



PYTHON - PANDAS

Antoni Oller Arcas

antoni.oller@upc.edu



Barcelona 6 de Abril de 2024

Planificació

03/02	04/02	05/02	06/02	07/02
Toni	Jesus	Juan	Juan	Toni
INTRO	B3. Business Intelligence: reports i panells		Processar dades en entorns distribuïts	Cuestionario
Pandas		Pandas	Proyecto	Examen práctico
		Proyecto		Proyecto

Análisis de datos

Procesos para extraer información útil a partir de datos en bruto.

Ciclo de Vida del Análisis de Datos:

- Recopilación de Datos: bases de datos, APIs, archivos, sensores, etc.
- Limpieza y Preparación: Tratamiento de valores nulos, errores y datos inconsistentes
- Modelado e Interpretación
- Comunicación de Resultados: Visualizaciones claras, informes, dashboards y toma de decisiones



Excel

python: características

- Simplicidad y legibilidad:
- No necesita llaves {} ni puntos y comas.
- Utiliza la indentación para definir bloques de código.
- Multiparadigma:
 - Imperativa
 - Orientada a objetos (OO)
 - Funcional
- Portabilidad e interpretación.
- Gestión automática de memoria.
- Biblioteca estándar y ecosistema amplio.



¿Qué es Google Colab?

- Google Colab (o Colaboratory) es un entorno gratuito de cuadernos Jupyter.
- No requiere configuración, se ejecuta completamente en la nube.
- Ofrece acceso gratuito a GPUs para aprendizaje automático (machine learning).
- Se integra de forma sencilla con Google Drive.



Te damos la bienvenida a Colab

colab.research.google.com/?hl=es

Te damos la bienvenida a Colaboratory

Archivo Editar Ver Insertar

Índice

- Primeros pasos
- Ciencia de datos
- Aprendizaje automático
- Más recursos
- Ejemplos destacados
- Sección

Abrir cuaderno

Ejemplos >





















Recientes >

Google Drive >

GitHub >

Subir >

Buscar cuadernos

Título	Abierto por última vez	Abierto por primera vez	
 Sensor map.ipynb	23 de octubre	22 de octubre	 
 Te damos la bienvenida a Colaboratory	23 de octubre	28 oct 2022	
 Untitled1.ipynb	19 de octubre	19 de octubre	 
 Pinhole camera.ipynb	10 de octubre	10 de octubre	 
 Getting Started with OpenCV	9 de septiembre	8 de septiembre	 
 0 Python OpenCV.ipynb	9 de septiembre	8 de septiembre	 
 Webodm.ipynb	6 de julio	6 de julio	 

+ Nuevo cuaderno

Cancelar

Celdas de Código vs Celdas de Texto

Celdas de Código:

- Se utilizan para escribir y ejecutar código.
- Los resultados se muestran directamente debajo del código.

Celdas de Texto:

- Se utilizan para añadir notas o documentación.
- Admiten markdown para el formato del texto.

Untitled2.ipynb - Colaboratory

colab.research.google.com/drive/1sSng1XWlwC_oD-vuAWP5c679ynbpgQto#scrollTo=6Mu0HOFmAl6u

E-learning-UNMA... ChatGPT Clipchamp Designer Apprendre à lire le... How educators ca... Minecraft: Educati... Microsoft account Learn NEAR on Yo... How to check batt...

Untitled2.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda [Se han guardado todos los cambios](#)

+ Código + Texto

RAM Disco

Welcome to Google Collab

jupyter CO

```
a=200
b=100
a+b
```

300

0 s completado a las 6:47

Uso de Python con Google Colab

- Google Colab es compatible con Python 2.x y 3.x.
- El entorno predeterminado es Python 3.
- Simplemente escribe código Python en una celda de código y ejecútalo.
- Los resultados, gráficos y widgets interactivos se muestran en tiempo real.

