	91	SE	10	10.22	-7.35	8.79
4996	2015	6	20	LLEIDA	199	NE	3	17.22	-5.55	11.38
4997	2011	10	12	GIRONA	114	SO	2	39.17	9.78	14.70
4998	1951	11	11	GIRONA	157	S	2	14.19	13.69	0.25

4999 rows × 10 columns

In [ ]:
```
# df[df['obser'] == 'BCN']

# df.loc[df.obser == 'BCN'].loc[df.component == 'N'].loc[df.plujamm > 8]
```

In [66]:
```
df.describe()
```

Out[66]:

	anyy	mes	dia	ventkh	plujamm	tmax	tmin
count	4999.000000	4999.000000	4999.000000	4999.000000	4999.000000	4999.000000	4999.000000
mean	1984.016403	6.495899	15.434687	100.480896	5.039808	25.065465	5.041690
std	19.934309	3.473148	8.731287	58.143425	3.149124	12.736527	8.670017
min	1950.000000	1.000000	1.000000	0.000000	0.000000	-9.500000	-9.990000
25%	1967.000000	4.000000	8.000000	51.000000	2.000000	16.245000	-2.480000
50%	1984.000000	6.000000	15.000000	99.000000	5.000000	26.040000	4.770000
75%	2001.000000	10.000000	23.000000	151.000000	8.000000	35.595000	12.755000
max	2018.000000	12.000000	30.000000	200.000000	10.000000	44.990000	20.000000

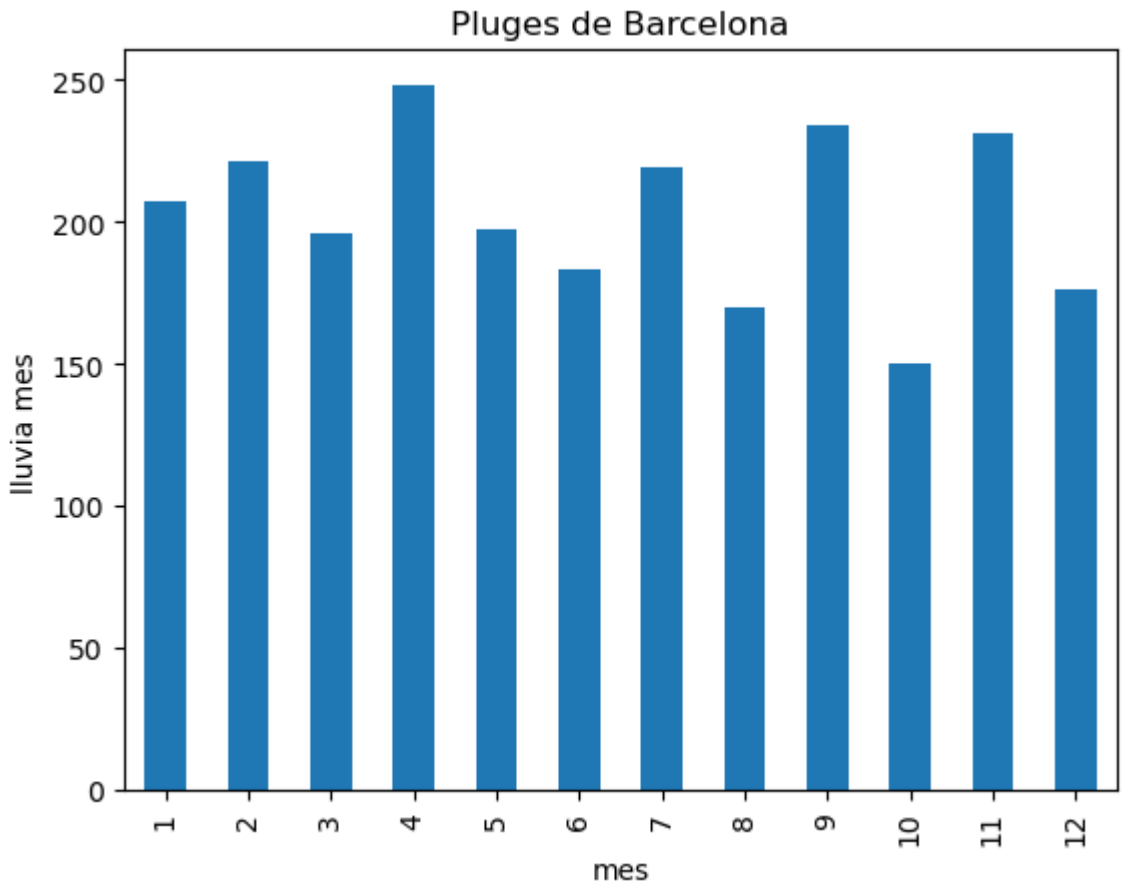
In [67]:
```
df[df['obser'] == 'BCN']
#df.loc[df.obser == 'BCN'].loc[df.component == 'N'].loc[df.plujamm > 8]

df.loc[(df.obser == 'BCN') & (df.component == 'N') & (df.plujamm > 8)]

# df_bcn = df.loc[(df.obser == 'BCN') & (df.component == 'N') & (df.plujamm > 8)]
```

