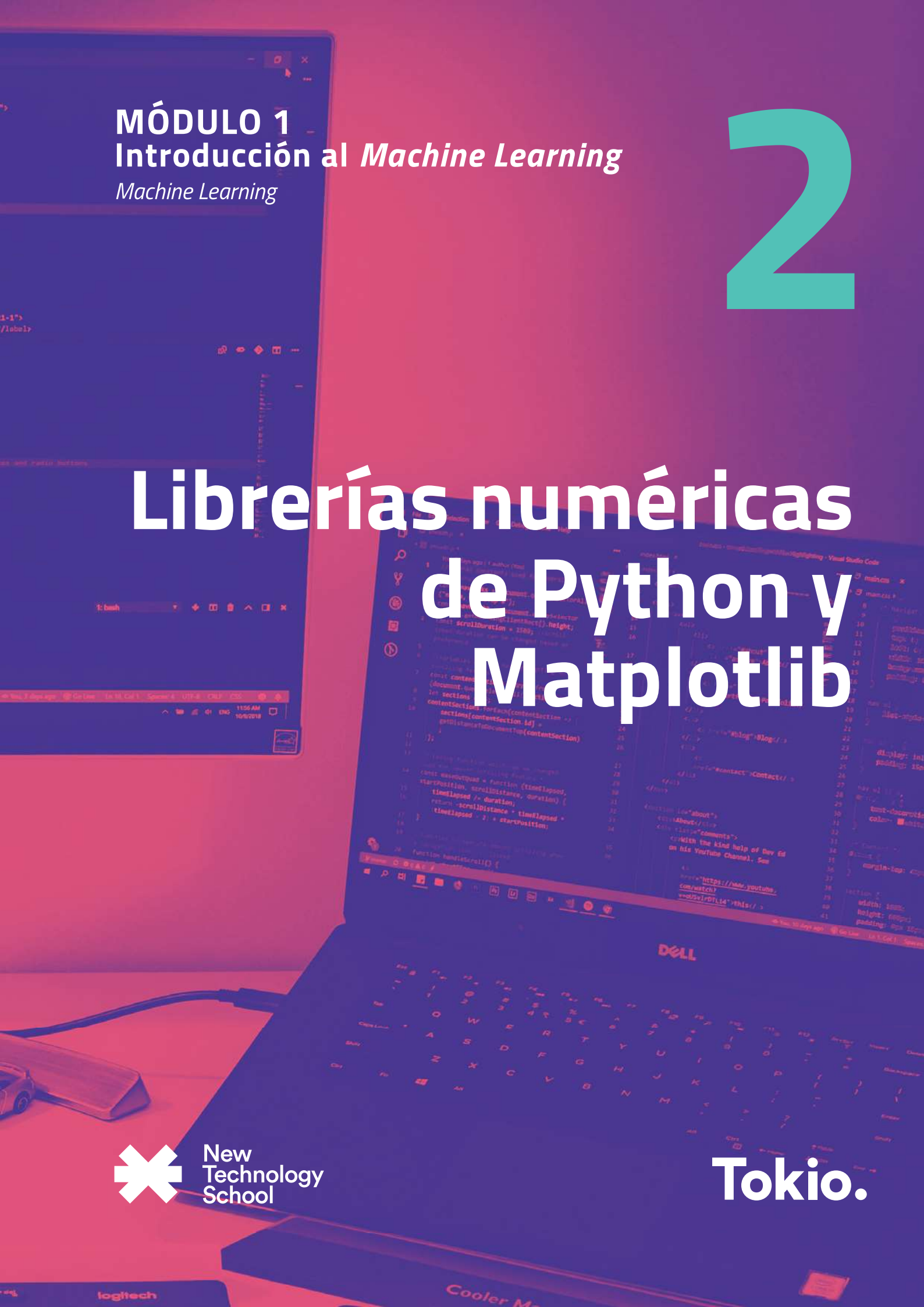
MÓDULO 1
Introducción al *Machine Learning*
Machine Learning

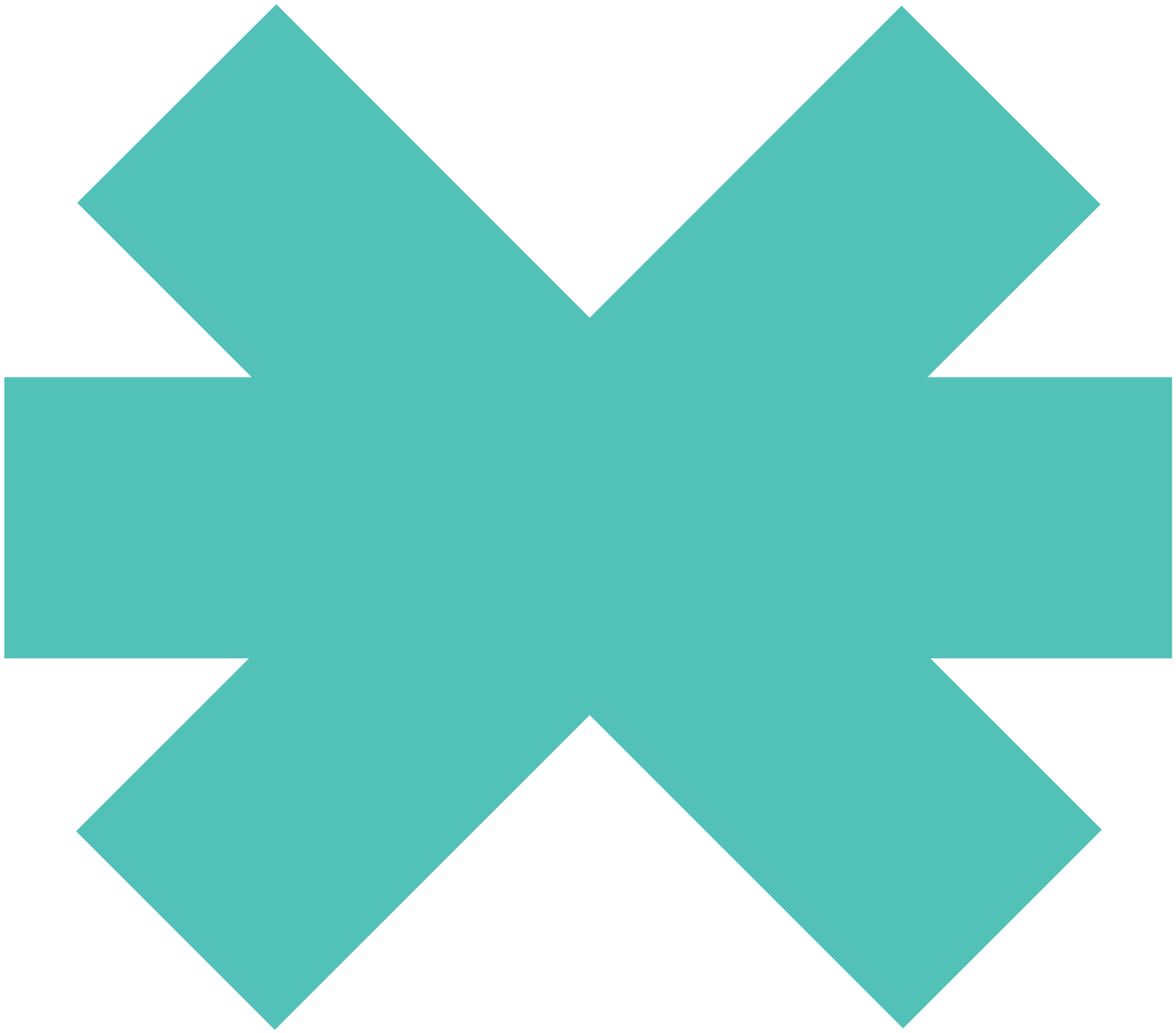
2

Librerías numéricas de Python y Matplotlib



New Technology School

Tokio.



2 Librerías numéricas de Python y Matplotlib

Sumario

En el análisis de datos y el desarrollo de modelos en Python, es recomendable el uso de **librerías especializadas** para facilitar y optimizar el trabajo de los profesionales de la ciencia de datos, ingeniería y otras disciplinas.

En primer lugar, exploraremos Pandas, una librería básica que se basa en NumPy y que permite manejar y analizar datos de manera eficiente.

- Pandas ofrece estructuras de datos flexibles y potentes, como los DataFrames y Series, que son fundamentales para organizar, manipular y realizar operaciones complejas sobre conjuntos de datos.
- A lo largo de esta sección, aprenderemos cómo utilizar Pandas para llevar a cabo tareas comunes en el análisis de datos, desde la carga y limpieza de datos hasta su transformación y análisis.

Posteriormente, nos adentraremos en Matplotlib, una librería que se ha convertido en el estándar de facto para la visualización de datos en dos dimensiones en Python.

- Matplotlib proporciona una amplia gama de funciones para crear gráficos personalizados, desde diagramas de barras hasta complejos mapas de calor.
- Veremos cómo integrar Matplotlib con Pandas para generar visualizaciones directamente a partir de nuestros datos, lo que nos permitirá comunicar hallazgos de manera clara y efectiva.

A lo largo de este tema, descubriremos cómo estas herramientas se complementan entre sí, permitiéndonos no solo analizar grandes volúmenes de datos sino también **representarlos gráficamente de manera intuitiva**, facilitando la toma de decisiones basada en datos.

Ten en cuenta que: Las explicaciones del código se incluyen a lo largo de las imágenes que se adjuntan a continuación; estas están encabezadas por una almohadilla (#) y escritas en color azul.

2.1 Introducción al análisis de datos con Pandas

Pandas es una librería de análisis de datos en Python que nos proporciona las estructuras y las herramientas necesarias para realizar análisis de manera rápida. Se articula sobre la **biblioteca NumPy** y nos permite afrontar situaciones donde debemos manejar datos reales que requieren seguir un proceso de carga, limpieza, filtrado, reducción y análisis.

```
# Importamos pandas
import pandas as pd
```

```
# Otros imports
# La siguiente instruccion permite incorporar las graficas
# en este documento (sin que se abra una nueva ventana para cada grafica)
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np
```

Figura 1.

Estructura de Pandas

Pandas presenta dos estructuras de datos principales:

- *Dataframe.*
- *Series.*

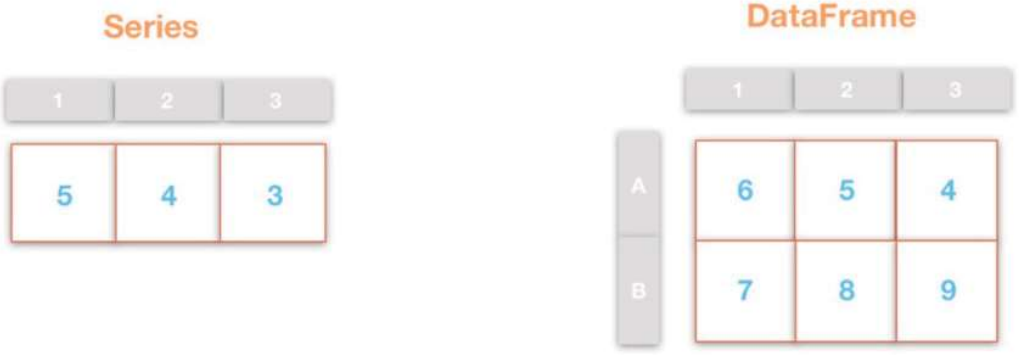


Figura 2.

DATAFRAME

El *dataframe* es una **estructura de datos tabular** formada por columnas y filas ordenadas. Para facilitar su comprensión, vamos a recurrir a un ejemplo sobre la **creación de un *dataframe*** (tabla) de un circuito de listas, donde se muestra un diccionario que consta de dos *keys*, nombre y edad, así como de su correspondiente lista de valores.

```
alumnos = {
    "nombre": ["Cristian", "Benito", "Laura"],
    "edad": [35,29,19]
}
print(alumnos)
print(type(alumnos))
```

```
{'nombre': ['Cristian', 'Benito', 'Laura'], 'edad': [35, 29, 19]}
<class 'dict'>
```

```
df = pd.DataFrame(alumnos)
print(type(df))
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
# Visualización del df sin formato.
# Lo imprime el print de Python
print(df)
```

```
     nombre  edad
0  Cristian    35
1    Benito    29
2     Laura    19
```

```
# Visualización del df con un formato
# especial que nos proporciona Jupyter
df
```

	nombre	edad
0	Cristian	35
1	Benito	29
2	Laura	19

Figura 3.

¿La estructura del *dataframe* es fija? ¿Podemos modificar algún aspecto?

```
# Cambiar el orden de las columnas
df2 = pd.DataFrame(alumnos, columns = ["edad", "nombre"])
df2
```

	edad	nombre
0	35	Cristian
1	29	Benito
2	19	Laura

```
# Renombrar columnas
df2 = df2.rename(columns = {"edad": "Edad", "nombre": "Nombre"})
df2
```

	Edad	Nombre
0	35	Cristian
1	29	Benito
2	19	Laura

```
# Cambiar los indices de numeros a letras
df2 = pd.DataFrame(alumnos, columns = ["nombre", "edad"],
                   index = ["a", "b", "c"])
df2
```

	nombre	edad
a	Cristian	35
b	Benito	29
c	Laura	19

```
# Capitalizar o convertir a mayusculas o minusculas
# title: Capitaliza un texto
# upper: Mayusculas
# lower: Minusculas
df2 = df2.rename(columns=str.title, index=str.upper)
df2
```

	Nombre	Edad
A	Cristian	35
B	Benito	29
C	Laura	19

```
# Añadir un prefijo
df2 = df2.add_prefix("X_")
df2
```

	X_Nombre	X_Edad
A	Cristian	35
B	Benito	29
C	Laura	19

```
# Añadir un sufijo
df2 = df2.add_suffix("_Y")
df2
```

	X_Nombre_Y	X_Edad_Y
A	Cristian	35
B	Benito	29
C	Laura	19

Figura 4.

SERIES

Series es un **objeto unidimensional** (1D) similar a la columna de una tabla. Si queremos crear *series* para un **listado de nombres**, procederemos del siguiente modo:

```
nombres = ["Cristian", "David", "Eva"]
print(type(nombres))
serie_nombres = pd.Series(nombres, index=[1,2,3])
print(serie_nombres)
print(type(serie_nombres))
```

```
<class 'list'>
1    Cristian
2       David
3         Eva
dtype: object
<class 'pandas.core.series.Series'>
```

Figura 5.

Crear un *dataframe* con un objeto del tipo *series*

```
# Crear un dataframe con un objeto tipo Series
nombres = ["Cristian", "David", "Eva"]
serie_nombres = pd.Series(nombres, name="Nombre")
df = serie_nombres.to_frame()
print(type(df))
df
```

```
<class 'pandas.core.frame.DataFrame'>
```

	Nombre
0	Cristian
1	David
2	Eva

```
# Crear un dataframe con varios objetos del tipo Series
nombres = ["Cristian", "David", "Eva"]
edades = [30,40,50]

serie_nombres = pd.Series(nombres)
serie_edades = pd.Series(edades)

# Creamos un diccionario que integra dos series en su interior
frame = {
    "Nombre": serie_nombres,
    "Edad": serie_edades
}

df = pd.DataFrame(frame)
df
```

	Nombre	Edad
0	Cristian	30
1	David	40
2	Eva	50

Figura 6

```
# También tenemos un atajo, podemos crear un dataframe
# con varias listas directamente

nombres = ["Cristian", "David", "Eva"]
edades = [30,40,50]

# Creamos un diccionario que integra las dos listas
frame = {
    "Nombre": nombres,
    "Edad": edades
}

df = pd.DataFrame(frame)
df
```

	Nombre	Edad
0	Cristian	30
1	David	40
2	Eva	50

Figura 7.

Ampliar nuestro *dataframe*: añadir columnas y filas

```
# Añadir una Serie o lista nueva a nuestro dataframe ya creado
ciudades = ["Madrid", "BCN", "Londres"]
serie_ciudades = pd.Series(ciudades)

df["Ciudad"] = serie_ciudades # Añadir la serie
# df["Ciudad"] = ciudades     # Añadir la lista
df
```

	Nombre	Edad	Ciudad
0	Cristian	30	Madrid
1	David	40	BCN
2	Eva	50	Londres

```
# Reordenar nuestras columnas del DF
listaOrdenColumnas = ["Nombre", "Ciudad", "Edad"]
df = df[listaOrdenColumnas]
df
```

	Nombre	Ciudad	Edad
0	Cristian	Madrid	30
1	David	BCN	40
2	Eva	Londres	50

```
# Forma de añadir una nueva fila al DF
nombre_input = input("Introduzca un nombre:")
ciudad_input = input("Introduzca una ciudad:")
edad_input = int(input("Introduzca una edad:"))
nueva_fila = {
    "Nombre": nombre_input,
    "Ciudad": ciudad_input,
    "Edad": edad_input
}
print(nueva_fila)

df = df.append(nueva_fila, ignore_index=True)
df
```

```
Introduzca un nombre:Carlos
Introduzca una ciudad:Tokio
Introduzca una edad:20
{'Nombre': 'Carlos', 'Ciudad': 'Tokio', 'Edad': 20}
```

	Nombre	Ciudad	Edad
0	Cristian	Madrid	30
1	David	BCN	40
2	Eva	Londres	50
3	Carlos	Tokio	20

```
# Vamos a simular un error para ver como corregirlo
nombre_input = "Sara"
ciudades_input = "Madrid"
edades_input =  25
nueva_fila = {
    # ERROR INTENCIONADO. Estamos poniendo Nombres
    # en vez de Nombre (veremos que nos crea esta nueva
    # columna y nos descuadra el DF)
    "Nombres" : nombre_input,
    "Ciudad" : ciudades_input,
    "Edad" : edades_input
}
print (nueva_fila)

df = df.append(nueva_fila, ignore_index = True)
df
```

```
{'Nombres': 'Sara', 'Ciudad': 'Madrid', 'Edad': 25}
```

	Nombre	Ciudad	Edad	Nombres
0	Cristian	Madrid	30	NaN
1	David	BCN	40	NaN
2	Eva	Londres	50	NaN
3	Carlos	Tokio	20	NaN
4	NaN	Madrid	25	Sara

```
# Eliminamos la columna erronea Nombres
# axis=1 significa borrar columna
df = df.drop(["Nombres"], axis=1)
df
```

	Nombre	Ciudad	Edad
0	Cristian	Madrid	30
1	David	BCN	40
2	Eva	Londres	50
3	Carlos	Tokio	20
4	NaN	Madrid	25

```
# Eliminamos filas con errores (con valores NaN)
# La siguiente linea borraria la fila 4
df = df.drop([4], axis=0) # axis=0 significa borrar fila

# La siguiente linea borraria la fila 4 y 5
#df = df.drop([4,5], axis=0)

# La siguiente linea borra el rango especificado
# (de 4 a 6 en saltos de 1)
#df = df.drop(range(4,6,1), axis=0)

df
```

	Nombre	Ciudad	Edad
0	Cristian	Madrid	30
1	David	BCN	40
2	Eva	Londres	50
3	Carlos	Tokio	20

Figura 8.

