Pandas

¿Qué es pandas?

- >Módulo de python para manipular conjuntos de datos
- La estructura principal es el DataFrame: Representación de datos organizados de forma tabular, en filas y columnas
- El objeto DataFrame proporciona un amplio conjunto de métodos para operar con los datos (operaciones aritméticas, filtrados, ordenaciones, etc.)
- >Para utilizar pandas debe ser importado:

import pandas

>O más comúnmente:

import pandas as pd

Instalación

>En caso de no disponer de este módulo, debemos instalarlo desde la línea de comandos con:

>pip install pandas

- ➤O desde el terminal del propio Spyder con !pip install pandas
- En caso de no disponer de la utilidad pip, se puede descargar e instalar siguiendo este tutorial:

https://tecnonucleous.com/2018/01/28/como-instalar-pip-para-python-en-windows-mac-y-linux/

Creación de un DataFrame I

- >Se puede crear un DataFrame desde distintas fuentes de datos:
 - •Desde una lista de diccionarios. Cada diccionario de la lista contiene los datos de una fila, siendo las claves los nombres de las columnas:

```
import pandas as pd

datos=[{"nombre":"paco","edad":20},
    {"nombre":"ana","edad":30},
    {"nombre":"laura","edad":32}]

df=pd.DataFrame(datos)
    print(df)

cada fila tiene un índice
numérico

dedad nombre
0 20 paco
1 45 ana
2 30 maria
3 32 laura
```

Si queremos que el índice de fila sea uno de los campos en lugar de un valor numérico:

de no Data Framo (datas)

edad nombre

df=pd.DataFrame(datos)
df.index=df["nombre"]
print(df)

edad nombre
nombre
paco 20 paco
ana 45 ana
maria 30 maria
laura 32 laura

Creación de un DataFrame II

 Desde un diccionario de listas. Las claves son los nombres de las columnas. y los valores los datos de cada columna:

24

18

27

```
import pandas as pd
                                                            ciudad temperatura
datos={"ciudad":["Málaga","Ávila",
                                                            Málaga
                                                            Ávila
           "Segovia", "Alicante"],
                                                            Segovia
        "temperatura":[24,17,18,27]}
                                                            Alicante
df=pd.DataFrame(datos)
print(df)
```

Creación de un DataFrame III

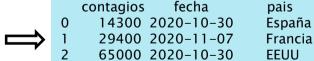
•Desde un fichero CSV. Se utiliza la función read_csv() de pandas. La primera fila del fichero será interpretada como la que contiene los nombres de columnas:

import pandas as pd
df=pd.read_csv("personas.csv", encoding="UTF-8")
print(df)



■Desde un fichero JSON. Se emplea la función read_json() de pandas. Se interpreta como una lista de diccionarios:

import pandas as pd
df=pd.read_json("contagios.json",encoding="UTF-8")
print(df)



Información sobre DataFrame

- El objeto DataFrame proporciona una serie de atributos para obtener información sobre el mismo.
- >En el ejemplo se muestran algunos de esos atributos:

Acceso al contenido de DataFrame

El atributo iloc del DataFrame proporciona acceso a su contenido. A través de un índice se accede a cada fila:

```
import pandas as pd
                                                                                ciudad
                                                                                           temperatura
datos={"ciudad":["Málaga","Ávila","Segovia","Alicante"],
                                                                                 Málaga
                                                                                               24
       "temperatura":[24,17,18,27]}
                                                                                 Ávila
                                                                                               17
df=pd.DataFrame(datos)
                                                                                                18
                                                                                 Segovia
                                                                                 Alicante
                                                                                                27
print(df)
                              ciudad
                                          Málaga
print(df.iloc[0])
                                               24
                              temperatura
```

print(df.iloc[0]["ciudad"])



Málaga

Iteración de un DataFrame

>Mediante el método *iterrows()* del DataFrame se puede interar sobre cada pareja numero_fila-serie.

Cada elemento de la serie puede ser accedido mediante índice

o nombre:

	edad	nombre
0	20	paco
1	45	ana
2	30	maria
3	32	laura

```
for k, v in df.iterrows():
    print(k,":",v[0])
```



0:20 1:45 2:30 3:32

```
for k, v in df.iterrows():
    print(k,":",v["nombre"])
```



0 : paco 1 : ana 2 : maria

3 : laura

Cálculos estadísticos

>A partir de una columna del DataFrame, podemos realizar una serie de cálculos aplicando los siguientes métodos:

- count(). total de elementos de la columna
- max(), min(). Valor máximo y mínimo de la columna, respectivamente
- sum(). Suma de todos los valores de la columna
- mean(). Valor medio de los valores de la columna

Filtrado de elementos

>Mediante el método *query()* del DataFrame obtenemos el subconjunto de elementos que cumplen la condición:

```
personas.json
```

```
[{"nombre":"Juan", "edad":30,"email":"jan@gmail.com"},
{"nombre":"Maria", "edad":40,"email":"mar@gmail.com"},
{"nombre":"Elena", "edad":35,"email":"ele@gmail.com"},
{"nombre":"Jose", "edad":38,"email":"jose@gmail.com"}
```

```
import pandas as pd
df=pd.read_json("personas.json",encoding="UTF-8")
filtro=df.query("edad>35")
print(filtro)
```



```
edad email nombre
1 40 mar@gmail.com Maria
3 38 jose@gmail.com Jose
```

>Si el valor de comparación está en una variable:

filtro=df.query("edad>@variable")

Agrupamientos

➤ Con el método *groupby()* de DataFrame podemos agrupar datos por algún valor común de columna y aplicar algún tipo de cálculo estadístico sobre el grupo

```
alumnos.json

[{"alumno":"Juan", "curso":"Java"},
{"alumno":"Pedro", "curso":"Java"},
{"alumno":"Elena", "curso":"Python"},
{"alumno":"Juan", "curso":"Net"}]
```

El campo de agrupación será el índice del resultado (Series o DataFrame) de aplicar un método al grupo

```
import pandas as pd

df=pd.read_json("alumnos.json",encoding="UTF-8")

grupos=df.groupby("alumno")

#serie con el nombre de alumno como índice

#y total de cursos por alumno como columna

print(grupos["curso"].count())

alumno

Elena 1

Juan 2

Pedro 1
```

Cálculos sobre un grupo

➤Los métodos de cálculo se pueden aplicar sobre todas las columnas de un grupo (DataFrame) o sobre una columna en concreto (Series)

empresa.csv

empleado, edad, salario, departamento Juan, 35, 1200, ventas Marcos, 39, 1300, informática Raquel, 28, 1100, ventas María, 45, 1700, informática

DataFrame

import pandas as pd edad salario df=pd.read_csv("empresa.csv",encoding="UTF-8") departamento grupo=df.groupby("departamento") informática 42.0 1500.0 Series print(grupo.mean()) #se aplica a todas las columnas 31.5 1150.0 ventas departamento print(grupo["edad"].mean()) #se aplica a solo a esa columna informática 42.0 31.5 ventas DataFrame empleado edad salario print(grupo.max()) departamento informática María 1700 Raquel 1200 ventas

Acceso al contenido del grupo

- >En el caso de un DataFrame, se puede recorrer con iterrows().
- >En el caso de un Series, se puede convertir a diccionario:

alumnos.json

```
[{"alumno":"Juan", "curso":"Java"},
{"alumno":"Pedro", "curso":"Java"},
{"alumno":"Elena", "curso":"Python"},
{"alumno":"Juan", "curso":"Net"}]
```

```
import pandas as pd

df=pd.read_json("alumnos.json",encoding="UTF-8")

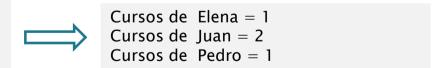
totales=df.groupby("alumno")["curso"].count()

#se convierte a diccionario para poder recorrerlo

#la clave es el índice y el valor la columna

for k,v in dict(totales).items():

print("Cursos de ",k,"=",v)
```



En ambos casos, se puede usar el atributo index para acceder a

os índices:

 $print(totales.sort_values(ascending = False).index[0])$



Juan

Ordenación y transformación de datos

Ordenación DataFrame

El método sort_values() permite ordenar un DataFrame por un campo, indicando en el atributo by el campo de ordenación:

empresa.csv

empleado, edad, salario, departamento Juan, 35, 1200, ventas Marcos, 39, 1300, informática Raquel, 28, 1100, ventas María, 45, 1700, informática 

	empleado	edad	salario	departamento
2		28	1100	ventas
0	Juan	35	1200	ventas
1	Marcos	39	1300	informática
3	María	45	1700	informática

import pandas as pd
df=pd.read_csv("empresa.csv",encoding="UTF-8")
print(df.groupby("departamento").mean().sort_values(by=["salario"]))



	edad s	salario
departamento		
ventas	31.5	1150.0
informática	42.0	1500.0

El método se puede aplicar a un objeto *Series*, sin el *by*.

 $print(df.group by ("departamento")["salario"].mean(). \textbf{sort_values()})$



1150
1500

Transformación de columnas

>Mediante el método map() de DataFrame es posible transformar los datos de una columna en otros, resultantes de la aplicación de una función:

empresa.csv

empleado, edad, salario, departamento Juan, 35, 1200, ventas Marcos, 39, 1300, informática Raquel, 28, 1100, ventas María, 45, 1700, informática

Incrementa en un 10% el salario de todos los empleados

def multi(valor):
 return valor*1.10
df=pd.read_csv("empresa.csv",encoding="UTF-8")
df["salario"]=df["salario"].map(multi)
print(df)



en	npleado	edad	salario	departamento
0	Juan	35	1320.0	ventas
1	Marcos	39	1430.0	informática
2	Raquel	28	1210.0	ventas
3	María	45	1870.0	informática

Graficos

- DataFrame dispone de un método *plot()* para representar el DataFrame o agrupamiento de DataFrame, en forma gráfica.
- >A través del atributo *kind*, se establece el tipo de gráfico

```
Cursos.json
[{"curso":"Java","precio":140},
{"curso":"Spring","precio":210},
{"curso":"Python","precio":160},
{"curso":"Angular","precio":90}]
```

```
import pandas as pd
df=pd.read_json("cursos.json",encoding="UTF-8")
df.index=df["curso"]
print(df["precio"].plot(kind="bar"))
```