Out[67]:

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmitjana
35	1967	9	26	BCN	141	N	10	14.23	-8.29	11.26
480	1998	11	26	BCN	55	N	9	25.14	1.55	11.79
850	2012	6	20	BCN	120	N	9	22.25	-6.16	14.21
1171	1998	9	22	BCN	63	N	10	17.51	10.07	3.72
1253	1956	8	29	BCN	129	N	9	44.52	-4.03	24.28
2284	2005	12	9	BCN	189	N	9	37.99	18.67	9.66
2450	1953	10	13	BCN	45	N	9	44.00	18.52	12.74
2789	1965	4	10	BCN	176	N	9	43.29	0.37	21.46
4346	1992	12	27	BCN	72	N	9	14.99	-0.11	7.55
4432	1962	1	17	BCN	172	N	9	16.38	-2.77	9.57
4565	1990	2	20	BCN	41	N	10	19.73	-5.28	12.51
4802	1985	3	29	BCN	70	N	9	35.35	15.34	10.01

In [71]:
```
plot_data = df[df['obser'] == 'BCN']

plot_data = plot_data.groupby('mes')['plujamm'].sum()

#aquesta part substitueix a l'altre comentada
ax = plot_data.plot(kind='bar', title='Pluges de Barcelona')
ax.set_xlabel('mes')
ax.set_ylabel('lluvia mes')

#plot_data.plot(kind='bar')
```

Out[71]:
```
Text(0, 0.5, 'lluvia mes')
```



In [81]:
```
df2 = df[df['anyy'] > 2016]
df2 = df2[['anyy', 'mes', 'plujamm']]
# df2.groupby(['anyy']).sum()
df2.groupby(['anyy', 'mes'])["plujamm"].sum()

#print(df2)
```

Out[81]:

```
anyy  mes
2017  1      45
      2      13
      3      37
      4      21
      5      30
      6       8
      7      39
      8      32
      9      31
      10     40
      11     31
      12     31
2018  1      38
      2      34
      3      16
      4      31
      5      45
      6       0
      7       9
      8      53
      9       6
      10     45
      11     31
      12     36
Name: plujamm, dtype: int64
```

In [99]:
```
# Practica P01.
# Ejercicio 1 : Carga en tu cuaderno python el siguiente fichero : obser2021.csv

import numpy as np
import pandas as pd

df = pd.read_csv(r".\dat\obser2021.csv", sep=";")
df
```

Out[99]:

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmitjana
0	1973	12	9	TARRACO	191	NO	10	21.22	-2.45	11.83
1	2011	10	24	BCN	138	N	6	39.77	16.68	11.55
2	1998	6	6	VIC	192	NE	0	19.26	15.06	2.10
3	1987	1	12	MASNOU	36	NE	10	29.64	13.40	8.12
4	1954	2	10	STACOL	114	S	0	30.56	1.87	14.34
...	...	...	...	...	...	...	...	...	...	...
4994	1980	3	9	BCN	18	N	3	-0.79	-8.28	3.74
4995	1969	11	26	BERGA	91	SE	10	10.22	-7.35	8.79
4996	2015	6	20	LLEIDA	199	NE	3	17.22	-5.55	11.38
4997	2011	10	12	GIRONA	114	SO	2	39.17	9.78	14.70
4998	1951	11	11	GIRONA	157	S	2	14.19	13.69	0.25