Fusionar *dataframes*

```
alumnos1 = {
    "nombre": ["Cristian", "Benito"],
    "edad": [35,29]
}
df1 = pd.DataFrame(alumnos1)

alumnos2 = {
    "nombre": ["Carlos", "Sara"],
    "edad": [31,18],
    "altura": [180,170]
}
df2 = pd.DataFrame(alumnos2)
```

```
df1
```

	nombre	edad
0	Cristian	35
1	Benito	29

```
df2
```

	nombre	edad	altura
0	Carlos	31	180
1	Sara	18	170

```
# Utilizamos append para fusionar los DF
df3 = df1.append(df2)
df3
```

	nombre	edad	altura
0	Cristian	35	NaN
1	Benito	29	NaN
0	Carlos	31	180.0
1	Sara	18	170.0

```
# Nos damos cuenta de que los identificadores de pandas
# son los originales, por lo que aparecen duplicados el 0 y el 1
# Para corregirlo le diremos que ignore los index al fusionar
# Una vez que el DF este fusionado, reasignará los ids
df3 = df1.append(df2, ignore_index = True)
df3
# NaN significa Not A Number (valor nulo (null))
```

	nombre	edad	altura
0	Cristian	35	NaN
1	Benito	29	NaN
2	Carlos	31	180.0
3	Sara	18	170.0

Figura 9.

Búsqueda y eliminación de nulos

```
df3.isnull()
```

	nombre	edad	altura
0	False	False	True
1	False	False	True
2	False	False	False
3	False	False	False

```
# Número de nulos/NaN por columna
df3.isnull().sum()
```

```
nombre    0
edad      0
altura    2
dtype: int64
```

```
# Número total de nulos
sumatorioDeNulos = df3.isnull().sum().sum()
print(sumatorioDeNulos)
```

```
2
```

```
# Mostrar las filas que tienen nulos
condicion = df3.isnull().any(axis=1)
df3[condicion]
```

	nombre	edad	altura
0	Cristian	35	NaN
1	Benito	29	NaN

```
# Recordemos como era el DF
df3
```

	nombre	edad	altura
0	Cristian	35	NaN
1	Benito	29	NaN
2	Carlos	31	180.0
3	Sara	18	170.0

```
# Eliminamos los NaN
# Creo un DF nuevo (df4) para no perder los datos
# originales en caso de error o equivocación
df4 = df3.dropna()
df4
```

	nombre	edad	altura
2	Carlos	31	180.0
3	Sara	18	170.0

```
# Reseteamos los identificadores o índices de Pandas
df4 = df4.reset_index(drop=True)
df4
```

	nombre	edad	altura
0	Carlos	31	180.0
1	Sara	18	170.0

```
# Búsqueda y sustitución de nulos / NaN
df3 = df3.fillna(0)
df3
# Si quisieramos sustituir los NaN de columnas concretas
listadoDeColumnas = ["edad", "altura"]
df3 = df3[listadoDeColumnas].fillna(0)
df3
```

	edad	altura
0	35	0.0
1	29	0.0
2	31	180.0
3	18	170.0

Figura 10.

2.2 Ejemplo práctico con Pandas

Trabajaremos con el **fichero *weather.csv*** y explicaremos cómo importarlo, explorarlo, transformarlo y exportarlo.

importación

```
ruta = "./weather.csv"
df = pd.read_csv(ruta, delimiter=",")
df
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

96453 rows × 12 columns

Figura 11.

Tamaño de nuestro DF

```
print("Número de filas:", len(df))
print("Número de columnas:", len(df.columns))
```

```
Número de filas: 96453
Número de columnas: 12
```

Figura 12.

Explorar columnas y valores

```
print(df.columns)
```

```
Index(['Formatted Date', 'Summary', 'Precip Type', 'Temperature (C)',
       'Apparent Temperature (C)', 'Humidity', 'Wind Speed (km/h)',
       'Wind Bearing (degrees)', 'Visibility (km)', 'Loud Cover',
       'Pressure (millibars)', 'Daily Summary'],
      dtype='object')
```

```
print(df.values)
```

```
[['2006-04-01 00:00:00.000 +0200' 'Partly Cloudy' 'rain' ... 0.0 1015.13
  'Partly cloudy throughout the day.']
 ['2006-04-01 01:00:00.000 +0200' 'Partly Cloudy' 'rain' ... 0.0 1015.63
  'Partly cloudy throughout the day.']
 ['2006-04-01 02:00:00.000 +0200' 'Mostly Cloudy' 'rain' ... 0.0 1015.94
  'Partly cloudy throughout the day.']
 ...
 ['2016-09-09 21:00:00.000 +0200' 'Partly Cloudy' 'rain' ... 0.0 1015.66
  'Partly cloudy starting in the morning.']
 ['2016-09-09 22:00:00.000 +0200' 'Partly Cloudy' 'rain' ... 0.0 1015.95
  'Partly cloudy starting in the morning.']
 ['2016-09-09 23:00:00.000 +0200' 'Partly Cloudy' 'rain' ... 0.0 1016.16
  'Partly cloudy starting in the morning.']]
```

Figura 13.

Información general del DF

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            96453 non-null  object
 1   Summary                   96453 non-null  object
 2   Precip Type               95936 non-null  object
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  float64
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object
dtypes: float64(8), object(4)
memory usage: 8.8+ MB
```

Figura 14.

Visualizar una muestra de datos con *head* y *tail*

```
print(df.head()) # Primeras 5 filas por defecto
```

```
                 Formatted Date        Summary Precip Type  Temperature (C)  \
0  2006-04-01 00:00:00.000 +0200  Partly Cloudy        rain         9.472222
1  2006-04-01 01:00:00.000 +0200  Partly Cloudy        rain         9.355556
2  2006-04-01 02:00:00.000 +0200  Mostly Cloudy        rain         9.377778
3  2006-04-01 03:00:00.000 +0200  Partly Cloudy        rain         8.288889
4  2006-04-01 04:00:00.000 +0200  Mostly Cloudy        rain         8.755556

   Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                  7.388889      0.89            14.1197
1                  7.227778      0.86            14.2646
2                  9.377778      0.89             3.9284
3                  5.944444      0.83            14.1036
4                  6.977778      0.83            11.0446

   Wind Bearing (degrees)  Visibility (km)  Loud Cover  Pressure (millibars)  \
0                   251.0          15.8263         0.0               1015.13
1                   259.0          15.8263         0.0               1015.63
2                   204.0          14.9569         0.0               1015.94
3                   269.0          15.8263         0.0               1016.41
4                   259.0          15.8263         0.0               1016.51

                      Daily Summary
0  Partly cloudy throughout the day.
1  Partly cloudy throughout the day.
2  Partly cloudy throughout the day.
3  Partly cloudy throughout the day.
4  Partly cloudy throughout the day.
```

```
print(df.tail()) # Últimas 5 filas por defecto
```

```
                      Formatted Date        Summary Precip Type  \
96448  2016-09-09 19:00:00.000 +0200  Partly Cloudy        rain
96449  2016-09-09 20:00:00.000 +0200  Partly Cloudy        rain
96450  2016-09-09 21:00:00.000 +0200  Partly Cloudy        rain
96451  2016-09-09 22:00:00.000 +0200  Partly Cloudy        rain
96452  2016-09-09 23:00:00.000 +0200  Partly Cloudy        rain

       Temperature (C)  Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
96448        26.016667                 26.016667      0.43            10.9963
96449        24.583333                 24.583333      0.48            10.0947
96450        22.038889                 22.038889      0.56             8.9838
96451        21.522222                 21.522222      0.60            10.5294
96452        20.438889                 20.438889      0.61             5.8765

       Wind Bearing (degrees)  Visibility (km)  Loud Cover  \
96448                    31.0          16.1000         0.0
96449                    20.0          15.5526         0.0
96450                    30.0          16.1000         0.0
96451                    20.0          16.1000         0.0
96452                    39.0          15.5204         0.0

       Pressure (millibars)                         Daily Summary
96448               1014.36  Partly cloudy starting in the morning.
96449               1015.16  Partly cloudy starting in the morning.
96450               1015.66  Partly cloudy starting in the morning.
96451               1015.95  Partly cloudy starting in the morning.
96452               1016.16  Partly cloudy starting in the morning.
```

```
print(df.head(3)) # Primeras 3 filas
```

```
                 Formatted Date        Summary Precip Type  Temperature (C)  \
0  2006-04-01 00:00:00.000 +0200  Partly Cloudy        rain         9.472222
1  2006-04-01 01:00:00.000 +0200  Partly Cloudy        rain         9.355556
2  2006-04-01 02:00:00.000 +0200  Mostly Cloudy        rain         9.377778

   Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                  7.388889      0.89            14.1197
1                  7.227778      0.86            14.2646
2                  9.377778      0.89             3.9284

   Wind Bearing (degrees)  Visibility (km)  Loud Cover  Pressure (millibars)  \
0                   251.0          15.8263         0.0               1015.13
1                   259.0          15.8263         0.0               1015.63
2                   204.0          14.9569         0.0               1015.94

                      Daily Summary
0  Partly cloudy throughout the day.
1  Partly cloudy throughout the day.
2  Partly cloudy throughout the day.
```

```
print(df.tail(3)) # Últimas 3 filas
```

```
                  Formatted Date        Summary Precip Type  \
96450  2016-09-09 21:00:00.000 +0200  Partly Cloudy        rain
96451  2016-09-09 22:00:00.000 +0200  Partly Cloudy        rain
96452  2016-09-09 23:00:00.000 +0200  Partly Cloudy        rain

       Temperature (C)  Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
96450        22.038889                 22.038889      0.56             8.9838
96451        21.522222                 21.522222      0.60            10.5294
96452        20.438889                 20.438889      0.61             5.8765

       Wind Bearing (degrees)  Visibility (km)  Loud Cover  \
96450                    30.0          16.1000         0.0
96451                    20.0          16.1000         0.0
96452                    39.0          15.5204         0.0

       Pressure (millibars)                        Daily Summary
96450               1015.66  Partly cloudy starting in the morning.
96451               1015.95  Partly cloudy starting in the morning.
96452               1016.16  Partly cloudy starting in the morning.
```

Figura 15.

Visualizar un fragmento de datos usando *slicing*

```
# Hacer slicing al dataframe, obtener 5 filas
# ubicadas entre las filas 20 y 25
df[20:26]
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
20	2006-04-01 20:00:00.000 +0200	Mostly Cloudy	rain	11.550000	11.550000	0.77	7.3899	147.0	11.0285	0.0	1015.85	Partly cloudy throughout the day.
21	2006-04-01 21:00:00.000 +0200	Mostly Cloudy	rain	11.183333	11.183333	0.76	4.9266	160.0	9.9820	0.0	1015.77	Partly cloudy throughout the day.
22	2006-04-01 22:00:00.000 +0200	Partly Cloudy	rain	10.116667	10.116667	0.79	6.6493	163.0	15.8263	0.0	1015.40	Partly cloudy throughout the day.
23	2006-04-01 23:00:00.000 +0200	Mostly Cloudy	rain	10.200000	10.200000	0.77	3.9284	152.0	14.9569	0.0	1015.51	Partly cloudy throughout the day.
24	2006-04-10 00:00:00.000 +0200	Partly Cloudy	rain	10.422222	10.422222	0.62	16.9855	150.0	15.8263	0.0	1014.40	Mostly cloudy throughout the day.
25	2006-04-10 01:00:00.000 +0200	Partly Cloudy	rain	9.911111	7.566667	0.66	17.2109	149.0	15.8263	0.0	1014.20	Mostly cloudy throughout the day.

```
# Crear un DF de un rango concreto del DF original
df_trozo = df[20:26]
print(type(df_trozo))
df_trozo
```

```
<class 'pandas.core.frame.DataFrame'>
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
20	2006-04-01 20:00:00.000 +0200	Mostly Cloudy	rain	11.550000	11.550000	0.77	7.3899	147.0	11.0285	0.0	1015.85	Partly cloudy throughout the day.
21	2006-04-01 21:00:00.000 +0200	Mostly Cloudy	rain	11.183333	11.183333	0.76	4.9266	160.0	9.9820	0.0	1015.77	Partly cloudy throughout the day.
22	2006-04-01 22:00:00.000 +0200	Partly Cloudy	rain	10.116667	10.116667	0.79	6.6493	163.0	15.8263	0.0	1015.40	Partly cloudy throughout the day.
23	2006-04-01 23:00:00.000 +0200	Mostly Cloudy	rain	10.200000	10.200000	0.77	3.9284	152.0	14.9569	0.0	1015.51	Partly cloudy throughout the day.
24	2006-04-10 00:00:00.000 +0200	Partly Cloudy	rain	10.422222	10.422222	0.62	16.9855	150.0	15.8263	0.0	1014.40	Mostly cloudy throughout the day.
25	2006-04-10 01:00:00.000 +0200	Partly Cloudy	rain	9.911111	7.566667	0.66	17.2109	149.0	15.8263	0.0	1014.20	Mostly cloudy throughout the day.

```
# Observamos que este DF es más pequeño y ocupa menos
df_trozo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 20 to 25
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            6 non-null      object
 1   Summary                   6 non-null      object
 2   Precip Type               6 non-null      object
 3   Temperature (C)           6 non-null      float64
 4   Apparent Temperature (C)  6 non-null      float64
 5   Humidity                  6 non-null      float64
 6   Wind Speed (km/h)         6 non-null      float64
 7   Wind Bearing (degrees)    6 non-null      float64
 8   Visibility (km)           6 non-null      float64
 9   Loud Cover                6 non-null      float64
 10  Pressure (millibars)      6 non-null      float64
 11  Daily Summary             6 non-null      object
dtypes: float64(8), object(4)
memory usage: 708.0+ bytes
```

Figura 16.

Acceder a una columna en concreto

```
# Accedemos a la columna Humidity, esto devuelve una Serie
df["Humidity"]
```

```
0        0.89
1        0.86
2        0.89
3        0.83
4        0.83
         ...
96448    0.43
96449    0.48
96450    0.56
96451    0.60
96452    0.61
Name: Humidity, Length: 96453, dtype: float64
```

```
df["Humidity"].head(5) # Mostramos las primeras 5 filas
```

```
0    0.89
1    0.86
2    0.89
3    0.83
4    0.83
Name: Humidity, dtype: float64
```

```
df["Humidity"][20:25] # Mostrar las filas entre la 20 y la 25
```

```
20    0.77
21    0.76
22    0.79
23    0.77
24    0.62
Name: Humidity, dtype: float64
```

Figura 17.

Realizar cálculos sobre una columna en concreto

```
min_h = df["Humidity"].min()
max_h = df["Humidity"].max()

id_min_h = df["Humidity"].idxmin()
id_max_h = df["Humidity"].idxmax()

print("Humedad minima: {} -> Posicion: {}".format(min_h, id_min_h))
print("Humedad maxima: {} -> Posicion: {}".format(max_h, id_max_h))

# Promedio entre el total de registros de la columna
mean_h = df["Humidity"].mean()
# Omitimos los nulos para hacer el promedio
mean_h = df["Humidity"].mean(skipna = True)
sum_h = df["Humidity"].sum()

print("Humedad promedio:", round(mean_h,4)) # Redondeado a 4 decimales
print("Humedad sumatorio:", sum_h)
```

```
Humedad minima: 0.0 -> Posicion: 19958
Humedad maxima: 1.0 -> Posicion: 319
Humedad promedio: 0.7349
Humedad sumatorio: 70883.20999999999
```

```
# Accedemos a la fila de la humedad minima
df[19958:19959]
# O lo que es lo mismo:
# df[id_min_h:id_min_h+1]
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
19958	2008-02-17 14:00:00.000 +0100	Partly Cloudy	snow	-1.111111	-1.111111	0.0	4.4275	12.0	9.982	0.0	1043.04	Partly cloudy starting in the morning continui...

```
# Podemos hacer el programa más interactivo y hacer que sea el
# usuario quien solicite el rango para ver las filas que quiera
minimo = int(input("Introduzca minimo: "))
maximo = int(input("Introduzca maximo: "))
df[minimo:maximo+1]
```

```
Introduzca minimo: 20
Introduzca maximo: 22
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
20	2006-04-01 20:00:00.000 +0200	Mostly Cloudy	rain	11.550000	11.550000	0.77	7.3899	147.0	11.0285	0.0	1015.85	Partly cloudy throughout the day.
21	2006-04-01 21:00:00.000 +0200	Mostly Cloudy	rain	11.183333	11.183333	0.76	4.9266	160.0	9.9820	0.0	1015.77	Partly cloudy throughout the day.
22	2006-04-01 22:00:00.000 +0200	Partly Cloudy	rain	10.116667	10.116667	0.79	6.6493	163.0	15.8263	0.0	1015.40	Partly cloudy throughout the day.

Figura 18.

Ordenación

```
# Ordenamos una columna
temps = df["Temperature (C)"]
print(type(temps)) # Esto es una serie
print(temps)
```

```
<class 'pandas.core.series.Series'>
0         9.472222
1         9.355556
2         9.377778
3         8.288889
4         8.755556
           ...
96448    26.016667
96449    24.583333
96450    22.038889
96451    21.522222
96452    20.438889
Name: Temperature (C), Length: 96453, dtype: float64
```

```
# De mayor a menor
temps_ordenadas = temps.sort_values(ascending=False)
print(temps_ordenadas)
```

```
12759    39.905556
12737    39.588889
12711    38.983333
12664    38.983333
12712    38.983333
           ...
55495   -20.783333
55494   -21.111111
55490   -21.111111
55493   -21.111111
54847   -21.822222
Name: Temperature (C), Length: 96453, dtype: float64
```

```
# Ordenamos todo el dataframe en funcion a una columna
df_ordenado = df.sort_values(by=["Temperature (C)"],
                             ascending=False)
df_ordenado
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
12759	2007-07-22 15:00:00.000 +0200	Clear	rain	39.905556	37.538889	0.13	23.5865	250.0	9.9820	0.0	1007.55	Partly cloudy starting in the afternoon.
12737	2007-07-21 17:00:00.000 +0200	Partly Cloudy	rain	39.588889	37.600000	0.15	12.2038	283.0	10.3523	0.0	1010.24	Partly cloudy starting in the morning continui...
12664	2007-07-19 16:00:00.000 +0200	Partly Cloudy	rain	38.983333	37.461111	0.18	8.9355	175.0	9.9820	0.0	1013.34	Partly cloudy starting in the afternoon.
12712	2007-07-20 16:00:00.000 +0200	Partly Cloudy	rain	38.983333	36.800000	0.15	15.4077	133.0	9.9820	0.0	1011.93	Partly cloudy starting in the afternoon contin...
12711	2007-07-20 15:00:00.000 +0200	Partly Cloudy	rain	38.983333	36.627778	0.14	12.7190	167.0	9.9820	0.0	1012.34	Partly cloudy starting in the afternoon contin...
...	...	...	...	...	...	...	...	...	...	...	...	...
55495	2012-02-09 07:00:00.000 +0100	Foggy	snow	-20.783333	-20.783333	0.80	4.4275	181.0	1.7871	0.0	1032.33	Foggy until morning.
55494	2012-02-09 06:00:00.000 +0100	Foggy	snow	-21.111111	-21.111111	0.71	3.2200	190.0	1.9320	0.0	1032.60	Foggy until morning.
55493	2012-02-09 05:00:00.000 +0100	Foggy	snow	-21.111111	-21.111111	0.78	4.8300	180.0	2.5760	0.0	1033.30	Foggy until morning.
55490	2012-02-09 02:00:00.000 +0100	Foggy	snow	-21.111111	-21.111111	0.74	3.2200	200.0	1.9320	0.0	1034.30	Foggy until morning.
54847	2012-02-10 07:00:00.000 +0100	Foggy	snow	-21.822222	-21.822222	0.80	3.0751	323.0	1.3685	0.0	1033.66	Foggy starting in the morning continuing until...

96453 rows × 12 columns

```
# Ordenamos todo el dataframe en funcion a una columna principal
# (temperatura) y a una columna secundaria (temperatura aparente)
df_ordenado = df.sort_values(by=["Temperature (C)",
                                 "Apparent Temperature (C)"],
                             ascending=False)
df_ordenado
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
12759	2007-07-22 15:00:00.000 +0200	Clear	rain	39.905556	37.538889	0.13	23.5865	250.0	9.9820	0.0	1007.55	Partly cloudy starting in the afternoon.
12737	2007-07-21 17:00:00.000 +0200	Partly Cloudy	rain	39.588889	37.600000	0.15	12.2038	283.0	10.3523	0.0	1010.24	Partly cloudy starting in the morning continui...
12664	2007-07-19 16:00:00.000 +0200	Partly Cloudy	rain	38.983333	37.461111	0.18	8.9355	175.0	9.9820	0.0	1013.34	Partly cloudy starting in the afternoon.
12712	2007-07-20 16:00:00.000 +0200	Partly Cloudy	rain	38.983333	36.800000	0.15	15.4077	133.0	9.9820	0.0	1011.93	Partly cloudy starting in the afternoon contin...
12711	2007-07-20 15:00:00.000 +0200	Partly Cloudy	rain	38.983333	36.627778	0.14	12.7190	167.0	9.9820	0.0	1012.34	Partly cloudy starting in the afternoon contin...
...	...	...	...	...	...	...	...	...	...	...	...	...
55495	2012-02-09 07:00:00.000 +0100	Foggy	snow	-20.783333	-20.783333	0.80	4.4275	181.0	1.7871	0.0	1032.33	Foggy until morning.
55490	2012-02-09 02:00:00.000 +0100	Foggy	snow	-21.111111	-21.111111	0.74	3.2200	200.0	1.9320	0.0	1034.30	Foggy until morning.
55493	2012-02-09 05:00:00.000 +0100	Foggy	snow	-21.111111	-21.111111	0.78	4.8300	180.0	2.5760	0.0	1033.30	Foggy until morning.
55494	2012-02-09 06:00:00.000 +0100	Foggy	snow	-21.111111	-21.111111	0.71	3.2200	190.0	1.9320	0.0	1032.60	Foggy until morning.
54847	2012-02-10 07:00:00.000 +0100	Foggy	snow	-21.822222	-21.822222	0.80	3.0751	323.0	1.3685	0.0	1033.66	Foggy starting in the morning continuing until...

96453 rows × 12 columns

```
# Ordenamos todo el dataframe en funcion a una columna principal
# (temperatura) y a una columna secundaria (temperatura aparente)
# Y ponemos los nulos o NaN arriba del todo (first)
# o abajo del todo (last)
df_ordenado = df.sort_values(by=["Temperature (C)",
                                 "Apparent Temperature (C)"],
                             ascending=False,
                             na_position="first")
df_ordenado
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
12759	2007-07-22 15:00:00.000 +0200	Clear	rain	39.905556	37.538889	0.13	23.5865	250.0	9.9820	0.0	1007.55	Partly cloudy starting in the afternoon.
12737	2007-07-21 17:00:00.000 +0200	Partly Cloudy	rain	39.588889	37.600000	0.15	12.2038	283.0	10.3523	0.0	1010.24	Partly cloudy starting in the morning continui...
12664	2007-07-19 16:00:00.000 +0200	Partly Cloudy	rain	38.983333	37.461111	0.18	8.9355	175.0	9.9820	0.0	1013.34	Partly cloudy starting in the afternoon.
12712	2007-07-20 16:00:00.000 +0200	Partly Cloudy	rain	38.983333	36.800000	0.15	15.4077	133.0	9.9820	0.0	1011.93	Partly cloudy starting in the afternoon contin...
12711	2007-07-20 15:00:00.000 +0200	Partly Cloudy	rain	38.983333	36.627778	0.14	12.7190	167.0	9.9820	0.0	1012.34	Partly cloudy starting in the afternoon contin...
...	...	...	...	...	...	...	...	...	...	...	...	...
55495	2012-02-09 07:00:00.000 +0100	Foggy	snow	-20.783333	-20.783333	0.80	4.4275	181.0	1.7871	0.0	1032.33	Foggy until morning.
55490	2012-02-09 02:00:00.000 +0100	Foggy	snow	-21.111111	-21.111111	0.74	3.2200	200.0	1.9320	0.0	1034.30	Foggy until morning.
55493	2012-02-09 05:00:00.000 +0100	Foggy	snow	-21.111111	-21.111111	0.78	4.8300	180.0	2.5760	0.0	1033.30	Foggy until morning.
55494	2012-02-09 06:00:00.000 +0100	Foggy	snow	-21.111111	-21.111111	0.71	3.2200	190.0	1.9320	0.0	1032.60	Foggy until morning.
54847	2012-02-10 07:00:00.000 +0100	Foggy	snow	-21.822222	-21.822222	0.80	3.0751	323.0	1.3685	0.0	1033.66	Foggy starting in the morning continuing until...

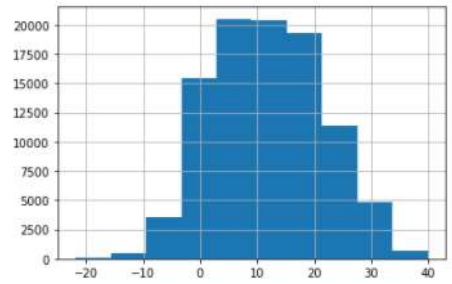
96453 rows × 12 columns

Figura 19.

Gráficos de Pandas. Histograma

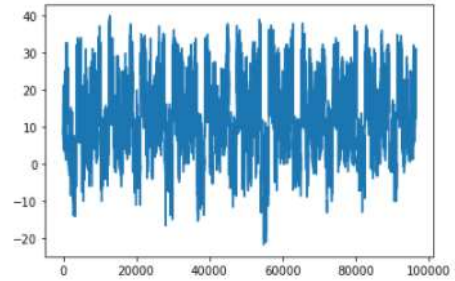
```
df['Temperature (C)'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x198e1eb2250>
```




```
df['Temperature (C)'].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x198e1a9d0d0>
```




```
grafico = df['Temperature (C)'].plot(kind='kde')
```

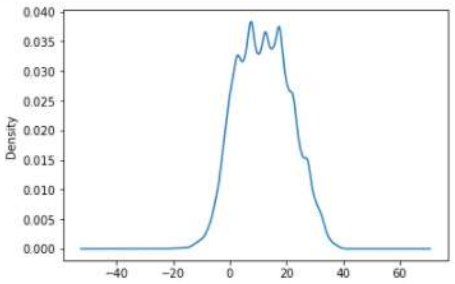


Figura 20.

Otras funciones de formato

```
# Convertir a mayusculas la columna Summary
df["Summary"].str.upper()
# Si el cambio anterior si quiere hacer permanente:
# df["Summary"] = df["Summary"].str.upper()
```

```
0        PARTLY CLOUDY
1        PARTLY CLOUDY
2        MOSTLY CLOUDY
3        PARTLY CLOUDY
4        MOSTLY CLOUDY
            ...
96448    PARTLY CLOUDY
96449    PARTLY CLOUDY
96450    PARTLY CLOUDY
96451    PARTLY CLOUDY
96452    PARTLY CLOUDY
Name: Summary, Length: 96453, dtype: object
```

```
# Convertir a minusculas la columna Summary
df["Summary"].str.lower()
```

```
0        partly cloudy
1        partly cloudy
2        mostly cloudy
3        partly cloudy
4        mostly cloudy
            ...
96448    partly cloudy
96449    partly cloudy
96450    partly cloudy
96451    partly cloudy
96452    partly cloudy
Name: Summary, Length: 96453, dtype: object
```

```
# Capitalizar la columna Summary
df["Summary"].str.capitalize()
```

```
0        Partly cloudy
1        Partly cloudy
2        Mostly cloudy
3        Partly cloudy
4        Mostly cloudy
            ...
96448    Partly cloudy
96449    Partly cloudy
96450    Partly cloudy
96451    Partly cloudy
96452    Partly cloudy
Name: Summary, Length: 96453, dtype: object
```

```
# Capitalizar todas las palabras de la columna Summary
df["Summary"].str.title()
```

```
0        Partly Cloudy
1        Partly Cloudy
2        Mostly Cloudy
3        Partly Cloudy
4        Mostly Cloudy
            ...
96448    Partly Cloudy
96449    Partly Cloudy
96450    Partly Cloudy
96451    Partly Cloudy
96452    Partly Cloudy
Name: Summary, Length: 96453, dtype: object
```

```
# Intercambiar mayusculas por minusculas
df["Summary"].str.swapcase()
```

```
0        pARTLY cLOUDY
1        pARTLY cLOUDY
2        mOSTLY cLOUDY
3        pARTLY cLOUDY
4        mOSTLY cLOUDY
            ...
96448    pARTLY cLOUDY
96449    pARTLY cLOUDY
96450    pARTLY cLOUDY
96451    pARTLY cLOUDY
96452    pARTLY cLOUDY
Name: Summary, Length: 96453, dtype: object
```

```
# Mostraremos todo el dataframe (todas las columnas)
# que cumplan una condición
# df[condicion]
df[df["Summary"] == "Foggy"] # Si queremos evaluar si es dintinto: !=
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
66	2006-04-11 18:00:00.000 +0200	Foggy	rain	10.911111	10.911111	0.86	22.3951	311.0	2.6565	0.0	1004.61	Foggy in the evening.
67	2006-04-11 19:00:00.000 +0200	Foggy	rain	8.800000	5.294444	0.99	26.5006	339.0	2.6565	0.0	1004.99	Foggy in the evening.
72	2006-04-12 00:00:00.000 +0200	Foggy	rain	8.200000	5.072222	0.96	20.4470	341.0	3.1073	0.0	1004.80	Foggy overnight and breezy in the morning.
73	2006-04-12 01:00:00.000 +0200	Foggy	rain	8.177778	4.372222	0.93	27.8691	19.0	3.2039	0.0	1004.89	Foggy overnight and breezy in the morning.
126	2006-04-14 06:00:00.000 +0200	Foggy	rain	5.211111	5.211111	0.92	4.7656	178.0	1.2236	0.0	1013.40	Mostly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
95858	2016-09-14 05:00:00.000 +0200	Foggy	rain	13.144444	13.144444	0.99	3.2039	2.0	2.2218	0.0	1015.33	Foggy in the morning.
95859	2016-09-14 06:00:00.000 +0200	Foggy	rain	13.822222	13.822222	0.94	4.8300	10.0	1.9642	0.0	1015.44	Foggy in the morning.
95860	2016-09-14 07:00:00.000 +0200	Foggy	rain	13.872222	13.872222	0.94	3.2039	321.0	1.9803	0.0	1015.64	Foggy in the morning.
95907	2016-09-16 06:00:00.000 +0200	Foggy	rain	14.905556	14.905556	1.00	7.5992	169.0	1.6100	0.0	1016.51	Partly cloudy starting in the afternoon.
96228	2016-09-28 15:00:00.000 +0200	Foggy	rain	22.800000	22.800000	0.32	10.9158	231.0	0.7245	0.0	1027.69	Partly cloudy starting in the afternoon.

7148 rows × 12 columns

Figura 21.

Condiciones simples

df[“columna”][“condicion”]

Se **devuelven todos los valores** de esa columna que cumplan la condición.