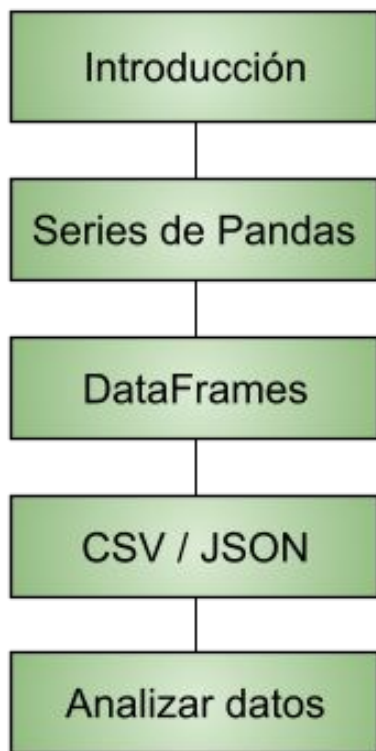


<https://github.com/pandas-dev/pandas>

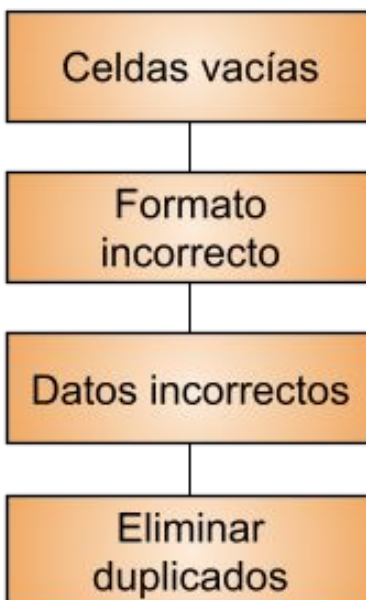
Paquete de Python que proporciona estructuras de datos, flexibles para hacer que trabajar con datos 'relacionales' o 'etiquetados'

- Realizar análisis de datos prácticos
- Herramienta de análisis/manipulación de datos de código abierto
- Estructuras de datos: DataFrames (tablas) y Series (columnas/ filas individuales)
- Manipulación de datos
- Integración: numpy, matplotlib, bases de datos
- Carga y exportación de datos: CSV, JSON, Excel, ...

Básico



Limpiando datos



Avanzado



Series de Pandas

- Columna en una tabla
- Array unidimensional que contiene datos de cualquier tipo
- <https://pandas.pydata.org/pandas-docs/stable/reference/series.html>

```

▶ import pandas as pd

precipitation = [1, 7, 2]

myvar = pd.Series(precipitation)

print(myvar)
print("=====")
print(myvar[0])

```

```

➞ 0    1
   1    7
   2    2
dtype: int64
=====
1

```

Series de Pandas

Series de Pandas

- **Etiquetas:** 'index' permite crear etiquetas

```
[135] import pandas as pd

precipitation = [15.4, 70, 2, 24]

myvar = pd.Series(precipitation, index = ["avgAirTemperature", "precipitation", "minAirTemperature", "maxAirTemperature"])

print(myvar)
print("=====")
print(myvar["avgAirTemperature"])
print(myvar["precipitation"])
print(myvar["minAirTemperature"])
print(myvar["maxAirTemperature"])
```

```
avgAirTemperature    15.4
precipitation        70.0
minAirTemperature     2.0
maxAirTemperature    24.0
dtype: float64
=====
15.4
70.0
2.0
24.0
```

Series de Pandas

- Series de pandas a partir de un **diccionario**

```
[136] import pandas as pd

precipitation = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(precipitation)

print(myvar)

print("=====")

myvar2 = pd.Series(precipitation, index = ["day1", "day2"])

print(myvar2)
```

```
day1    420
day2    380
day3    390
dtype: int64
=====
day1    420
day2    380
dtype: int64
```

Conjuntos de datos (**DataFrames**)

```
[147] import pandas as pd

mydataset = {
    'day': ["1", "2", "3"],
    'precipitation': [3, 7, 2]
}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

	day	precipitation
0	1	3
1	2	7
2	3	2

Tablas multidimensionales
(**DataFrames**)

Una serie es como una
columna, un DataFrame es
toda la tabla.

Especificación:

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

DataFrames

Conjuntos de datos (DataFrames)

```
import pandas as pd

data = {
    "precipitation": [420, 380, 390],
    "avgAirTemperature": [50, 40, 45]
}

myvar = pd.DataFrame(data)

print(myvar)

print("=====")

print(myvar.loc[0])
print("-----")
print(myvar.loc[1])
print("-----")
print(myvar.loc[2])
print("-----")
print(myvar.loc[[0, 2]])
```

	precipitation	avgAirTemperature
0	420	50
1	380	40
2	390	45

LocateRow (loc)

Permite retornar uno o más filas.

Shape (shape)

Proporciona la 'forma' (filas y columnas) de un DataFrame

Groupby (groupby)
agrupar los datos en base a un criterio

Conjuntos de datos (DataFrames)

```
[138] import pandas as pd

data = {
    "precipitation": [420, 380, 390],
    "avgAirTemperature": [50, 40, 45]
}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
print(df.loc["day1"])
print("-----")
print(df.loc["day2"])
print("-----")
print(df.loc["day3"])
print("-----")
print(df.loc[["day1", "day3"]])
```

	precipitation	avgAirTemperature
day1	420	50
day2	380	40
day3	390	45

Named Indexes

‘index’ permita crear nuevas etiquetas

Locate Named Indexes

‘loc’: Permite retornar uno o más filas.

Conjuntos de datos (DataFrames) cargados desde ficheros (JSON/CSV)

```
import pandas as pd

#pd.options.display.max_rows = 9999
df = pd.read_csv('weatherData.csv')

#print(df)
print("-----")
print(df.info())
print("-----")
print(df.head(10))
print("-----")
# last 5 rows
print(df.tail())
```

```
4      41.72
5      41.72
6      41.72
7      41.72
8      41.72
9      41.72
```

	weatherStation.location.coordinates.1	avgAirTemperature	precipitation	\
0	1.840278	18.4	0.0	
1	1.840278	17.0	0.0	
2	1.840278	14.9	0.2	
3	1.840278	14.0	0.0	
4	1.840278	11.8	0.0	
5	1.840278	12.2	0.0	
6	1.840278	12.5	0.0	
7	1.840278	13.6	0.0	
8	1.840278	14.4	0.0	
9	1.840278	15.6	0.0	

CSV / JSON

Pandas puede cargar datasets que están en ficheros (CSV, JSON) hacia **DataFrames**

Conjuntos de datos (DataFrames) : Analizando DataFrames

```
import pandas as pd

#pd.options.display.max_rows = 9999
df = pd.read_csv('weatherData.csv')

#print(df)
print("-----")
print(df.info())
print("-----")
print(df.head(10))
print("-----")
# last 5 rows
print(df.tail())
```

RangeIndex: 3487 entries, 0 to 3486

Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	lastUpdated	3487 non-null	object
1	weatherStation.name	3487 non-null	object
2	weatherStation.province	3487 non-null	object
3	weatherStation.location.coordinates.0	3487 non-null	float64
4	weatherStation.location.coordinates.1	3487 non-null	float64
5	avgAirTemperature	3458 non-null	float64
6	precipitation	3480 non-null	float64
7	minAirTemperature	3468 non-null	float64
8	maxAirTemperature	3458 non-null	float64

dtypes: float64(6), object(3)

memory usage: 245.3+ KB

None

```
-----
      lastUpdated weatherStation.name weatherStation.province \
0  2023-11-16T20:38:37.792Z      MANRESA      BARCELONA
1  2023-11-16T20:38:37.820Z      MANRESA      BARCELONA
2  2023-11-16T20:38:37.823Z      MANRESA      BARCELONA
3  2023-11-16T20:38:37.825Z      MANRESA      BARCELONA
4  2023-11-16T20:38:37.827Z      MANRESA      BARCELONA
5  2023-11-16T20:38:37.830Z      MANRESA      BARCELONA
```

Analizar datos

info: información sobre los datos

head: primeras 10 filas

tail: últimas 5 filas

Conjuntos de datos (DataFrames) : Limpiando datos

```
import pandas as pd
#pd.options.display.max_rows = 9999
df = pd.read_csv('weatherData.csv')

print("-----")
print(df.info())
print("-----")

df.insert(0, 'ID', range(1, 1 + len(df)))

print("-----")
print(df.info())
print("-----")

print(df.loc[236])
print(df.loc[251])
print(df.loc[682])
print(df.loc[1220])
print(df.loc[1221])
print(df.loc[1322])

df["avgAirTemperature"].fillna(999, inplace=True)
df["precipitation"].fillna(999, inplace=True)
df["minAirTemperature"].fillna(999, inplace=True)
df["maxAirTemperature"].fillna(999, inplace=True)
print("-----")
print(df.loc[236])
print(df.loc[251])
print(df.loc[682])
print(df.loc[1220])
print(df.loc[1221])
print(df.loc[1322])
```

Limpiando Datos

- Nueva columna ID
- Renombrar una columna (df.rename(old, new))
- Reemplazar un valor no válido a un valor determinado (999 en el ejemplo) o por otros valores: media, mediana, moda.

avgAirTemperature
NaN

avgAirTemperature
999.0

Conjuntos de datos (DataFrames) : Limpiando datos

```
meanAirTmp = df["avgAirTemperature"].mean()
meanPrecipitation = df["precipitation"].mean()
meanMinAirTemperature = df["minAirTemperature"].mean()
meanMaxAirTemperature = df["maxAirTemperature"].mean()

print("-----")
df["avgAirTemperature"] = df["avgAirTemperature"].fillna(meanAirTmp)
df["precipitation"] = df["precipitation"].fillna(meanPrecipitation)
df["minAirTemperature"] = df["minAirTemperature"].fillna(meanMinAirTemperature)
df["maxAirTemperature"] = df["maxAirTemperature"].fillna(meanMaxAirTemperature)
```

- Reemplazar un valor no válido a un valor determinado: media

avgAirTemperature NaN

Limpiando
Datos

Conjuntos de datos (DataFrames) : Limpiando datos

```
##  
## Median ([2, 4, 6, 8]) = (4 + 6) / 2 = 5.  
##  
medianAirTmp = df["avgAirTemperature"].median()  
medianPrecipitation = df["precipitation"].median()  
medianMinAirTemperature = df["minAirTemperature"].median()  
medianMaxAirTemperature = df["maxAirTemperature"].median()  
  
df["avgAirTemperature"] = df["avgAirTemperature"].fillna(medianAirTmp)  
df["precipitation"] = df["precipitation"].fillna(medianPrecipitation)  
df["minAirTemperature"] = df["minAirTemperature"].fillna(medianMinAirTemperature)  
df["maxAirTemperature"] = df["maxAirTemperature"].fillna(medianMaxAirTemperature)
```

- Reemplazar un valor no válido a un valor determinado: mediana

avgAirTemperature NaN

Limpiando
Datos

Conjuntos de datos (DataFrames) : Limpiando datos

```
###
### Mode ([1, 2, 2, 3, 4, 4, 4] = 4
###
modeAirTmp = df["avgAirTemperature"].mode()[0]
modePrecipitation = df["precipitation"].mode()[0]
modeMinAirTemperature = df["minAirTemperature"].mode()[0]
modeMaxAirTemperature = df["maxAirTemperature"].mode()[0]

print("-----")
print("modifiquem")
df["avgAirTemperature"] = df["avgAirTemperature"].fillna(modeAirTmp)
df["precipitation"] = df["precipitation"].fillna(modePrecipitation)
df["minAirTemperature"] = df["minAirTemperature"].fillna(modeMinAirTemperature)
df["maxAirTemperature"] = df["maxAirTemperature"].fillna(modeMaxAirTemperature)
```

- Reemplazar un valor no válido a un valor determinado: moda

avgAirTemperature NaN

Limpiando
Datos

Conjuntos de datos (DataFrames) : Limpiando datos

```
df['lastUpdated'] = pd.to_datetime(df['lastUpdated'])
```

```
## 2 decimales para lat / long
```

```
df['weatherStation.location.coordinates.0'] = df['weatherStation.location.coordinates.0'].round(2)
```

```
df['weatherStation.location.coordinates.1'] = df['weatherStation.location.coordinates.1'].round(2)
```

Reemplazar un formato incorrecto: fecha

```
lastUpdated 2023-11-16T20:38:37.792Z
```

```
lastUpdated 2023-11-16 20:38:38.276000+00:00
```

Número de decimales:

```
lat: 41.72
```

```
long: 1.840278
```

```
lat: 41.72
```

```
long: 1.84
```

Formato
incorrecto

Conjuntos de datos (DataFrames) : Limpiando datos

```
df.loc[236, 'avgAirTemperature']=20  
df.loc[236, 'precipitation']=200  
df.loc[236, 'minAirTemperature']=5  
df.loc[236, 'maxAirTemperature']=70
```

Datos incorrectos

```
## Se modifica la máxima temperatura de el aire en el caso que supere los 50 grados  
for x in df.index:  
    if df.loc[x, "maxAirTemperature"] > 50:  
        df.loc[x, "maxAirTemperature"] = 50
```

Conjuntos de datos (DataFrames) : Limpiando datos

Eliminar
duplicados

```
df = df.rename(columns={'weatherStation.province': 'province'})

print(df.loc[236])

filas_filtradas = df.query('province == "GIRONA"')
numFilas = filas_filtradas.shape[0]
print("-----")
print(numFilas)
print("-----")
print(filas_filtradas.head(3))

for x in df.index:
    if df.loc[x, "province"] == 'GIRONA':
        df.drop(x, inplace = True)

filas_filtradas = df.query('province == "GIRONA"')
numFilas = filas_filtradas.shape[0]
print("-----")
print(numFilas)
print("-----")
```

Avanzado: Correlaciones

Correlaciones

Correlaciones

Calcula las relaciones entre cada columna en el data set: [-1- 1]:

1: correlación perfecta

0.9 buena relación. Si se aumenta un valor, es probable que el otro también aumente.

-0.9: igual que '0.9'. Si se aumenta un valor, es probable que el otro disminuya.

0.2 significa NO una buena relación

```
#pd.options.display.max_rows = 9999
df = pd.read_csv('weatherData.csv')

print(df.info())
df['lastUpdated'] = pd.to_datetime(df['lastUpdated'])

df = df.drop('weatherStation.name', axis=1)
df = df.drop('weatherStation.province', axis=1)
df = df.drop('weatherStation.location.coordinates.0', axis=1)
df = df.drop('weatherStation.location.coordinates.1', axis=1)

###
print(df.corr())
```

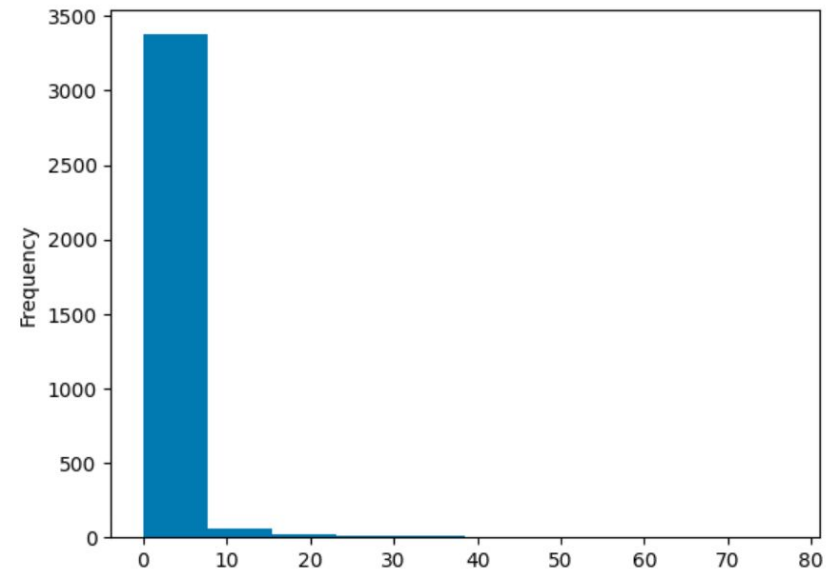
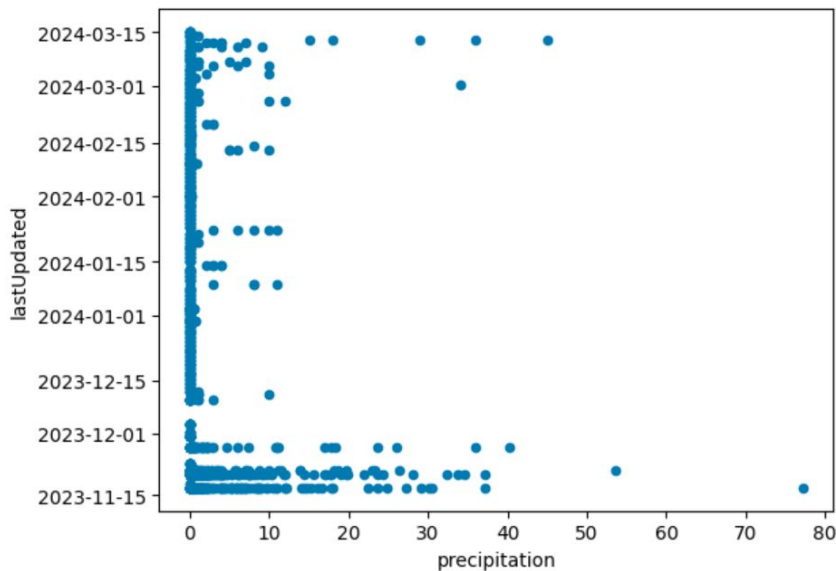
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3487 entries, 0 to 3486
Data columns (total 9 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   lastUpdated                             3487 non-null   object
1   weatherStation.name                     3487 non-null   object
2   weatherStation.province                 3487 non-null   object
3   weatherStation.location.coordinates.0   3487 non-null   float64
4   weatherStation.location.coordinates.1   3487 non-null   float64
5   avgAirTemperature                       3458 non-null   float64
6   precipitation                           3480 non-null   float64
7   minAirTemperature                       3468 non-null   float64
8   maxAirTemperature                       3458 non-null   float64
dtypes: float64(6), object(3)
memory usage: 245.3+ KB
None
```

	lastUpdated	avgAirTemperature	precipitation	\
lastUpdated	1.000000	-0.314057	-0.059228	
avgAirTemperature	-0.314057	1.000000	0.023077	
precipitation	-0.059228	0.023077	1.000000	
minAirTemperature	-0.287832	0.952472	0.047328	
maxAirTemperature	-0.278037	0.950346	-0.010075	

	minAirTemperature	maxAirTemperature
lastUpdated	-0.287832	-0.278037
avgAirTemperature	0.952472	0.950346
precipitation	0.047328	-0.010075
minAirTemperature	1.000000	0.818873
maxAirTemperature	0.818873	1.000000

Avanzado: Gráficas

Gráficas



Utiliza el módulo Pyplot de la biblioteca Matplotlib, para visualizar el diagrama en la pantalla

Ejercicios

DATASET: weatherData.csv

1. Calcular la temperatura media del aire para cada estación meteorológica.
2. Encontrar la temperatura máxima registrada en todo el dataset y la fecha en la que ocurrió.
3. Filtrar el dataset para mostrar solo los registros de una provincia específica.
4. Calcular la cantidad total de precipitación registrada en cada provincia.
5. Calcular el promedio de temperatura del aire por mes en todo el dataset.
6. Filtrar el dataset para mostrar solo los registros de una estación meteorológica específica.
7. Encontrar el día con la precipitación más alta registrada en todo el dataset y la estación meteorológica donde ocurrió.
8. Crear una visualización gráfica que muestre la relación entre la temperatura del aire y la precipitación para un rango de fechas específico.
9. Encontrar las coordenadas (latitud y longitud) de la estación meteorológica más cercana a un punto de interés dado.