4999 rows × 10 columns

In [100…
```
df[['dia', 'ventkh', 'component']]
```

Out[100]:

	dia	ventkh	component
0	9	191	NO
1	24	138	N
2	6	192	NE
3	12	36	NE
4	10	114	S
…	…	…	…
4994	9	18	N
4995	26	91	SE
4996	20	199	NE
4997	12	114	SO
4998	11	157	S

4999 rows × 3 columns

In [101…
```
# Ejercicio 2: Que hace esta programación ?

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

Out[101]:

anyy	mes	plujamm
2017	**1**	45
	2	13
	3	37
	4	21
	5	30
	6	8
	7	39
	8	32
	9	31
	10	40
	11	31
	12	31
2018	**1**	38
	2	34
	3	16
	4	31
	5	45
	6	0
	7	9
	8	53
	9	6
	10	45
	11	31
	12	36

In [123…

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

maximo = df['anyy'].max() - 5
df5a = df [df['anyy'] > maximo ]
#print (df5a)
df5a
```

```
      anyy  mes  dia    obser  ventkh component  plujamm   tmax   tmin  \
13    2015    9    5  TARRACO      36        SO        0  32.02   3.89
40    2018    8    6   STACOL     185        SE        9  38.81  17.64
44    2015    8   12   LLEIDA      56         S        9  42.55  16.68
76    2014   11   10   LLEIDA     157        SO        1  30.98  -5.63
101   2015    4   23   MASNOU     194        SO        8  30.75  10.41
...    ...  ...  ...      ...     ...       ...      ...    ...    ...
4920  2018    8   24    BERGA     147         S        4  41.90   7.05
4932  2014    2    7    HOSPI     107        NO        0  30.41   1.68
4938  2017   11    8   STACOL     113        NO        1  40.15  -6.47
4980  2015   11   23   LLEIDA     113        NO        5  39.88  15.92
4996  2015    6   20   LLEIDA     199        NE        3  17.22  -5.55

      tmitjana
13       14.07
40       10.59
44       12.93
76       18.30
101      10.17
...        ...
4920     17.43
4932     14.37
4938     23.31
4980     11.98
4996     11.38

[354 rows x 10 columns]
```

Out[123]:

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmitjana
13	2015	9	5	TARRACO	36	SO	0	32.02	3.89	14.07
40	2018	8	6	STACOL	185	SE	9	38.81	17.64	10.59
44	2015	8	12	LLEIDA	56	S	9	42.55	16.68	12.93
76	2014	11	10	LLEIDA	157	SO	1	30.98	-5.63	18.30
101	2015	4	23	MASNOU	194	SO	8	30.75	10.41	10.17
...	...	...	...	...	...	...	...	...	...	...
4920	2018	8	24	BERGA	147	S	4	41.90	7.05	17.43
4932	2014	2	7	HOSPI	107	NO	0	30.41	1.68	14.37
4938	2017	11	8	STACOL	113	NO	1	40.15	-6.47	23.31
4980	2015	11	23	LLEIDA	113	NO	5	39.88	15.92	11.98
4996	2015	6	20	LLEIDA	199	NE	3	17.22	-5.55	11.38

354 rows × 10 columns

In [169…
```
# Ejercicio 3: b)
# Localiza el observatorio con la temperatura máxima

# df2.groupby(['anyy', 'mes']).sum()---------------df5a [df5a['obser'] and df5a['tm
#df5ot = df5a.groupby(["obser",'tmax'])

# el meu sense filtrar--------------print(df5a[df5a['tmax'] == df5a['tmax'].max()])
df5a[df5a['tmax'] == df5a['tmax'].max()]
```

Out[169]:

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

In [167…
```
# Ejercicio 3: c)
# Localiza el observatorio con la temperatura mínima en Febrero

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

Out[167]:

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

In [170…
```
# Ejercicio 3: d)
# Observatorio, día mes y año con el viento más fuerte
df5a [df5a['ventkh'] == df5a['ventkh'].max()]
```

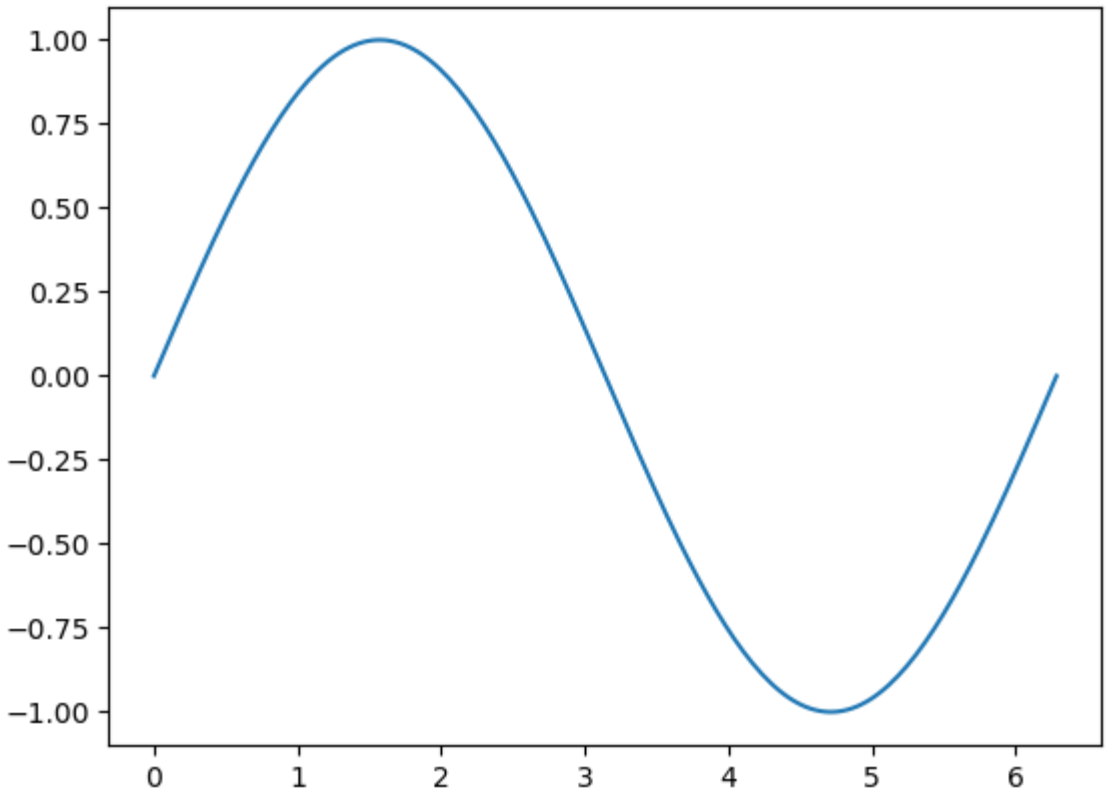
Out[170]:

	anyy	mes	dia	obser	ventkh	component	plujamm	tmax	tmin	tmitjana
1266	2014	5	5	LLEIDA	200	NO	10	44.35	18.56	12.90
2406	2014	8	26	BERGA	200	SE	2	25.40	8.09	8.65

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

In [ ]:
```
# Cómo presentarías los datos ?  --> DEMASIADOS DATOS DECIDIR ENTRE ANALIZAR POR OB
```

In [171…
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 2 * np.pi, 200)
y = np.sin(x)

fig, ax = plt.subplots()
ax.plot(x, y)
plt.show()
```