Con este método, **podemos mostrar**:

Una sola columna.	Todo el *dataframe* (no podemos mostrar un número determinado de columnas; para ello, usaremos *loc*).

```
media_humedad = df["Humidity"].mean()
print("Media de la humedad:", media_humedad)
df["Humidity"][df["Humidity"] <= media_humedad]
```

```
Media de la humedad: 0.7348989663358906

9        0.72
10       0.67
11       0.54
12       0.55
13       0.51
         ...
96448    0.43
96449    0.48
96450    0.56
96451    0.60
96452    0.61
Name: Humidity, Length: 40336, dtype: float64
```

```
# Mostrar una columna diferente, de la que se
# usa para evaluar la condicion
df["Temperature (C)"][df["Humidity"] <= media_humedad]
```

```
9        13.772222
10       16.016667
11       17.144444
12       17.800000
13       17.333333
            ...
96448    26.016667
96449    24.583333
96450    22.038889
96451    21.522222
96452    20.438889
Name: Temperature (C), Length: 40336, dtype: float64
```

```
# Mostraremos todo el dataframe (todas las columnas)
# que cumplan una condición
# df[condicion]
df[df["Humidity"] <= media_humedad]
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
9	2006-04-01 09:00:00.000 +0200	Partly Cloudy	rain	13.772222	13.772222	0.72	12.5258	279.0	9.9820	0.0	1017.22	Partly cloudy throughout the day.
10	2006-04-01 10:00:00.000 +0200	Partly Cloudy	rain	16.016667	16.016667	0.67	17.5651	290.0	11.2056	0.0	1017.42	Partly cloudy throughout the day.
11	2006-04-01 11:00:00.000 +0200	Partly Cloudy	rain	17.144444	17.144444	0.54	19.7869	316.0	11.4471	0.0	1017.74	Partly cloudy throughout the day.
12	2006-04-01 12:00:00.000 +0200	Partly Cloudy	rain	17.800000	17.800000	0.55	21.9443	281.0	11.2700	0.0	1017.59	Partly cloudy throughout the day.
13	2006-04-01 13:00:00.000 +0200	Partly Cloudy	rain	17.333333	17.333333	0.51	20.6885	289.0	11.2700	0.0	1017.48	Partly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

40336 rows × 12 columns

```
# Mostraremos todo el dataframe (todas las columnas)
# que cumplan una condición
# df[condicion]
df[df["Summary"] == "Foggy"] # Si queremos evaluar si es dintinto: !=
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
66	2006-04-11 18:00:00.000 +0200	Foggy	rain	10.911111	10.911111	0.86	22.3951	311.0	2.6565	0.0	1004.61	Foggy in the evening.
67	2006-04-11 19:00:00.000 +0200	Foggy	rain	8.800000	5.294444	0.99	26.5006	339.0	2.6565	0.0	1004.99	Foggy in the evening.
72	2006-04-12 00:00:00.000 +0200	Foggy	rain	8.200000	5.072222	0.96	20.4470	341.0	3.1073	0.0	1004.80	Foggy overnight and breezy in the morning.
73	2006-04-12 01:00:00.000 +0200	Foggy	rain	8.177778	4.372222	0.93	27.8691	19.0	3.2039	0.0	1004.89	Foggy overnight and breezy in the morning.
126	2006-04-14 06:00:00.000 +0200	Foggy	rain	5.211111	5.211111	0.92	4.7656	178.0	1.2236	0.0	1013.40	Mostly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
95858	2016-09-14 05:00:00.000 +0200	Foggy	rain	13.144444	13.144444	0.99	3.2039	2.0	2.2218	0.0	1015.33	Foggy in the morning.
95859	2016-09-14 06:00:00.000 +0200	Foggy	rain	13.822222	13.822222	0.94	4.8300	10.0	1.9642	0.0	1015.44	Foggy in the morning.
95860	2016-09-14 07:00:00.000 +0200	Foggy	rain	13.872222	13.872222	0.94	3.2039	321.0	1.9803	0.0	1015.64	Foggy in the morning.
95907	2016-09-16 06:00:00.000 +0200	Foggy	rain	14.905556	14.905556	1.00	7.5992	169.0	1.6100	0.0	1016.51	Partly cloudy starting in the afternoon.
96228	2016-09-28 15:00:00.000 +0200	Foggy	rain	22.800000	22.800000	0.32	10.9158	231.0	0.7245	0.0	1027.69	Partly cloudy starting in the afternoon.

7148 rows × 12 columns

```
# Condición multiple con AND (&)
# &: and (Y lógico)
# |: or (O lógico)
df[ (df["Summary"] == "Foggy") & (df["Precip Type"] == "snow")  ]
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
1562	2006-12-13 02:00:00.000 +0100	Foggy	snow	-0.483333	-4.150000	1.00	11.0929	219.0	0.4830	0.0	1031.56	Foggy throughout the day.
1563	2006-12-13 03:00:00.000 +0100	Foggy	snow	-0.483333	-4.061111	0.96	10.7387	200.0	0.3220	0.0	1031.47	Foggy throughout the day.
1564	2006-12-13 04:00:00.000 +0100	Foggy	snow	-0.922222	-3.477778	1.00	7.0679	206.0	0.1610	0.0	1031.23	Foggy throughout the day.
1565	2006-12-13 05:00:00.000 +0100	Foggy	snow	-1.038889	-4.400000	1.00	9.4990	199.0	0.1610	0.0	1031.41	Foggy throughout the day.
1566	2006-12-13 06:00:00.000 +0100	Foggy	snow	-1.088889	-4.438889	1.00	9.4346	219.0	0.3220	0.0	1031.98	Foggy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
93003	2016-03-18 06:00:00.000 +0100	Foggy	snow	-0.133333	-0.133333	0.92	3.1234	201.0	0.8855	0.0	1020.21	Foggy in the morning.
93265	2016-03-28 05:00:00.000 +0200	Foggy	snow	-0.783333	-2.950000	0.99	6.1019	91.0	3.1073	0.0	1016.30	Foggy in the morning.
93266	2016-03-28 06:00:00.000 +0200	Foggy	snow	-1.111111	-1.111111	0.93	0.0000	0.0	0.1610	0.0	1016.41	Foggy in the morning.
93267	2016-03-28 07:00:00.000 +0200	Foggy	snow	-1.044444	-1.044444	1.00	0.0000	0.0	0.7728	0.0	1016.60	Foggy in the morning.
93506	2016-03-09 06:00:00.000 +0100	Foggy	snow	-0.027778	-2.705556	0.99	7.8568	320.0	3.0107	0.0	1014.76	Foggy in the morning.

2982 rows × 12 columns

Figura 22.

Condiciones complejas: *loc*

Con el método anterior, bien mostramos el *dataframe* completo, bien mostramos tan solo una columna. Por tanto, si queremos aplicar un condicional y mostrar dos, tres o cuatro columnas, no podremos hacerlo, sino que tendremos que recurrir al **método *loc***, cuya estructura es la siguiente:

```
df.loc[condicion, [columna1, column2, columna3]]
```

Se **devuelven todos los valores de esas columnas que cumplan la condición**. Así, mediante este método, podremos mostrar todas las columnas que queramos.

```
# Estructura de loc
# df.loc[condicion, [columna1, column2, columna3]]
df.loc[ df["Humidity"] <= media_humedad, ["Humidity", "Temperature (C)"] ]
```

	Humidity	Temperature (C)
9	0.72	13.772222
10	0.67	16.016667
11	0.54	17.144444
12	0.55	17.800000
13	0.51	17.333333
...	...	...
96448	0.43	26.016667
96449	0.48	24.583333
96450	0.56	22.038889
96451	0.60	21.522222
96452	0.61	20.438889

40336 rows × 2 columns

```
# Alternativa de sintaxis al bloque anterior
condicion = df["Humidity"] <= media_humedad
lista_columnas = ["Humidity", "Temperature (C)"]
df.loc[ condicion, lista_columnas ]
```

	Humidity	Temperature (C)
9	0.72	13.772222
10	0.67	16.016667
11	0.54	17.144444
12	0.55	17.800000
13	0.51	17.333333
...	...	...
96448	0.43	26.016667
96449	0.48	24.583333
96450	0.56	22.038889
96451	0.60	21.522222
96452	0.61	20.438889

40336 rows × 2 columns

```
# Mostrar todas las columnas (:) que cumplan la condición
df.loc[ df["Humidity"] <= media_humedad, : ]
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
9	2006-04-01 09:00:00.000 +0200	Partly Cloudy	rain	13.772222	13.772222	0.72	12.5258	279.0	9.9820	0.0	1017.22	Partly cloudy throughout the day.
10	2006-04-01 10:00:00.000 +0200	Partly Cloudy	rain	16.016667	16.016667	0.67	17.5651	290.0	11.2056	0.0	1017.42	Partly cloudy throughout the day.
11	2006-04-01 11:00:00.000 +0200	Partly Cloudy	rain	17.144444	17.144444	0.54	19.7869	316.0	11.4471	0.0	1017.74	Partly cloudy throughout the day.
12	2006-04-01 12:00:00.000 +0200	Partly Cloudy	rain	17.800000	17.800000	0.55	21.9443	281.0	11.2700	0.0	1017.59	Partly cloudy throughout the day.
13	2006-04-01 13:00:00.000 +0200	Partly Cloudy	rain	17.333333	17.333333	0.51	20.6885	289.0	11.2700	0.0	1017.48	Partly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

40336 rows × 12 columns

Figura 23.

Más funcionalidades de *loc*

También podemos utilizar *loc* para filtrar por filas y por columnas sin necesidad de una condición específica.

```
# Podemos usar loc para filtrar columnas simplemente.
# En este ejemplo vemos todos los registros de humedad y temperatura (no hay condición)
df.loc[ : , ["Humidity", "Temperature (C)"] ]
```

	Humidity	Temperature (C)
0	0.89	9.472222
1	0.86	9.355556
2	0.89	9.377778
3	0.83	8.288889
4	0.83	8.755556
...	...	...
96448	0.43	26.016667
96449	0.48	24.583333
96450	0.56	22.038889
96451	0.60	21.522222
96452	0.61	20.438889

96453 rows × 2 columns

```
# Y podemos usar loc para filtrar un rango de filas y columnas
df.loc[ 10:15 , ["Humidity", "Temperature (C)"] ]
```

	Humidity	Temperature (C)
10	0.67	16.016667
11	0.54	17.144444
12	0.55	17.800000
13	0.51	17.333333
14	0.47	18.877778
15	0.46	18.911111

```
# O podemos usar loc para filtrar simplemente por filas
df.loc[5]  # Fila número 5
```

```
Formatted Date                2006-04-01 05:00:00.000 +0200
Summary                                      Partly Cloudy
Precip Type                                           rain
Temperature (C)                                    9.22222
Apparent Temperature (C)                           7.11111
Humidity                                              0.85
Wind Speed (km/h)                                  13.9587
Wind Bearing (degrees)                                 258
Visibility (km)                                    14.9569
Loud Cover                                               0
Pressure (millibars)                               1016.66
Daily Summary                Partly cloudy throughout the day.
Name: 5, dtype: object
```

```
df.loc[10:12] # Filas en el rango de filas 10:12
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
10	2006-04-01 10:00:00.000 +0200	Partly Cloudy	rain	16.016667	16.016667	0.67	17.5651	290.0	11.2056	0.0	1017.42	Partly cloudy throughout the day.
11	2006-04-01 11:00:00.000 +0200	Partly Cloudy	rain	17.144444	17.144444	0.54	19.7869	316.0	11.4471	0.0	1017.74	Partly cloudy throughout the day.
12	2006-04-01 12:00:00.000 +0200	Partly Cloudy	rain	17.800000	17.800000	0.55	21.9443	281.0	11.2700	0.0	1017.59	Partly cloudy throughout the day.

```
df[10:12]
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
10	2006-04-01 10:00:00.000 +0200	Partly Cloudy	rain	16.016667	16.016667	0.67	17.5651	290.0	11.2056	0.0	1017.42	Partly cloudy throughout the day.
11	2006-04-01 11:00:00.000 +0200	Partly Cloudy	rain	17.144444	17.144444	0.54	19.7869	316.0	11.4471	0.0	1017.74	Partly cloudy throughout the day.

Figura 24.

Otra herramienta condicional: *iloc*

El método *iloc* se utiliza en los *dataframes* para seleccionar los elementos **en base a su ubicación**.

A diferencia de *loc*, con *iloc* **podemos seleccionar filas**, PERO TAMBIÉN COLUMNAS, a través de su índice.

```
df.iloc[0] # Primera fila
```

```
Formatted Date                2006-04-01 00:00:00.000 +0200
Summary                                      Partly Cloudy
Precip Type                                           rain
Temperature (C)                                    9.47222
Apparent Temperature (C)                           7.38889
Humidity                                              0.89
Wind Speed (km/h)                                  14.1197
Wind Bearing (degrees)                                 251
Visibility (km)                                    15.8263
Loud Cover                                               0
Pressure (millibars)                               1015.13
Daily Summary                Partly cloudy throughout the day.
Name: 0, dtype: object
```

```
df.iloc[-1] # Última fila
```

```
Formatted Date                2016-09-09 23:00:00.000 +0200
Summary                                      Partly Cloudy
Precip Type                                           rain
Temperature (C)                                    20.4389
Apparent Temperature (C)                           20.4389
Humidity                                              0.61
Wind Speed (km/h)                                  5.8765
Wind Bearing (degrees)                                 39
Visibility (km)                                    15.5204
Loud Cover                                               0
Pressure (millibars)                               1016.16
Daily Summary                Partly cloudy starting in the morning.
Name: 96452, dtype: object
```

```
df.iloc[:, 0] # Primera columna
```

```
0        2006-04-01 00:00:00.000 +0200
1        2006-04-01 01:00:00.000 +0200
2        2006-04-01 02:00:00.000 +0200
3        2006-04-01 03:00:00.000 +0200
4        2006-04-01 04:00:00.000 +0200
                    ...
96448    2016-09-09 19:00:00.000 +0200
96449    2016-09-09 20:00:00.000 +0200
96450    2016-09-09 21:00:00.000 +0200
96451    2016-09-09 22:00:00.000 +0200
96452    2016-09-09 23:00:00.000 +0200
Name: Formatted Date, Length: 96453, dtype: object
```

```
df.iloc[:, -1] # Última columna
```

```
0              Partly cloudy throughout the day.
1              Partly cloudy throughout the day.
2              Partly cloudy throughout the day.
3              Partly cloudy throughout the day.
4              Partly cloudy throughout the day.
                       ...
96448    Partly cloudy starting in the morning.
96449    Partly cloudy starting in the morning.
96450    Partly cloudy starting in the morning.
96451    Partly cloudy starting in the morning.
96452    Partly cloudy starting in the morning.
Name: Daily Summary, Length: 96453, dtype: object
```

```
df.iloc[0:5] # Primeras cinco filas
# Esto sería casi igual que con el loc
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

```
df.iloc[:, 0:5] # Primeras cinco columnas
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778
...	...	...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889

96453 rows × 5 columns

```
df.iloc[10:15, 0:5]
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)
10	2006-04-01 10:00:00.000 +0200	Partly Cloudy	rain	16.016667	16.016667
11	2006-04-01 11:00:00.000 +0200	Partly Cloudy	rain	17.144444	17.144444
12	2006-04-01 12:00:00.000 +0200	Partly Cloudy	rain	17.800000	17.800000
13	2006-04-01 13:00:00.000 +0200	Partly Cloudy	rain	17.333333	17.333333
14	2006-04-01 14:00:00.000 +0200	Partly Cloudy	rain	18.877778	18.877778

```
# Primera, tercera y segunda filas. Y todas las columnas
df.iloc[[0,2,1], :]
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.

```
# Todas las filas de la primera, tercera y segunda columnas
df.iloc[:, [0,2,1]]
```

	Formatted Date	Precip Type	Summary
0	2006-04-01 00:00:00.000 +0200	rain	Partly Cloudy
1	2006-04-01 01:00:00.000 +0200	rain	Partly Cloudy
2	2006-04-01 02:00:00.000 +0200	rain	Mostly Cloudy
3	2006-04-01 03:00:00.000 +0200	rain	Partly Cloudy
4	2006-04-01 04:00:00.000 +0200	rain	Mostly Cloudy
...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	rain	Partly Cloudy
96449	2016-09-09 20:00:00.000 +0200	rain	Partly Cloudy
96450	2016-09-09 21:00:00.000 +0200	rain	Partly Cloudy
96451	2016-09-09 22:00:00.000 +0200	rain	Partly Cloudy
96452	2016-09-09 23:00:00.000 +0200	rain	Partly Cloudy

96453 rows × 3 columns

```
df.iloc[1:11, [2,5,7]]
```

	Precip Type	Humidity	Wind Bearing (degrees)
1	rain	0.86	259.0
2	rain	0.89	204.0
3	rain	0.83	269.0
4	rain	0.83	259.0
5	rain	0.85	258.0
6	rain	0.95	259.0
7	rain	0.89	260.0
8	rain	0.82	259.0
9	rain	0.72	279.0
10	rain	0.67	290.0

Figura 25.

Consideraciones finales acerca de *iloc*

Es importante tener en cuenta que *iloc* devuelve una *serie* Pandas cuando se selecciona una fila, y un *dataframe* cuando se seleccionan varias. Por consiguiente, en caso de que resulte necesario seleccionar un *dataframe* con una única columna, debemos pasar una lista con la columna, no un escalar.

Por otro lado, al seleccionar varias filas o columnas con *inicio:fin*, debemos recordar que los valores irán desde el principio hasta el final menos uno; por ejemplo, iloc[1:3] solamente seleccionará la segunda y tercera fila, 1 y 3. Además, tendremos en cuenta que los índices en Python comienzan en 0.

Figura 14. Robot de reparto Scout de Amazon

Agrupaciones (*groupby*)

En numerosas ocasiones, resulta muy interesante realizar agrupaciones en base a un campo (sexo, país...).

```
# Vamos a agrupar por el Summary (por el resumen del tiempo)
grouped_df = df.groupby("Summary")
#print(type(grouped_df))

# Mostramos el primer registro de cada grupo
grouped_df.first()
```

Summary	Formatted Date	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
Breezy	2006-11-05 05:00:00.000 +0100	rain	6.088889	1.572222	0.76	29.0444	289.0	6.3434	0.0	1021.45	Overcast until night and breezy overnight.
Breezy and Dry	2012-03-18 11:00:00.000 +0100	rain	21.111111	21.111111	0.26	33.8100	240.0	9.9820	0.0	1021.60	Partly cloudy starting in the afternoon.
Breezy and Foggy	2006-03-13 07:00:00.000 +0100	snow	-2.172222	-10.044444	0.90	37.4808	18.0	1.4651	0.0	1014.15	Breezy starting overnight continuing until aft...
Breezy and Mostly Cloudy	2006-04-12 08:00:00.000 +0200	rain	6.144444	1.494444	0.93	30.8637	349.0	10.5455	0.0	1004.10	Foggy overnight and breezy in the morning.
Breezy and Overcast	2006-08-31 10:00:00.000 +0200	rain	15.944444	15.944444	0.68	33.9066	289.0	11.2700	0.0	1016.52	Mostly cloudy until night and breezy in the af...
Breezy and Partly Cloudy	2006-08-21 00:00:00.000 +0200	rain	21.966667	21.966667	0.72	30.8315	310.0	15.8263	0.0	1015.64	Mostly cloudy starting overnight.
Clear	2006-04-18 07:00:00.000 +0200	rain	8.688889	8.688889	0.93	1.4329	290.0	5.8443	0.0	1012.96	Partly cloudy until night.
Dangerously Windy and Partly Cloudy	2007-01-29 13:00:00.000 +0100	rain	8.944444	3.483333	0.49	63.8526	307.0	11.4471	0.0	1009.05	Mostly cloudy throughout the day and windy sta...
Drizzle	2012-04-12 10:00:00.000 +0200	rain	14.716667	14.716667	0.59	24.7940	248.0	11.1090	0.0	1002.78	Light rain in the morning.
Dry	2007-07-20 10:00:00.000 +0200	rain	34.055556	33.005556	0.28	13.8138	179.0	9.9820	0.0	1014.03	Partly cloudy starting in the afternoon contin...
Dry and Mostly Cloudy	2012-08-31 14:00:00.000 +0200	rain	33.672222	31.388889	0.17	19.8352	149.0	10.3523	0.0	1011.83	Partly cloudy throughout the day.
Dry and Partly Cloudy	2007-04-23 13:00:00.000 +0200	rain	20.072222	20.072222	0.17	2.8014	310.0	9.9820	0.0	1025.13	Partly cloudy starting in the morning continui...
Foggy	2006-04-11 18:00:00.000 +0200	rain	10.911111	10.911111	0.86	22.3951	311.0	2.6565	0.0	1004.61	Foggy in the evening.
Humid and Mostly Cloudy	2006-08-04 14:00:00.000 +0200	rain	20.894444	20.894444	0.82	8.8711	342.0	11.3183	0.0	1005.05	Mostly cloudy until night.
Humid and Overcast	2010-08-06 13:00:00.000 +0200	rain	24.055556	24.055556	0.89	10.0786	170.0	6.1663	0.0	1008.56	Partly cloudy throughout the day.
Humid and Partly Cloudy	2006-07-02 15:00:00.000 +0200	rain	22.661111	22.661111	0.84	15.5204	1.0	11.1251	0.0	1017.99	Mostly cloudy throughout the day.
Light Rain	2012-04-12 09:00:00.000 +0200	rain	12.011111	12.011111	0.75	12.1716	230.0	11.1090	0.0	1002.70	Light rain in the morning.
Mostly Cloudy	2006-04-01 02:00:00.000 +0200	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
Overcast	2006-04-10 22:00:00.000 +0200	rain	14.422222	14.422222	0.58	20.0123	159.0	15.8263	0.0	1007.85	Mostly cloudy throughout the day.
Partly Cloudy	2006-04-01 00:00:00.000 +0200	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
Rain	2016-10-21 23:00:00.000 +0200	rain	9.738889	9.738889	0.96	4.2826	286.0	4.6529	0.0	1014.10	Rain throughout the day.
Windy	2008-01-28 02:00:00.000 +0100	rain	4.994444	-0.866667	0.45	42.5362	340.0	15.8263	0.0	1019.01	Breezy starting overnight continuing until mor...
Windy and Dry	2012-04-29 12:00:00.000 +0200	rain	27.222222	26.344444	0.24	40.2500	150.0	9.9820	0.0	1020.20	Partly cloudy starting in the morning continui...

Windy and Foggy	2006-07-08 18:00:00.000 +0200	rain	19.633333	19.633333	0.76	45.9333	90.0	2.3506	0.0	1014.92	Partly cloudy starting in the afternoon contin...
Windy and Mostly Cloudy	2007-02-01 13:00:00.000 +0100	rain	10.050000	10.050000	0.53	47.2857	308.0	11.0768	0.0	1015.30	Partly cloudy starting overnight continuing un...
Windy and Overcast	2006-03-12 23:00:00.000 +0100	rain	0.511111	-7.116667	0.66	45.5308	20.0	11.9784	0.0	1012.71	Foggy in the morning and breezy starting in th...
Windy and Partly Cloudy	2006-03-29 17:00:00.000 +0200	rain	12.100000	12.100000	0.57	41.6990	279.0	11.4471	0.0	1008.47	Partly cloudy throughout the day and breezy st...

```
# Cuantos grupos tenemos
print("Número de grupos:", len(grouped_df))
```

```
Número de grupos: 27
```

```
# Veamos las categorias y sus coincidencias
print(grouped_df.size())
```

```
Summary
Breezy                                54
Breezy and Dry                         1
Breezy and Foggy                      35
Breezy and Mostly Cloudy             516
Breezy and Overcast                  528
Breezy and Partly Cloudy             386
Clear                              10890
Dangerously Windy and Partly Cloudy     1
Drizzle                                39
Dry                                    34
Dry and Mostly Cloudy                  14
Dry and Partly Cloudy                  86
Foggy                                7148
Humid and Mostly Cloudy                40
Humid and Overcast                      7
Humid and Partly Cloudy                17
Light Rain                             63
Mostly Cloudy                      28094
Overcast                           16597
Partly Cloudy                      31733
Rain                                   10
Windy                                   8
Windy and Dry                           1
Windy and Foggy                         4
Windy and Mostly Cloudy                35
Windy and Overcast                     45
Windy and Partly Cloudy                67
dtype: int64
```

```
df.groupby("Summary").mean()
```

Summary	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
Breezy	7.922016	3.387654	0.637778	32.143948	233.018519	9.577115	0.0	563.917593
Breezy and Dry	21.111111	21.111111	0.260000	33.810000	240.000000	9.982000	0.0	1021.600000
Breezy and Foggy	-0.510317	-7.403492	0.938571	33.477880	160.628571	1.621960	0.0	1008.934000
Breezy and Mostly Cloudy	11.093411	8.680588	0.637054	33.386345	227.639535	11.478302	0.0	1000.622984
Breezy and Overcast	7.241614	3.492235	0.763144	33.037566	213.526515	11.067012	0.0	1002.114924
Breezy and Partly Cloudy	12.492761	9.989349	0.545803	33.532796	259.282383	11.326058	0.0	996.398212
Clear	11.925109	11.040338	0.729708	8.141352	179.180257	11.441788	0.0	951.763532
Dangerously Windy and Partly Cloudy	8.944444	3.483333	0.490000	63.852600	307.000000	11.447100	0.0	1009.050000
Drizzle	10.847578	10.011681	0.867949	10.356428	177.307692	8.069815	0.0	1014.931538
Dry	29.083660	28.273529	0.230294	14.713979	230.294118	10.250965	0.0	1016.391765
Dry and Mostly Cloudy	26.838492	25.929365	0.242143	13.667750	187.785714	10.115400	0.0	1014.872143
Dry and Partly Cloudy	26.605749	25.982235	0.240814	12.304519	224.465116	10.987501	0.0	1017.242558
Foggy	1.464035	-0.210419	0.950765	7.171649	168.668439	1.551411	0.0	1007.475207
Humid and Mostly Cloudy	20.886389	20.886389	0.874250	10.058877	153.425000	9.732852	0.0	1012.887250
Humid and Overcast	21.515079	21.515079	0.881429	9.740500	138.857143	9.052800	0.0	1014.550000
Humid and Partly Cloudy	21.568301	21.568301	0.848824	9.938435	201.647059	10.633576	0.0	1011.974118
Light Rain	10.021517	8.560317	0.888095	14.673489	180.761905	6.657989	0.0	1011.054762
Mostly Cloudy	12.629334	11.624994	0.725069	11.418404	192.049299	11.117234	0.0	1010.840591
Overcast	7.516502	5.789636	0.837232	12.027738	183.532747	9.275112	0.0	1005.632402
Partly Cloudy	16.024782	15.394033	0.648571	10.115130	190.161094	11.811517	0.0	1013.079063
Rain	10.096111	9.607222	0.947000	5.797610	211.800000	1.779050	0.0	1017.318000

Windy	6.804861	2.009028	0.572500	42.165900	319.750000	10.712537	0.0	127.376250
Windy and Dry	27.222222	26.344444	0.240000	40.250000	150.000000	9.982000	0.0	1020.200000
Windy and Foggy	11.876389	9.769444	0.900000	44.178400	155.000000	1.903825	0.0	1011.975000
Windy and Mostly Cloudy	11.834603	9.754286	0.600000	43.117640	261.428571	11.159600	0.0	979.825143
Windy and Overcast	7.932963	3.696543	0.708667	43.378409	244.311111	9.806331	0.0	1006.446000
Windy and Partly Cloudy	9.968076	6.551244	0.528806	44.610937	295.119403	11.484106	0.0	953.291194

```
df.groupby("Summary").sum()
```

Summary	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
Breezy	427.788889	182.933333	34.44	1735.7732	12583.0	517.1642	0.0	30451.55
Breezy and Dry	21.111111	21.111111	0.26	33.8100	240.0	9.9820	0.0	1021.60
Breezy and Foggy	-17.861111	-259.122222	32.85	1171.7258	5622.0	56.7686	0.0	35312.69
Breezy and Mostly Cloudy	5724.200000	4479.183333	328.72	17227.3542	117462.0	5922.8036	0.0	516321.46
Breezy and Overcast	3823.572222	1843.900000	402.94	17443.8348	112742.0	5843.3823	0.0	529116.68
Breezy and Partly Cloudy	4822.205556	3855.888889	210.68	12943.6594	100083.0	4371.8584	0.0	384609.71
Clear	129864.438889	120229.283333	7946.52	88659.3190	1951273.0	124601.0717	0.0	10364704.86
Dangerously Windy and Partly Cloudy	8.944444	3.483333	0.49	63.8526	307.0	11.4471	0.0	1009.05
Drizzle	423.055556	390.455556	33.85	403.9007	6915.0	314.7228	0.0	39582.33
Dry	988.844444	961.300000	7.83	500.2753	7830.0	348.5328	0.0	34557.32
Dry and Mostly Cloudy	375.738889	363.011111	3.39	191.3485	2629.0	141.6156	0.0	14208.21
Dry and Partly Cloudy	2288.094444	2234.472222	20.71	1058.1886	19304.0	944.9251	0.0	87482.86
Foggy	10464.922222	-1504.072222	6796.07	51262.9474	1205642.0	11089.4868	0.0	7201432.78
Humid and Mostly Cloudy	835.455556	835.455556	34.97	402.3551	6137.0	389.3141	0.0	40515.49
Humid and Overcast	150.605556	150.605556	6.17	68.1835	972.0	63.3896	0.0	7101.85
Humid and Partly Cloudy	366.661111	366.661111	14.43	168.9534	3428.0	180.7708	0.0	17203.56
Light Rain	631.355556	539.300000	55.95	924.4298	11388.0	419.4533	0.0	63696.45
Mostly Cloudy	354808.511111	326592.577778	20370.10	320788.6360	5395433.0	312327.5708	0.0	28398555.57
Overcast	124751.388889	96090.583333	13895.54	199624.3669	3046093.0	153939.0391	0.0	16690480.98
Partly Cloudy	508514.416667	488498.844444	20581.09	320983.4138	6034382.0	374814.8586	0.0	32148037.90
Rain	100.961111	96.072222	9.47	57.9761	2118.0	17.7905	0.0	10173.18
Windy	54.438889	16.072222	4.58	337.3272	2558.0	85.7003	0.0	1019.01
Windy and Dry	27.222222	26.344444	0.24	40.2500	150.0	9.9820	0.0	1020.20
Windy and Foggy	47.505556	39.077778	3.60	176.7136	620.0	7.6153	0.0	4047.90
Windy and Mostly Cloudy	414.211111	341.400000	21.00	1509.1174	9150.0	390.5860	0.0	34293.88
Windy and Overcast	356.983333	166.344444	31.89	1952.0284	10994.0	441.2849	0.0	45290.07
Windy and Partly Cloudy	667.861111	438.933333	35.43	2988.9328	19773.0	769.4351	0.0	63870.51

```
# Preguntemos por una categoria
grouped_df.get_group("Light Rain")
```

	Formatted Date	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
52689	2012-04-12 09:00:00.000 +0200	rain	12.011111	12.011111	0.75	12.1716	230.0	11.109	0.0	1002.70	Light rain in the morning.
52728	2012-04-14 00:00:00.000 +0200	rain	12.611111	12.611111	0.62	5.2647	55.0	16.100	0.0	999.78	Light rain until morning.
52729	2012-04-14 01:00:00.000 +0200	rain	11.155556	11.155556	0.85	3.5420	149.0	16.100	0.0	999.72	Light rain until morning.
52731	2012-04-14 03:00:00.000 +0200	rain	9.627778	8.111111	0.94	10.5294	14.0	16.100	0.0	998.15	Light rain until morning.
52732	2012-04-14 04:00:00.000 +0200	rain	9.588889	8.050000	0.94	10.6099	31.0	16.100	0.0	997.64	Light rain until morning.
...	...	...	...	...	...	...	...	...	...	...	...
95422	2016-10-26 02:00:00.000 +0200	rain	11.683333	11.683333	0.90	10.8836	157.0	0.000	0.0	1020.30	Rain until afternoon.
95423	2016-10-26 03:00:00.000 +0200	rain	11.238889	11.238889	0.91	9.5795	153.0	0.000	0.0	1019.79	Rain until afternoon.
95430	2016-10-26 10:00:00.000 +0200	rain	12.288889	12.288889	0.92	7.4060	8.0	0.000	0.0	1020.90	Rain until afternoon.
95431	2016-10-26 11:00:00.000 +0200	rain	13.255556	13.255556	0.90	11.6564	9.0	0.000	0.0	1021.49	Rain until afternoon.
95432	2016-10-26 12:00:00.000 +0200	rain	14.183333	14.183333	0.86	15.1340	9.0	0.000	0.0	1021.88	Rain until afternoon.

63 rows × 11 columns

```
# Creacion de una agrupación doble (agrupación anidada)
grouped_df = df.groupby(["Summary", "Precip Type"])

# Mostramos el primer registro de cada grupo
grouped_df.first()
```

Summary	Precip Type	Formatted Date	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
Breezy	rain	2006-11-05 05:00:00.000 +0100	6.088889	1.572222	0.76	29.0444	289.0	6.3434	0.0	1021.45	Overcast until night and breezy overnight.
	snow	2007-01-27 01:00:00.000 +0100	-1.805556	-8.888889	0.72	30.7832	299.0	14.5866	0.0	1022.36	Partly cloudy starting in the afternoon contin...
Breezy and Dry	rain	2012-03-18 11:00:00.000 +0100	21.111111	21.111111	0.26	33.8100	240.0	9.9820	0.0	1021.60	Partly cloudy starting in the afternoon.
Breezy and	rain	2010-05-16 09:00:00.000 +0200	9.916667	6.361111	0.94	31.7814	282.0	1.9320	0.0	993.95	Foggy starting in the morning continuing until...

```
# Vamos a crear una columna nueva que se llame TempLevel,
# que por defecto esta a 0
# Si la temperatura es positiva, le pondremos un 1
df["TempLevel"] = 0 # Valor por defecto
df.loc[df["Temperature (C)"] > 0, "TempLevel"] = 1
df
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary	TempLevel
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.	1
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.	1
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.	1
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.	1
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.	1
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.	1
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.	1
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.	1
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.	1

96453 rows × 13 columns

```
df.loc[df["Temperature (C)"] < 0, :].head(3)
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary	TempLevel
1562	2006-12-13 02:00:00.000 +0100	Foggy	snow	-0.483333	-4.150000	1.00	11.0929	219.0	0.483	0.0	1031.56	Foggy throughout the day.	0
1563	2006-12-13 03:00:00.000 +0100	Foggy	snow	-0.483333	-4.061111	0.96	10.7387	200.0	0.322	0.0	1031.47	Foggy throughout the day.	0
1564	2006-12-13 04:00:00.000 +0100	Foggy	snow	-0.922222	-3.477778	1.00	7.0679	206.0	0.161	0.0	1031.23	Foggy throughout the day.	0

```
df.loc[df["Temperature (C)"] > 0, :].head(3)
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary	TempLevel
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.	1
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.	1
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.	1

```
# Creacion de una agrupación doble (agrupación anidada)
grouped_df = df.groupby(["Summary", "TempLevel"])

# Mostramos el promedio
grouped_df.mean()
```

Summary	TempLevel	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
Breezy	0	-1.953704	-9.302778	0.700000	32.833267	162.416667	8.251250	0.0	255.370833
	1	10.743651	7.013492	0.620000	31.947000	253.190476	9.955933	0.0	652.073810
Breezy and Dry	1	21.111111	21.111111	0.260000	33.810000	240.000000	9.982000	0.0	1021.600000
Breezy and Foggy	0	-2.345679	-9.893004	0.934815	33.626937	121.888889	1.629081	0.0	1009.389259
	1	5.684028	0.998611	0.951250	32.974812	291.375000	1.597925	0.0	1007.397500
Breezy and Mostly Cloudy	0	-2.492094	-10.094658	0.730769	33.709065	146.807692	12.001312	0.0	1021.541923
	1	11.814274	9.676825	0.632082	33.389221	231.928571	11.450550	0.0	999.513000
Breezy and Overcast	0	-2.302778	-9.751587	0.770000	32.837962	133.000000	10.340800	0.0	1017.579821
	1	8.374000	5.063536	0.762331	33.061248	223.080508	11.153173	0.0	1000.280106
Breezy and Partly Cloudy	0	-3.085714	-10.946032	0.644286	34.479300	161.285714	11.230900	0.0	1026.797143
	1	12.780490	10.376019	0.543984	33.515315	261.092348	11.327816	0.0	995.836755
Clear	0	-3.914340	-6.774002	0.824339	8.655022	189.839080	9.539805	0.0	940.489332
	1	14.246494	13.651158	0.715839	8.066069	177.618130	11.720537	0.0	953.415846
Dangerously Windy and Partly Cloudy	1	9.944444	3.483333	0.400000	63.852600	307.000000	11.447100	0.0	1009.050000
Drizzle	1	10.847578	10.011681	0.867949	10.356428	177.307692	8.069815	0.0	1014.931538
Dry	1	29.083660	26.273529	0.230294	14.713979	230.294118	10.250965	0.0	1016.391765
Dry and Mostly Cloudy	1	26.838492	25.929365	0.242143	13.667750	187.785714	10.115400	0.0	1014.872143
Dry and Partly Cloudy	1	26.605749	25.982235	0.240814	12.304519	224.465116	10.987501	0.0	1017.242558
Foggy	0	-3.681614	-5.910416	0.940355	7.303974	165.482227	1.551091	0.0	1014.842062
	1	5.147262	3.869608	0.958217	7.076932	170.949112	1.551640	0.0	1002.202052
Humid and Mostly Cloudy	1	20.886389	20.886389	0.874250	10.058877	153.425000	9.732852	0.0	1012.887250
Humid and Overcast	1	21.515079	21.515079	0.881429	9.740500	138.857143	9.052800	0.0	1014.550000
Humid and Partly Cloudy	1	21.568301	21.568301	0.848824	9.938435	201.647059	10.633576	0.0	1011.974118
Light Rain	1	10.021517	8.560317	0.888095	14.673489	180.761905	6.657989	0.0	1011.054762
Mostly Cloudy	0	-2.567076	-5.932349	0.825445	10.564861	175.022269	8.912340	0.0	1018.043505
	1	13.722910	12.888469	0.717846	11.479827	193.274611	11.275904	0.0	1010.322250
Overcast	0	-3.064158	-6.370500	0.855831	10.452095	172.097692	7.010504	0.0	1018.133250
	1	9.481903	8.048431	0.833777	12.320420	185.656855	9.695773	0.0	1003.310319
Partly Cloudy	0	-3.194076	-6.302103	0.819831	9.424126	185.369186	9.888255	0.0	1017.958122
	1	17.126186	16.637406	0.638756	10.154730	190.435711	11.921736	0.0	1012.799451
Rain	1	10.096111	9.607222	0.947000	5.797610	211.800000	1.779050	0.0	1017.318000
Windy	1	6.804861	2.009028	0.572500	42.165900	319.750000	10.712537	0.0	127.376250
Windy and Dry	1	27.222222	26.344444	0.240000	40.250000	150.000000	9.982000	0.0	1020.200000
Windy and Foggy	0	-2.150000	-10.577778	0.950000	43.856400	20.000000	1.368500	0.0	1013.710000
	1	16.551852	16.551852	0.883333	44.285733	200.000000	2.082267	0.0	1011.396667
Windy and Mostly Cloudy	1	11.834603	9.754286	0.600000	43.117640	261.428571	11.159600	0.0	979.825143
Windy and Overcast	0	-1.325926	-9.422222	0.790000	43.384133	69.666667	9.429233	0.0	1015.906667
	1	8.594312	4.633598	0.702857	43.378000	256.785714	9.833267	0.0	1005.770238
Windy and Partly Cloudy	1	9.968076	6.551244	0.528806	44.610937	295.119403	11.484106	0.0	953.291194

```
grouped_df.count()
```

Summary	TempLevel	Formatted Date	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
Breezy	0	12	12	12	12	12	12	12	12	12	12	12
	1	42	42	42	42	42	42	42	42	42	42	42
Breezy and Dry	1	1	1	1	1	1	1	1	1	1	1	1
Breezy and Foggy	0	27	27	27	27	27	27	27	27	27	27	27
	1	8	8	8	8	8	8	8	8	8	8	8
Breezy and Mostly Cloudy	0	26	26	26	26	26	26	26	26	26	26	26
	1	490	490	490	490	490	490	490	490	490	490	490
Breezy and Overcast	0	56	56	56	56	56	56	56	56	56	56	56
	1	472	472	472	472	472	472	472	472	472	472	472
Breezy and Partly Cloudy	0	7	7	7	7	7	7	7	7	7	7	7
	1	379	379	379	379	379	379	379	379	379	379	379
Clear	0	1392	1392	1392	1392	1392	1392	1392	1392	1392	1392	1392
	1	9498	9371	9498	9498	9498	9498	9498	9498	9498	9498	9498
Dangerously Windy and Partly Cloudy	1	1	1	1	1	1	1	1	1	1	1	1
Drizzle	1	39	39	39	39	39	39	39	39	39	39	39
Dry	1	34	34	34	34	34	34	34	34	34	34	34
Dry and Mostly Cloudy	1	14	14	14	14	14	14	14	14	14	14	14
Dry and Partly Cloudy	1	86	86	86	86	86	86	86	86	86	86	86
Foggy	0	2982	2982	2982	2982	2982	2982	2982	2982	2982	2982	2982
	1	4166	4135	4166	4166	4166	4166	4166	4166	4166	4166	4166
Humid and Mostly Cloudy	1	40	40	40	40	40	40	40	40	40	40	40
Humid and Overcast	1	7	7	7	7	7	7	7	7	7	7	7
Humid and Partly Cloudy	1	17	17	17	17	17	17	17	17	17	17	17
Light Rain	1	63	63	63	63	63	63	63	63	63	63	63
Mostly Cloudy	0	1886	1886	1886	1886	1886	1886	1886	1886	1886	1886	1886
	1	26208	26028	26208	26208	26208	26208	26208	26208	26208	26208	26208
Overcast	0	2600	2600	2600	2600	2600	2600	2600	2600	2600	2600	2600
	1	13997	13916	13997	13997	13997	13997	13997	13997	13997	13997	13997
Partly Cloudy	0	1720	1720	1720	1720	1720	1720	1720	1720	1720	1720	1720
	1	30013	29915	30013	30013	30013	30013	30013	30013	30013	30013	30013
Rain	1	10	10	10	10	10	10	10	10	10	10	10
Windy	1	8	8	8	8	8	8	8	8	8	8	8
Windy and Dry	1	1	1	1	1	1	1	1	1	1	1	1
Windy and Foggy	0	1	1	1	1	1	1	1	1	1	1	1
	1	3	3	3	3	3	3	3	3	3	3	3
Windy and Mostly Cloudy	1	35	35	35	35	35	35	35	35	35	35	35
Windy and Overcast	0	3	3	3	3	3	3	3	3	3	3	3
	1	42	42	42	42	42	42	42	42	42	42	42
Windy and Partly Cloudy	1	67	67	67	67	67	67	67	67	67	67	67

Figura 26.

Conversión

En ocasiones, necesitamos convertir una serie de Pandas en una lista para poder **trabajar de forma nativa desde Python.**

Conversión de una columna a lista

```
temps = df['Temperature (C)']
print(type(temps)) # Como se ve, temps es una serie de panda

temps_list = df['Temperature (C)'].tolist()
print(type(temps_list))

print("\nListado de temperaturas")
for t in temps_list:
    print(round(t, 2))
```

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

Listado de temperaturas
9.47
9.36
9.38
8.29
8.76
9.22
7.73
8.77
10.82
13.77
16.02
17.14
17.8
17.33
18.88
```

Figura 27.

Conversión de todo el *dataframe* a una lista

```
df_list = df.to_dict('list')
df_list
```

```
'2006-04-10 14:00:00.000 +0200',
'2006-04-10 15:00:00.000 +0200',
'2006-04-10 16:00:00.000 +0200',
'2006-04-10 17:00:00.000 +0200',
'2006-04-10 18:00:00.000 +0200',
'2006-04-10 19:00:00.000 +0200',
'2006-04-10 20:00:00.000 +0200',
'2006-04-10 21:00:00.000 +0200',
'2006-04-10 22:00:00.000 +0200',
'2006-04-10 23:00:00.000 +0200',
'2006-04-11 00:00:00.000 +0200',
'2006-04-11 01:00:00.000 +0200',
'2006-04-11 02:00:00.000 +0200',
'2006-04-11 03:00:00.000 +0200',
'2006-04-11 04:00:00.000 +0200',
'2006-04-11 05:00:00.000 +0200',
'2006-04-11 06:00:00.000 +0200',
'2006-04-11 07:00:00.000 +0200',
'2006-04-11 08:00:00.000 +0200',
'2006-04-11 09:00:00.000 +0200',
```

Figura 28.

Conversión de todo el *dataframe* a un diccionario

```
df_dict = df.to_dict('dict')
df_dict
```

```
{'Formatted Date': {0: '2006-04-01 00:00:00.000 +0200',
  1: '2006-04-01 01:00:00.000 +0200',
  2: '2006-04-01 02:00:00.000 +0200',
  3: '2006-04-01 03:00:00.000 +0200',
  4: '2006-04-01 04:00:00.000 +0200',
  5: '2006-04-01 05:00:00.000 +0200',
  6: '2006-04-01 06:00:00.000 +0200',
  7: '2006-04-01 07:00:00.000 +0200',
  8: '2006-04-01 08:00:00.000 +0200',
  9: '2006-04-01 09:00:00.000 +0200',
  10: '2006-04-01 10:00:00.000 +0200',
  11: '2006-04-01 11:00:00.000 +0200',
  12: '2006-04-01 12:00:00.000 +0200',
  13: '2006-04-01 13:00:00.000 +0200',
  14: '2006-04-01 14:00:00.000 +0200',
  15: '2006-04-01 15:00:00.000 +0200',
  16: '2006-04-01 16:00:00.000 +0200',
  17: '2006-04-01 17:00:00.000 +0200',
  18: '2006-04-01 18:00:00.000 +0200',
```

```
# Mostrar las claves
for i in df_dict:
    print(i)
```

```
Formatted Date
Summary
Precip Type
Temperature (C)
Apparent Temperature (C)
Humidity
Wind Speed (km/h)
Wind Bearing (degrees)
Visibility (km)
Loud Cover
Pressure (millibars)
Daily Summary
TempLevel
```

Figura 29.

Conversión de todo el *dataframe* a *series*

```
df_series = df.to_dict('series')
df_series
```

```
{'Formatted Date': 0        2006-04-01 00:00:00.000 +0200
 1        2006-04-01 01:00:00.000 +0200
 2        2006-04-01 02:00:00.000 +0200
 3        2006-04-01 03:00:00.000 +0200
 4        2006-04-01 04:00:00.000 +0200
                    ...
 96448    2016-09-09 19:00:00.000 +0200
 96449    2016-09-09 20:00:00.000 +0200
 96450    2016-09-09 21:00:00.000 +0200
 96451    2016-09-09 22:00:00.000 +0200
 96452    2016-09-09 23:00:00.000 +0200
 Name: Formatted Date, Length: 96453, dtype: object,
 'Summary': 0        Partly Cloudy
 1        Partly Cloudy
 2        Mostly Cloudy
 3        Partly Cloudy
 4        Mostly Cloudy
             ...
 96448    Partly Cloudy
```

```
# Acceder a una serie en concreto:
df_series["Formatted Date"]
```

```
0        2006-04-01 00:00:00.000 +0200
1        2006-04-01 01:00:00.000 +0200
2        2006-04-01 02:00:00.000 +0200
3        2006-04-01 03:00:00.000 +0200
4        2006-04-01 04:00:00.000 +0200
                    ...
96448    2016-09-09 19:00:00.000 +0200
96449    2016-09-09 20:00:00.000 +0200
96450    2016-09-09 21:00:00.000 +0200
96451    2016-09-09 22:00:00.000 +0200
96452    2016-09-09 23:00:00.000 +0200
Name: Formatted Date, Length: 96453, dtype: object
```

Figura 30.

Exportación

Vamos a seleccionar un fragmento de nuestro CSV para **exportarlo en varios formatos**.

```
df_ordenado = df.sort_values(by=["Temperature (C)"],
                             ascending=False)
# Primeras 100 filas y primeras 5 columnas
df_final = df_ordenado.iloc[0:100, 0:5]
df_final
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)
12759	2007-07-22 15:00:00.000 +0200	Clear	rain	39.905556	37.538889
12737	2007-07-21 17:00:00.000 +0200	Partly Cloudy	rain	39.588889	37.600000
12664	2007-07-19 16:00:00.000 +0200	Partly Cloudy	rain	38.983333	37.461111
12712	2007-07-20 16:00:00.000 +0200	Partly Cloudy	rain	38.983333	36.800000
12711	2007-07-20 15:00:00.000 +0200	Partly Cloudy	rain	38.983333	36.627778
...	...	...	...	...	...
56895	2012-07-05 15:00:00.000 +0200	Partly Cloudy	rain	37.150000	36.183333
62845	2013-08-09 13:00:00.000 +0200	Partly Cloudy	rain	37.150000	36.577778
26271	2008-09-07 15:00:00.000 +0200	Dry	rain	37.150000	35.444444
26272	2008-09-07 16:00:00.000 +0200	Dry and Partly Cloudy	rain	37.150000	35.444444
62825	2013-08-08 17:00:00.000 +0200	Partly Cloudy	rain	37.150000	36.094444

100 rows × 5 columns

```
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 12759 to 62825
Data columns (total 5 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            100 non-null    object
 1   Summary                   100 non-null    object
 2   Precip Type               100 non-null    object
 3   Temperature (C)           100 non-null    float64
 4   Apparent Temperature (C)  100 non-null    float64
dtypes: float64(2), object(3)
memory usage: 4.7+ KB
```

```
df_final
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)
12759	2007-07-22 15:00:00.000 +0200	Clear	rain	39.905556	37.538889
12737	2007-07-21 17:00:00.000 +0200	Partly Cloudy	rain	39.588889	37.600000
12664	2007-07-19 16:00:00.000 +0200	Partly Cloudy	rain	38.983333	37.461111
12712	2007-07-20 16:00:00.000 +0200	Partly Cloudy	rain	38.983333	36.800000
12711	2007-07-20 15:00:00.000 +0200	Partly Cloudy	rain	38.983333	36.627778
...	...	...	...	...	...
56895	2012-07-05 15:00:00.000 +0200	Partly Cloudy	rain	37.150000	36.183333
62845	2013-08-09 13:00:00.000 +0200	Partly Cloudy	rain	37.150000	36.577778
26271	2008-09-07 15:00:00.000 +0200	Dry	rain	37.150000	35.444444
26272	2008-09-07 16:00:00.000 +0200	Dry and Partly Cloudy	rain	37.150000	35.444444
62825	2013-08-08 17:00:00.000 +0200	Partly Cloudy	rain	37.150000	36.094444

100 rows × 5 columns

Figura 31.

Exportación a CSV

```
ruta_csv = r"./EXPORT_weather.csv" # r = raw
df_final.to_csv(ruta_csv, index = False, header = True, sep = ",")
```

Figura 32.

Exportación a CSV y, posteriormente, a ZIP

```
opciones_compresion = dict(method='zip', archive_name='EXPORT_weather.csv')
df_final.to_csv('EXPORT_weather_csv.zip', index=False, header=True, sep=',', compression=opciones_compresion)
```

Figura 33.

Exportación a Excel

```
ruta_excel = r"./EXPORT_weather.xlsx"
df_final.to_excel(ruta_excel, index = False, header = True, sheet_name='Tiempo')
```

Figura 34.

Exportación a JSON

```
ruta_json = r"./EXPORT_weather.json"
df_final.to_json(ruta_json, orient="columns")
```

Figura 35.

Exportación a HTML

```
ruta_html = r"./EXPORT_weather.html"
df_final.to_html(ruta_html, index = False)
```

Figura 36.

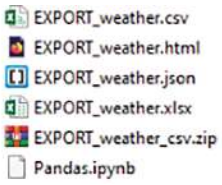


Figura 37.

2.3 Representación de datos con Matplotlib

Matplotlib es una librería de Python **especializada en la creación de gráficos en dos dimensiones**.

Figura 38. Logo de Matplotlib.

Permite **crear y personalizar los tipos de gráficos más comunes**, entre ellos:

Diagramas de barras	Histograma
Diagramas de sectores	Diagramas de caja y bigotes
Diagramas de violín	Diagramas de dispersión o puntos
Diagramas de líneas	Diagramas de áreas
Diagramas de contorno	Mapas de color

También permite la **combinación** de todos ellos.

Creación de gráficos con Matplotlib

Para crear un gráfico con Matplotlib es habitual seguir los siguientes pasos:

Importar el módulo pyplot.

Definir la **figura** que contendrá el gráfico, que es la región (ventana o página) donde se dibujará y los ejes sobre los que se dibujarán los datos. Para ello se utiliza la función subplots().

Dibujar los datos sobre los ejes. Para ello se utilizan distintas funciones dependiendo del tipo de gráfico que se quiera.

Personalizar el gráfico. Para ello existen multitud de funciones que permiten añadir un título, una leyenda, una rejilla, cambiar colores o personalizar los ejes.

Guardar el gráfico. Para ello se utiliza la función savefig().

Guardar el gráfico. Para ello se utiliza la función savefig().

```
# Importar el módulo pyplot con el alias plt
import matplotlib.pyplot as plt
# Crear la figura y los ejes
fig, ax = plt.subplots()
# Dibujar puntos
ax.scatter(x = [1, 2, 3], y = [3, 2, 1])
# Guardar el gráfico en formato png
plt.savefig('diagrama-dispersion.png')
# Mostrar el gráfico
plt.show()
```

```
<matplotlib.collections.PathCollection at 0x15ce78b8c40>
```

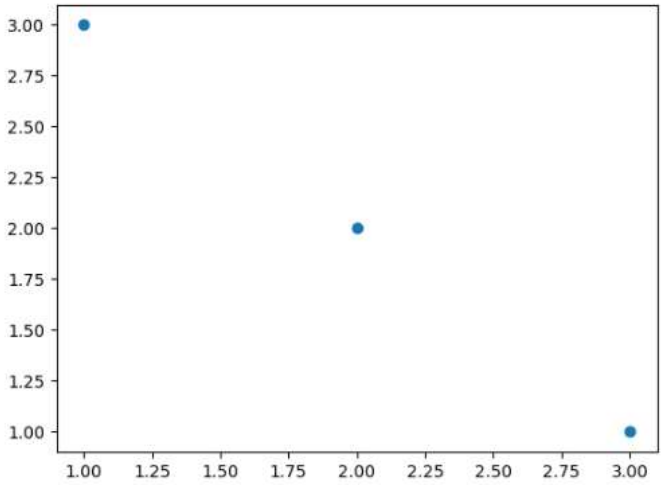


Figura 39.

Diagramas de dispersión o puntos

- scatter(x, y): Dibuja un diagrama de puntos con las coordenadas de la lista x en el eje X y las coordenadas de la lista y en el eje Y.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.scatter([1, 2, 3, 4], [1, 2, 0, 0.5])
plt.show()
```

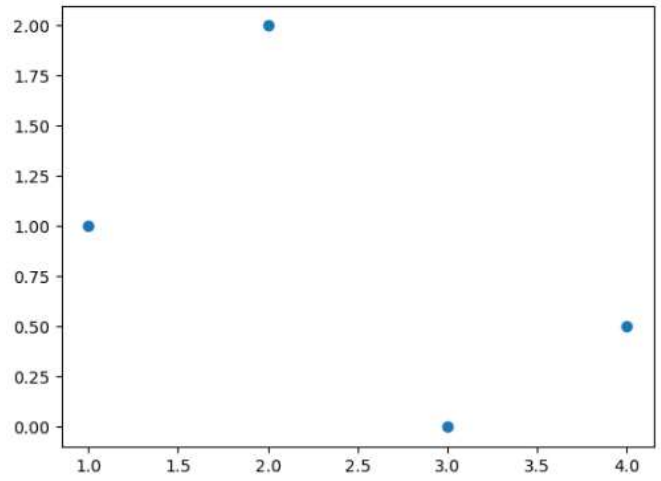


Figura 40.

Diagramas de líneas

- plot(x, y): Dibuja un polígono con los vértices dados por las coordenadas de la lista x en el eje X y las coordenadas de la lista y en el eje Y.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 2, 0, 0.5])
plt.show()
```

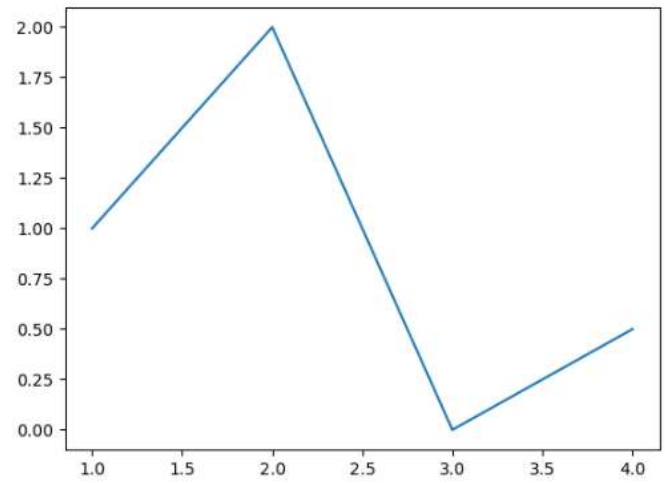


Figura 41.

Diagramas de áreas

- fill_between(x, y): Dibuja el área bajo el polígono con los vértices dados por las coordenadas de la lista x en el eje X y las coordenadas de la lista y en el eje Y.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.fill_between([1, 2, 3, 4], [1, 2, 0, 0.5])
plt.show()
```

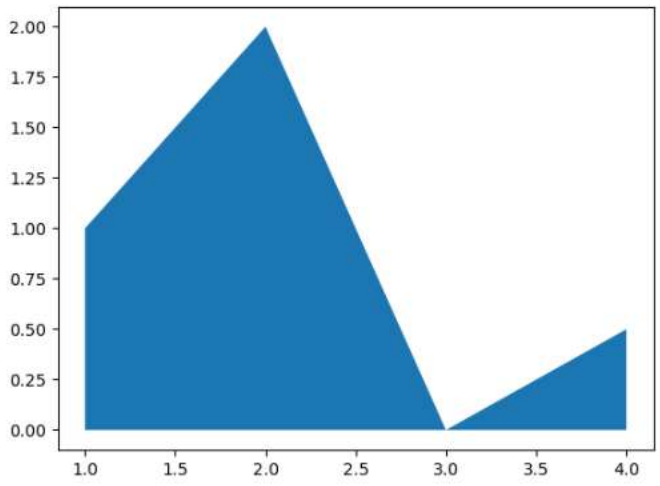


Figura 42.

Diagramas de barras verticales

- bar(x, y): Dibuja un diagrama de barras verticales donde x es una lista con la posición de las barras en el eje X, e y es una lista con la altura de las barras en el eje Y.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.bar([1, 2, 3], [3, 2, 1])
plt.show()
```

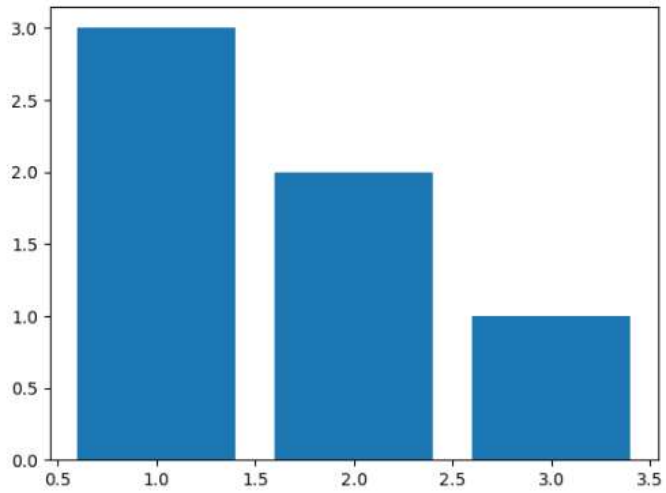


Figura 43.

Diagramas de barras horizontales

- barh(x, y): Dibuja un diagrama de barras horizontales donde x es una lista con la posición de las barras en el eje Y, e y es una lista con la longitud de las barras en el eje X.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.barh([1, 2, 3], [3, 2, 1])
plt.show()
```

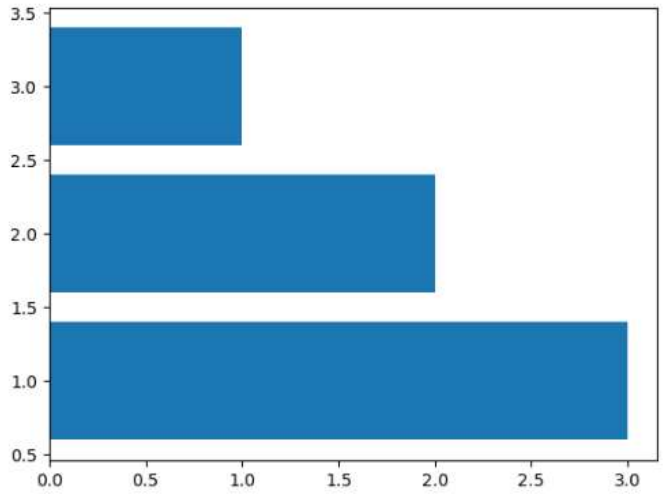


Figura 44.

Histogramas

- hist(x, bins): Dibuja un histograma con las frecuencias resultantes de agrupar los datos de la lista x en las clases definidas por la lista bins.

```
import numpy as np
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
x = np.random.normal(5, 1.5, size=1000)
ax.hist(x, np.arange(0, 11))
plt.show()
```

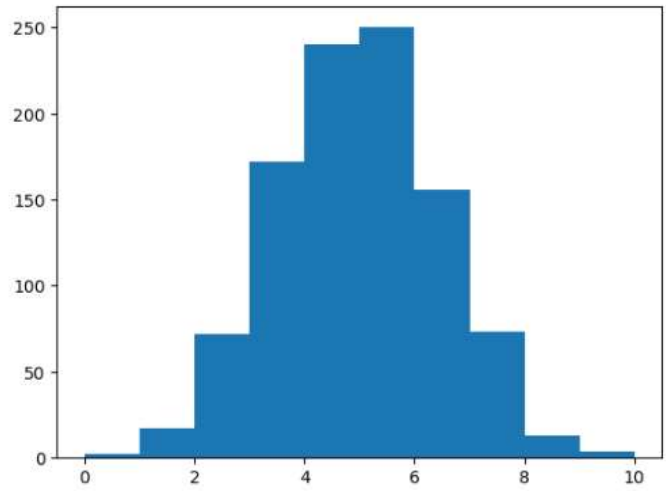


Figura 45.

Diagramas de sectores

- pie(x): Dibuja un diagrama de sectores con las frecuencias de la lista x.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.pie([5, 4, 3, 2, 1])
plt.show()
```

Figura 46.

Diagramas de caja y bigotes

- boxplot(x): Dibuja un diagrama de caja y bigotes con los datos de la lista x.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.boxplot([1, 2, 1, 2, 3, 4, 3, 3, 5, 7])
plt.show()
```

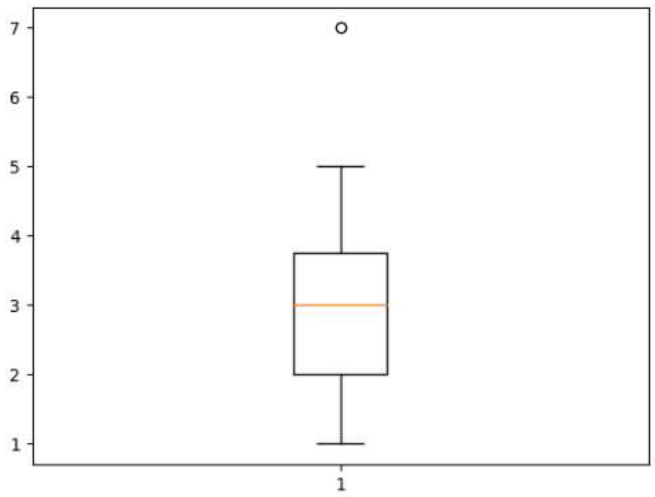


Figura 47.

Diagramas de violín

- violinplot(x): Dibuja un diagrama de violín con los datos de la lista x.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.violinplot([1, 2, 1, 2, 3, 4, 3, 3, 5, 7])
plt.show()
```

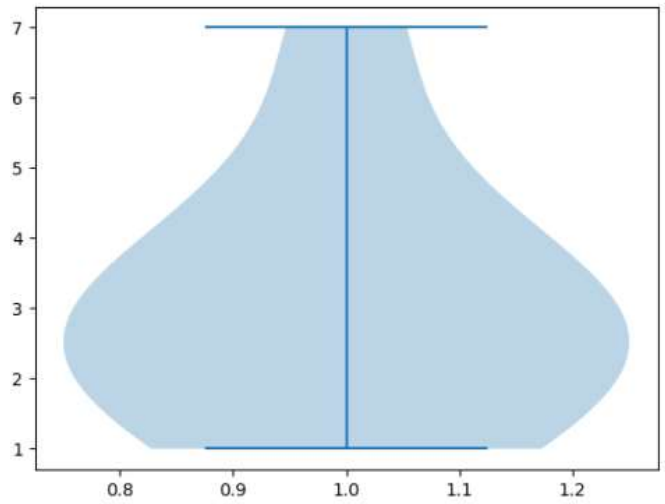


Figura 48.

Diagramas de contorno

- contourf(x, y, z): Dibuja un diagrama de contorno con las curvas de nivel de la superficie dada por los puntos con las coordenadas de las listas x, y y z en los ejes X, Y y Z respectivamente.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
x = np.linspace(-3.0, 3.0, 100)
y = np.linspace(-3.0, 3.0, 100)
x, y = np.meshgrid(x, y)
z = np.sqrt(x**2 + 2*y**2)
ax.contourf(x, y, z)
plt.show()
```

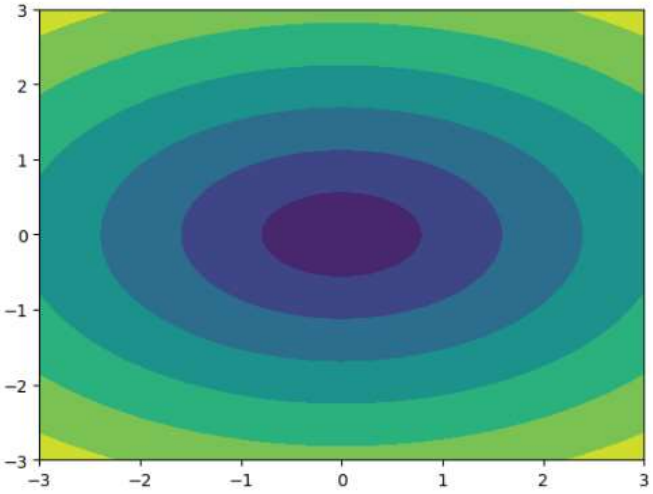


Figura 49.

Mapas de color

- imshow(x): Dibuja un mapa de color a partir de una matriz (array bidimensiona) x.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
x = np.random.random((16, 16))
ax.imshow(x)
plt.show()
```

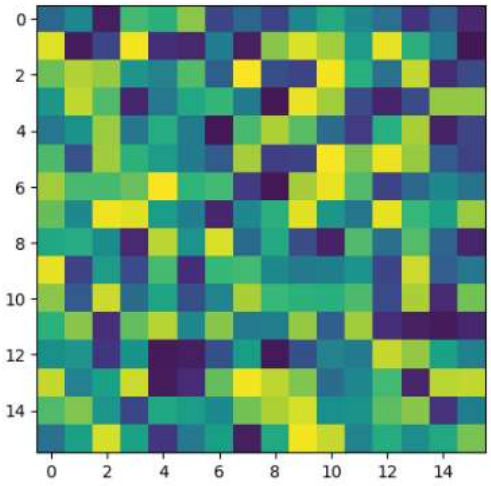


Figura 50.

- hist2d(x, y): Dibuja un mapa de color que simula un histograma bidimensional, donde los colores de los cuadrados dependen de las frecuencias de las clases de la muestra dada por las listas x e y.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
x, y = np.random.multivariate_normal(mean=[0.0, 0.0], cov=[[1.0, 0.4], [0.4, 0.5]], size=1000).T
ax.hist2d(x, y)
plt.show()
```

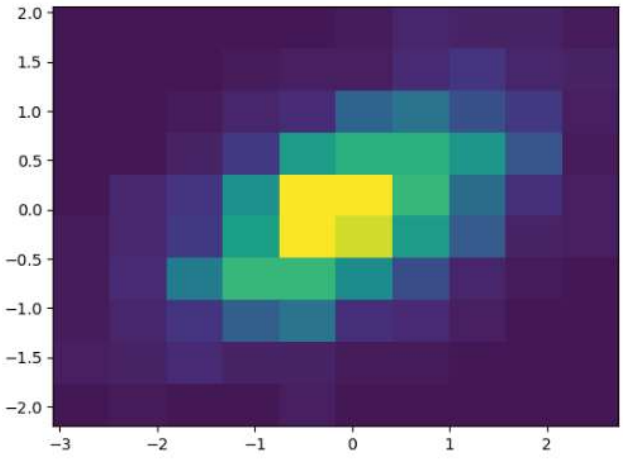


Figura 51.

Cambiar el aspecto de los gráficos

Los gráficos creados con Matplotlib son personalizables y puede cambiarse el aspecto de casi todos sus elementos. Los elementos que suelen modificarse más a menudo son:

Colores	Marcadores de puntos	Estilo de líneas	Títulos
Ejes	Leyenda	Rejilla	

Colores

Para cambiar el color de los objetos se utiliza el parámetro color = nombre-color, donde nombre-color es una cadena con el nombre del color de entre los colores disponibles.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5],
                'Barcelona':[24.5, 25.5, 26.5, 25, 26.5, 24.5, 25]}
ax.plot(dias, temperaturas['Madrid'], color = 'tab:purple')
ax.plot(dias, temperaturas['Barcelona'], color = 'tab:green')
plt.show()
```

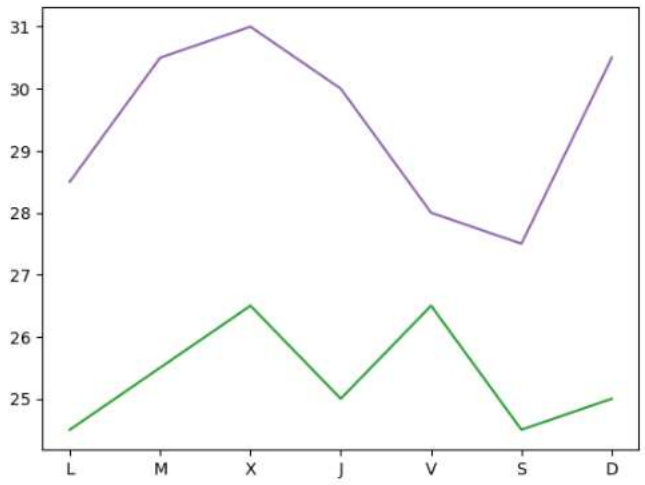


Figura 52.

Marcadores

Para cambiar la forma de los puntos marcadores se utiliza el parámetro marker = nombre-marcador donde nombre-marcador es una cadena con el nombre del marcador de entre los marcadores disponibles.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5],
                'Barcelona':[24.5, 25.5, 26.5, 25, 26.5, 24.5, 25]}
ax.plot(dias, temperaturas['Madrid'], marker = '^')
ax.plot(dias, temperaturas['Barcelona'], marker = 'o')
plt.show()
```

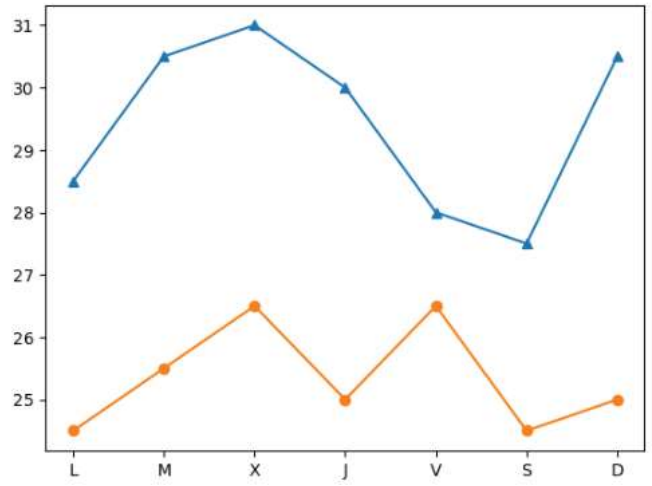


Figura 53.

Líneas

Para cambiar el estilo de las líneas se utiliza el parámetro linestyle = nombre-estilo donde nombre-estilo es una cadena con el nombre del estilo de entre los estilos disponibles.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5],
                'Barcelona':[24.5, 25.5, 26.5, 25, 26.5, 24.5, 25]}
ax.plot(dias, temperaturas['Madrid'], linestyle = 'dashed')
ax.plot(dias, temperaturas['Barcelona'], linestyle = 'dotted')
plt.show()
```

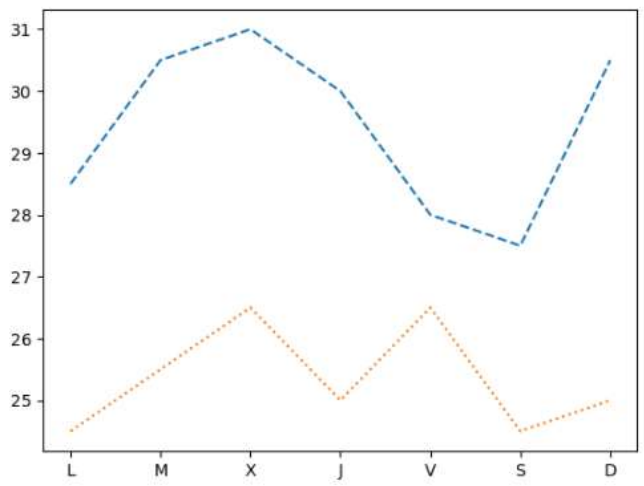


Figura 54.

Títulos

Para añadir un título principal al gráfico se utiliza el siguiente método:

- ax.set_title(titulo, loc=alineacion, fontdict=fuente) : Añade un título con el contenido de la cadena titulo a los ejes ax. El parámetro loc indica la alineación del título, que puede ser 'left' (izquierda), 'center' (centro) o 'right' (derecha), y el parámetro fontdict indica mediante un diccionario las características de la fuente (el tamaño fontisize, el grosor fontweight o el color color).

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5],
                'Barcelona':[24.5, 25.5, 26.5, 25, 26.5, 24.5, 25]}
ax.plot(dias, temperaturas['Madrid'])
ax.plot(dias, temperaturas['Barcelona'])
ax.set_title('Evolución de la temperatura diaria', loc = "left",
             fontdict = {'fontsize':14, 'fontweight':'bold', 'color':'tab:blue'})
plt.show()
```

Figura 55.

Ejes

Para cambiar el aspecto de los ejes se suelen utilizar los siguientes métodos:

- ax.set_xlabel(titulo): Añade un título con el contenido de la cadena titulo al eje x de ax. Se puede personalizar la alineación y la fuente con los mismos parámetros que para el título principal.
- ax.set_ylabel(titulo): Añade un título con el contenido de la cadena titulo al eje y de ax. Se puede personalizar la alineación y la fuente con los mismos parámetros que para el título principal.
- ax.set_xlim([limite-inferior, limite-superior]) : Establece los límites que se muestran en el eje x de ax.
- ax.set_ylim([limite-inferior, limite-superior]) : Establece los límites que se muestran en el eje y de ax.
- ax.set_xticks(marcas) : Dibuja marcas en el eje x de ax en las posiciones indicadas en la lista marcas.
- ax.set_yticks(marcas) : Dibuja marcas en el eje y de ax en las posiciones indicadas en la lista marcas.

- ax.set_xscale(escala) : Establece la escala del eje x de ax, donde el parámetro escala puede ser 'linear' (lineal) o 'log' (logarítmica).
- ax.set_yscale(escala) : Establece la escala del eje y de ax, donde el parámetro escala puede ser 'linear' (lineal) o 'log' (logarítmica).

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5],
                'Barcelona':[24.5, 25.5, 26.5, 25, 26.5, 24.5, 25]}
ax.plot(dias, temperaturas['Madrid'])
ax.plot(dias, temperaturas['Barcelona'])
ax.set_xlabel("Días", fontdict = {'fontsize':14, 'fontweight':'bold',
                                  'color':'tab:blue'})
ax.set_ylabel("Temperatura ºC")
ax.set_ylim([20,35])
ax.set_yticks(range(20, 35))
plt.show()
```

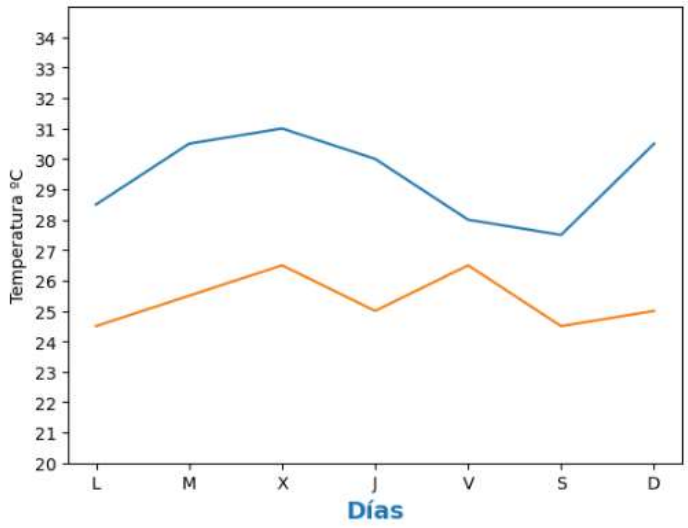


Figura 56.

Leyenda

Para añadir una leyenda a un gráfico se utiliza el siguiente método:

- ax.legend(leyendas, loc = posición) : Dibuja un leyenda en los ejes ax con los nombres indicados en la lista leyendas. El parámetro loc indica la posición en la que se dibuja la leyenda y puede ser 'upper left' (arriba izquierda), 'upper center' (arriba centro), 'upper right' (arriba derecha), 'center left' (centro izquierda), 'center' (centro), 'center right' (centro derecha), 'lower left' (abajo izquierda), 'lower center' (abajo centro), 'lower right' (abajo derecha). Se puede omitir la lista leyendas si se indica la leyenda de cada serie en la función que la dibuja mediante el parámetro label.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5],
                'Barcelona':[24.5, 25.5, 26.5, 25, 26.5, 24.5, 25]}
ax.plot(dias, temperaturas['Madrid'], label = 'Madrid')
ax.plot(dias, temperaturas['Barcelona'], label = 'Barcelona')
ax.legend(loc = 'upper right')
plt.show()
```

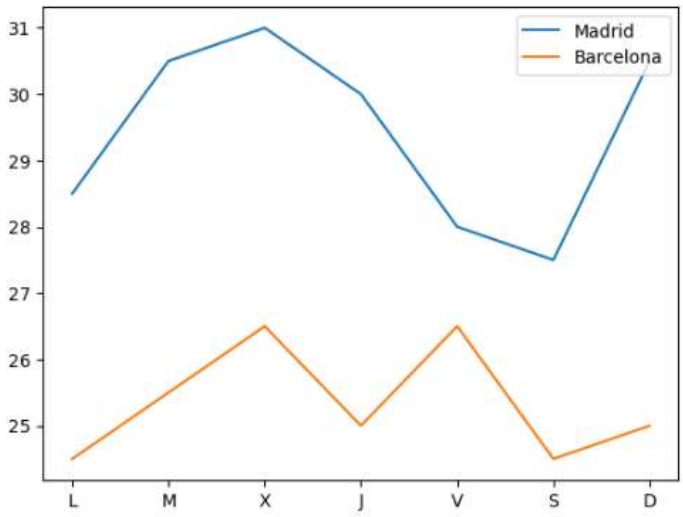


Figura 57.

Rejilla

- ax.grid(axis=ejes, color=color, linestyle=estilo) : Dibuja una rejilla en los ejes de ax. El parámetro axis indica los ejes sobre los que se dibuja la regilla y puede ser 'x' (eje x), 'y' (eje y) o 'both' (ambos). Los parámetros color y linestyle establecen el color y el estilo de las líneas de la rejilla, y pueden tomar los mismos valores vistos en los apartados de colores y líneas.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5],
                'Barcelona':[24.5, 25.5, 26.5, 25, 26.5, 24.5, 25]}
ax.plot(dias, temperaturas['Madrid'])
ax.plot(dias, temperaturas['Barcelona'])
ax.grid(axis = 'y', color = 'gray', linestyle = 'dashed')
plt.show()
```

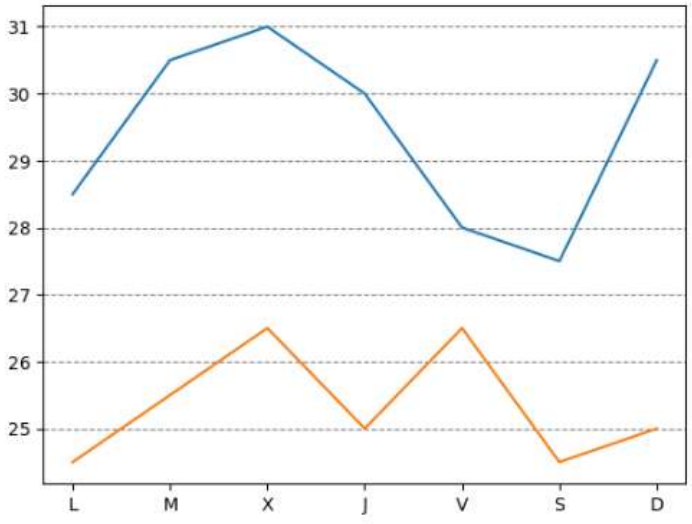


Figura 58.

Múltiples gráficos

Es posible dibujar varios gráficos en distintos ejes en una misma **figura** organizados en forma de tabla. Para ello, cuando se inicializa la **figura** y los ejes, hay que pasarle a la función subplots el número de filas y columnas de la tabla que contendrá los gráficos. Con esto los distintos ejes **se organizan en un array** y se puede acceder a cada uno de ellos a través de sus índices. Si se quiere que los distintos ejes compartan los mismos límites para los ejes se pueden pasar los parámetros sharex = True para el eje x o sharey = True para el eje y.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(2, 2, sharey = True)
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5],
                'Barcelona':[24.5, 25.5, 26.5, 25, 26.5, 24.5, 25]}
ax[0, 0].plot(dias, temperaturas['Madrid'])
ax[0, 1].plot(dias, temperaturas['Barcelona'], color = 'tab:orange')
ax[1, 0].bar(dias, temperaturas['Madrid'])
ax[1, 1].bar(dias, temperaturas['Barcelona'], color = 'tab:orange')
plt.show()
```

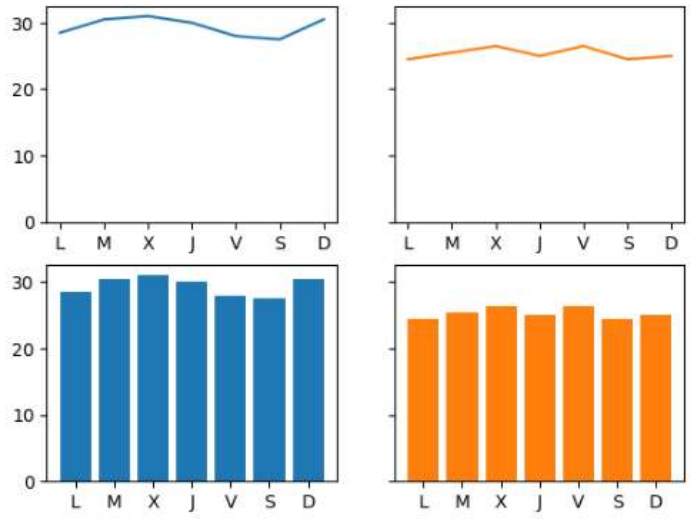


Figura 59.

Integración con Pandas

Matplotlib **se integra a la perfección con la librería Pandas**, permitiendo dibujar gráficos a partir de los datos de las series y *DataFrames* de Pandas.

- df.plot(kind=tipo, x=columnax, y=columnay, ax=ejes) : Dibuja un diagrama del tipo indicado por el parámetro kind en los ejes indicados en el parámetro ax, representando en el eje x la columna del parámetro x y en el eje y la columna del parámetro y. El parámetro kind puede tomar como argumentos 'line' (lineas), 'scatter' (puntos), 'bar' (barras verticales), 'barh' (barras horizontales), 'hist' (histograma), 'box' (cajas), 'density' (densidad), 'area' (area) o 'pie' (sectores). Es posible pasar otros parámetros para indicar el color, el marcador o el estilo de línea como se vió en los apartados anteriores.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.DataFrame({'Días':['L', 'M', 'X', 'J', 'V', 'S', 'D'],
                   'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5],
                   'Barcelona':[24.5, 25.5, 26.5, 25, 26.5, 24.5, 25]})
fig, ax = plt.subplots()
df.plot(x = 'Días', y = 'Madrid', ax = ax)
df.plot(x = 'Días', y = 'Barcelona', ax = ax)
plt.show()
```

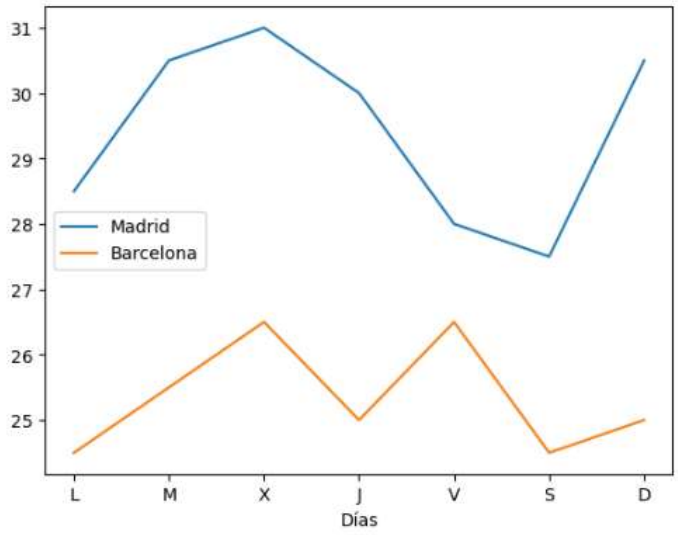


Figura 60.

Si no se indican los parámetros x e y se representa el índice de las filas en el eje x y una serie por cada columna del Dataframe. Las columnas no numéricas se ignoran.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.DataFrame({'Días':['L', 'M', 'X', 'J', 'V', 'S', 'D'],
                   'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5],
                   'Barcelona':[24.5, 25.5, 26.5, 25, 26.5, 24.5, 25]})
df = df.set_index('Días')
fig, ax = plt.subplots()
df.plot(ax = ax)
plt.show()
```

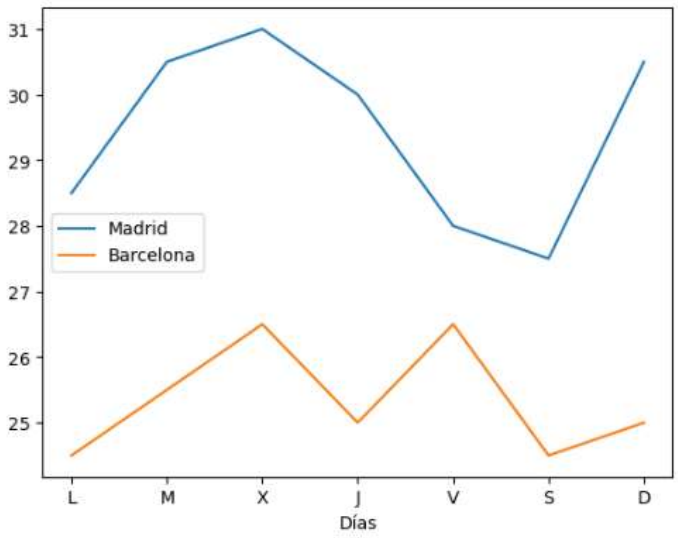


Figura 61.