01_Series

November 28, 2023

1 Pandas

¿Y qué es pandas?

Pandas es una biblioteca esencial en data science que ofrece herramientas en Python para trabajar con datos tabulares de manera eficiente. Permite cargar, limpiar, transformar y analizar datos de manera sencilla. Su estructura de datos principal, el DataFrame, se asemeja a una tabla de hoja de cálculo y facilita la exploración y manipulación de datos, siendo una herramienta fundamental para cualquier analista o científico de datos.

[6]:
```
import numpy as np
import pandas as pd
```

Y ahora un adelanto, vamos a leer un fichero excel que tiene este aspecto:

[8]:
```
df_ejemplo = pd.read_excel("./data/dataset_aviones.xlsx")
df_ejemplo.head(15)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
```

```
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
↪1\01_Series.ipynb Celda 7 line 1

----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%201/01_Series.ipynb#W6sZmlsZQ%3D%3D?
↪line=0'>1</a> df_ejemplo = pd.read_excel("./data/dataset_aviones.xlsx")
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%201/01_Series.ipynb#W6sZmlsZQ%3D%3D?
↪line=1'>2</a> df_ejemplo.head(15)


File c:
↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\excel\_bas
↪py:504, in read_excel(io, sheet_name, header, names, index_col, usecols,␣
↪dtype, engine, converters, true_values, false_values, skiprows, nrows,␣
↪na_values, keep_default_na, na_filter, verbose, parse_dates, date_parser,␣
↪date_format, thousands, decimal, comment, skipfooter, storage_options,␣
↪dtype_backend, engine_kwargs)
    502 if not isinstance(io, ExcelFile):
    503     should_close = True
--> 504     io = ExcelFile(
    505         io,
    506         storage_options=storage_options,
    507         engine=engine,
    508         engine_kwargs=engine_kwargs,
    509     )
    510 elif engine and engine != io.engine:
    511     raise ValueError(
    512         "Engine should not be specified when passing "
    513         "an ExcelFile - ExcelFile already has the engine set"
    514     )


File c:
↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\excel\_bas
↪py:1563, in ExcelFile.__init__(self, path_or_buffer, engine, storage_options,␣
↪engine_kwargs)
   1561     ext = "xls"
   1562 else:
-> 1563     ext = inspect_excel_format(
   1564         content_or_path=path_or_buffer, storage_options=storage_options
   1565     )
   1566     if ext is None:
   1567         raise ValueError(
   1568             "Excel file format cannot be determined, you must specify "
   1569             "an engine manually."
   1570         )


File c:
↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\excel\_bas
↪py:1419, in inspect_excel_format(content_or_path, storage_options)
   1416 if isinstance(content_or_path, bytes):
   1417     content_or_path = BytesIO(content_or_path)
```

```
-> 1419 with get_handle(
   1420     content_or_path, "rb", storage_options=storage_options, is_text=Fal e
   1421 ) as handle:
   1422     stream = handle.handle
   1423     stream.seek(0)

File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\common.
 ↪py:872, in get_handle(path_or_buf, mode, encoding, compression, memory_map,␣
 ↪is_text, errors, storage_options)
    863         handle = open(
    864             handle,
    865             ioargs.mode,
   (…)
    868             newline="",
    869         )
    870     else:
    871         # Binary mode
--> 872         handle = open(handle, ioargs.mode)
    873     handles.append(handle)
    875 # Convert BytesIO or file objects passed with an encoding

FileNotFoundError: [Errno 2] No such file or directory: './data/dataset_aviones
 ↪xlsx'
```

En las primeras sesiones de la unidad vamos a ver las dos clases, u objetos [o tipos] más importantes de Pandas: Series y DataFrame, y además veremos brevemente el tipo o clase Index. [¿Y por qué este orden...?]

1.1 Objetos Pandas: Series

En un nivel muy básico, los objetos Pandas pueden ser considerados como **versiones mejoradas de los arrays estructurados de NumPy en los que las filas y columnas se identifican con etiquetas en lugar de simples índices enteros.**

Como veremos en el transcurso de este capítulo, Pandas proporciona una gran cantidad de herramientas útiles, métodos y funcionalidad sobre las estructuras de datos básicas, pero **casi todo lo que sigue requerirá una comprensión de lo que son estas estructuras**

1.2 Introducción

Una `Serie` de Pandas es un array unidimensional de datos indexados. Se puede crear a partir de una lista o un array de la siguiente manera:

Una lista con indice

[9]:
```
# Heights of class
# Create series 5 values
data = pd.Series([1.60, 1.59, 1.87, 2.05, 1.75],
```

```
                  index = ["Estudiante 1", "Estudiante 2",
                           "Estudiante 3", "Estudiante 4",
                           "Estudiante 5"])
data
```

[9]:
```
Estudiante 1    1.60
Estudiante 2    1.59
Estudiante 3    1.87
Estudiante 4    2.05
Estudiante 5    1.75
dtype: float64
```

Como vemos en la salida, la `Serie` envuelve tanto una secuencia de valores como una secuencia de índices, a los que podemos acceder con los atributos `values` e `index`. Los `valores` son simplemente una matriz NumPy:

[12]:
```
data.values
```

[12]:
```
array([1.6 , 1.59, 1.87, 2.05, 1.75])
```

El `índice` es un objeto tipo array del tipo `pd.Index`(pandas index), del que hablaremos con más detalle.

[10]:
```
#atributo index
data.index
```

[10]:
```
Index(['Estudiante 1', 'Estudiante 2', 'Estudiante 3', 'Estudiante 4',
       'Estudiante 5'],
      dtype='object')
```

Al igual que con un array de NumPy, se puede acceder a los datos por el índice asociado mediante la notación de corchetes de Python:

[11]:
```
data[1]
```

```
C:\Users\victo\AppData\Local\Temp\ipykernel_18744\3862107824.py:1:
FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a
future version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  data[1]
```

[11]:
```
1.59
```

[12]:
```
data.iloc[-1]# iloc == index - loc
```

[12]:
```
1.75
```

[16]:
```
otra_serie = data[1:4].copy()
otra_serie[2] = 3.05
otra_serie.
```

```
C:\Users\victo\AppData\Local\Temp\ipykernel_18744\3501142499.py:2:
FutureWarning: Series.__setitem__ treating keys as positions is deprecated. In a
future version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To set a value by position, use `ser.iloc[pos] = value`
  otra_serie[2] = 3.05
```

[16]:
```
Estudiante 2    1.59
Estudiante 3    1.87
Estudiante 4    3.05
dtype: float64
```

[17]:
```
data[1:4:2]
```

[17]:
```
Estudiante 2    1.59
Estudiante 4    2.05
dtype: float64
```

Sin embargo, como veremos, **la `Serie` de Pandas es mucho más general y flexible que el array unidimensional de NumPy** que emula.

```
 <td style="text-align:left">
     <h3>Ejercicio pandas Series</h3>

<li>Crea un programa que te pida 5 nombres de ciudad por pantalla</li>
<li>Guarda esas ciudades en una lista</li>
<li>Crea un pandas Series a partir de la lista</li>
<li>Ponle el nombre "Ciudades"</li>
```

[ ]:
```

```

[ ]:
```

```

[ ]:
```

```

[19]:
```
listas_ciudades = []
for i in range(5):
    name = input(f"Introduzca nombre de ciudad ({(i + 1)}/5)")
    listas_ciudades.append(name)
sr_ciudades = pd.Series(listas_ciudades)
sr_ciudades.name = "Mis_Ciudades"
```

[21]:
```
print(sr_ciudades)
```

```
0    Jerez de la Frontera
1                   Cadiz
2                  Madrid
```

```
3                   Avila
4                 Linares
Name: Mis_Ciudades, dtype: object
```

[22]:
```
sr_ciudades
```

[22]:
```
0    Jerez de la Frontera
1                   Cadiz
2                  Madrid
3                   Avila
4                 Linares
Name: Mis_Ciudades, dtype: object
```

[ ]:
```

```