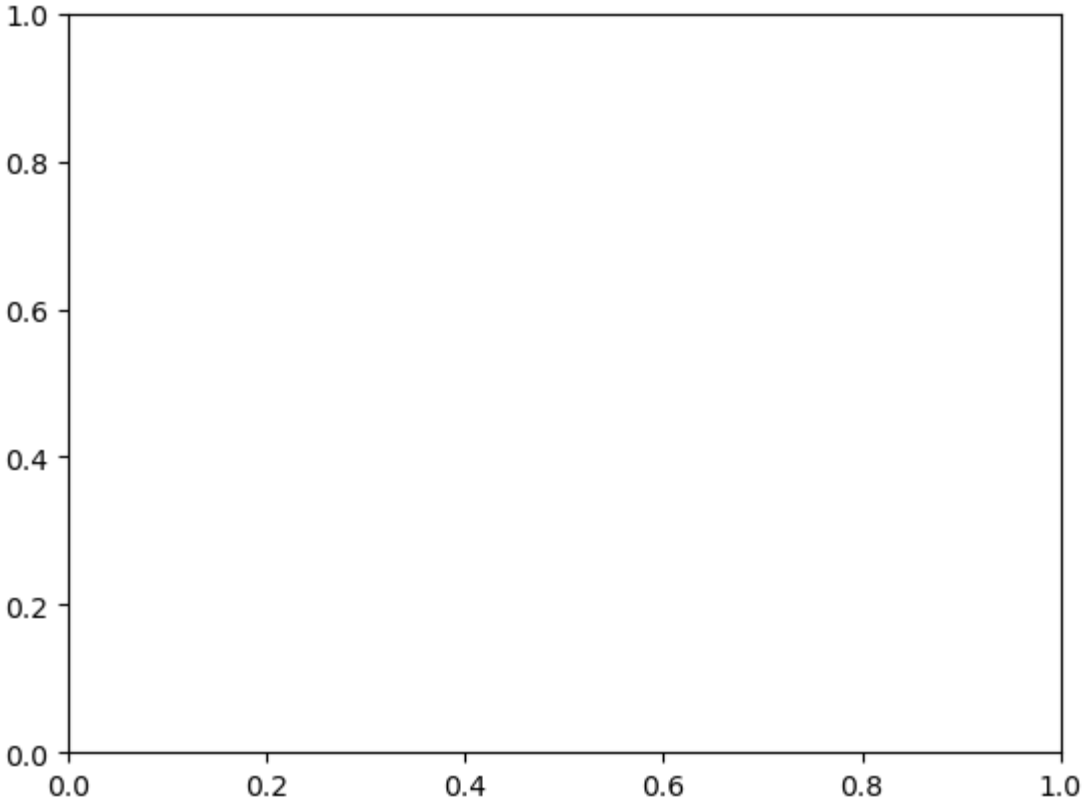


In [174…

```
fig, df5a  = plt.subplots()  # Create a figure containing a single axes.
ax.plot((['anyy'], ['tmax']))  # Plot some data on the axes.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[174], line 2
      1 fig, df5a  = plt.subplots()  # Create a figure containing a single axes.
----> 2 ax.plot((['anyy'], ['tmax']))

File ~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py:1688, in Axes.plot(sel
f, scalex, scaley, data, *args, **kwargs)
   1445 """
   1446 Plot y versus x as lines and/or markers.
   1447
   (...)
   1685 (``'green'``) or hex strings (``'#008000'``).
   1686 """
   1687 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1688 lines = [*self._get_lines(*args, data=data, **kwargs)]
   1689 for line in lines:
   1690     self.add_line(line)

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:311, in _process_plot_v
ar_args.__call__(self, data, *args, **kwargs)
    309     this += args[0],
    310     args = args[1:]
--> 311 yield from self._plot_args(
    312     this, kwargs, ambiguous_fmt_datakey=ambiguous_fmt_datakey)

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:501, in _process_plot_v
ar_args._plot_args(self, tup, kwargs, return_kwargs, ambiguous_fmt_datakey)
    499     self.axes.xaxis.update_units(x)
    500 if self.axes.yaxis is not None:
--> 501     self.axes.yaxis.update_units(y)
    503 if x.shape[0] != y.shape[0]:
    504     raise ValueError(f"x and y must have same first dimension, but "
    505                      f"have shapes {x.shape} and {y.shape}")

File ~\anaconda3\lib\site-packages\matplotlib\axis.py:1670, in Axis.update_units(se
lf, data)
   1668 neednew = self.converter != converter
   1669 self.converter = converter
-> 1670 default = self.converter.default_units(data, self)
   1671 if default is not None and self.units is None:
   1672     self.set_units(default)

File ~\anaconda3\lib\site-packages\matplotlib\category.py:105, in StrCategoryConver
ter.default_units(data, axis)
    103 # the conversion call stack is default_units -> axis_info -> convert
    104 if axis.units is None:
--> 105     axis.set_units(UnitData(data))
    106 else:
    107     axis.units.update(data)

File ~\anaconda3\lib\site-packages\matplotlib\category.py:181, in UnitData.__init__
(self, data)
    179 self._counter = itertools.count()
    180 if data is not None:
--> 181     self.update(data)

File ~\anaconda3\lib\site-packages\matplotlib\category.py:214, in UnitData.update(s
```

```
elf, data)
    212 # check if convertible to number:
    213 convertible = True
--> 214 for val in OrderedDict.fromkeys(data):
    215     # OrderedDict just iterates over unique values in data.
    216     _api.check_isinstance((str, bytes), value=val)
    217     if convertible:
    218         # this will only be called so long as convertible is True.

TypeError: unhashable type: 'numpy.ndarray'
```



In [ ]: