Clase 022 — Pandas: Series y DataFrame

Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 3, §§ 3.1–3.2 Introducing Pandas Objects. · Duración estimada: 90 min.

Clase 022 — Pandas: Series y DataFrame

> *Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 3, §§ 3.1–3.2 Introducing Pandas Objects. · Duración estimada: 90 min.*

Objetivo

Que el alumno entienda qué es una Series (ndarray + index) y un DataFrame (dict de Series alineadas por index), cómo se construyen desde 5 fuentes distintas, y por qué el index es el rasgo que distingue pandas de NumPy.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Crear Series y DataFrames desde dict, lista de tuplas, arrays NumPy, CSV y desde otro DataFrame.
2. Inspeccionar un DataFrame con head, tail, info, describe, dtypes, shape.
3. Acceder a columnas como atributo (df.col) y como key (df['col']) — y saber cuándo cada uno falla.
4. Modificar el index con set_index, reset_index, rename.
5. Convertir Series ↔ DataFrame ↔ ndarray cuando sea necesario.

Temas

#	Tema	Por qué importa
1	Series = ndarray + index	Index permite alineación automática.
2	DataFrame = dict de Series alineadas	Por eso df['col'] devuelve Series.
3	Construcción desde 5 fuentes	dict, lista de dicts, arrays, CSV, Series.
4	.loc vs .iloc vs []	Tres formas de acceso.
5	Index labels vs posición	El bug clásico cuando el index no es 0..N.
6	info y describe como first-look	Lo primero que mira un DS.

Versión profundizada — 2026

El tema moderno que vivía como complemento dentro de esta clase ahora tiene clase propia dedicada:

- Clase 032a — Polars: DataFrames modernos

Definiciones y características

Series
: Estructura 1D = ndarray + index labelled. s[label] accede por etiqueta, s.iloc[N] por posición. Característica clave: el index permite alineación automática entre Series.

DataFrame
: Estructura 2D = dict de Series que comparten el mismo index. Cada columna puede tener su propio dtype (a

diferencia de un ndarray). df['col'] devuelve Series.

Index

: Estructura que etiqueta cada fila (o columna). Puede ser entero default (RangeIndex), strings, fechas (DatetimeIndex), o jerárquico (MultiIndex). Inmutable — para cambiarlo, set_index/reset_index.

Alineación automática

: Cuando operas dos Series/DataFrames, pandas alinea por index, NO por posición. Posiciones sin match → NaN. Esto es lo que distingue pandas de NumPy.

dtype en pandas

: Cada columna tiene su dtype. Tipos clásicos: int64, float64, bool, object (strings o mixto). Modernos (NA-aware): Int64, Float64, boolean, string, category.

Dataset / recursos

Palmer Penguins (descargable con seaborn/palmerpenguins) — 344 filas × 7 columnas, públicas, sin issues de licencia. Reemplaza al iris dataset.

Ejercicios

1. Series desde dict. Crea Series con población de 5 ciudades. Accede por label y por posición.

2. DataFrame desde dict de listas. Construye DataFrame de 5 estudiantes (nombre, edad, nota). Inspecciona con info() y describe().

3. Lee Palmer Penguins. pd.read_csv desde URL pública. Reporta shape, dtypes, % de NaN por columna.

4. Index labeled. Setea species como index. Compara df.loc['Adelie'] vs df.iloc[0].

5. Alineación automática. Crea 2 Series con index parcialmente solapado. Súmalas. Observa los NaN.

Homework verificable

Notebook que: (a) carga Palmer Penguins y reporta info(), describe(), missing por col; (b) muestra los 3 métodos de acceso a una columna (df.col, df['col'], df.loc[:, 'col']); (c) cambia el index a species, vuelve a default con reset_index; (d) demuestra alineación automática sumando dos Series.

Criterio de aceptación: Carga sin error, los 3 accesos producen la misma Series, alineación produce NaN donde corresponde.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
AttributeError al hacer df.column with spa	Acceso como atributo solo funciona con nom
KeyError: 'columna' aunque está en el Data	Espacios invisibles o capitalización disti
Sumar dos Series y aparecen NaN inesperado	Index no coincide en algunos labels. Fix:
pd.read_csv infiere mal los dtypes	Pandas adivina por las primeras filas; si
DataFrame is not callable: TypeError: 'Dat	Hiciste df() por error en vez de df (parén

Preguntas frecuentes

¿df['col'] o df.col?

df['col'] siempre — funciona con cualquier nombre, no choca con métodos. df.col falla con espacios, choca con .shape, .head, etc. Solo úsalo en exploración rápida.

¿Por qué read_csv me da Unnamed: 0?

El CSV se guardó con index (df.to_csv('x.csv')). Fix: al leer, pd.read_csv('x.csv', index_col=0). O al guardar, df.to_csv('x.csv', index=False).

¿Cuándo Series y cuándo DataFrame de 1 columna?

Series es más liviana y la mayoría de operaciones la prefieren. DataFrame de 1 col cuando vas a concatenar con otros DataFrames o necesitas mantener la columna nombrada. s.to_frame() convierte.

¿copy() cuándo?

Cuando vas a modificar un subset y NO quieres afectar el original. subset = df[cond].copy(). Si solo lees, no necesitas copy — y consumes el doble de memoria.

¿Qué pasa con NaN en una columna int?

Pandas la promueve a float64 automáticamente (porque NumPy int no soporta NaN). Para mantener int: usa dtype nullable Int64 (mayúscula) que sí permite pd.NA.

¿Tengo que aprender pandas Y polars?

Por ahora, no. Pandas primero — es la base del curso, lo vas a ver en clases, libros, Stack Overflow y en cualquier código que toques. Cuando lo domines y te topes con un dataset que no entra en RAM o un pipeline lento, ahí asomate a polars (Parte 5). La transición es directa porque los conceptos (DataFrame, columnas, group_by, joins) son los mismos; lo que cambia es la sintaxis y el motor.

Referencias

- VanderPlas, cap. 3 §§ 3.1, 3.2.
- pandas user guide — DataFrame
- Palmer Penguins
- Polars user guide

Siguiente clase

Clase 023 — Pandas: indexación (loc, iloc, at, iat)

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
import numpy as np
import pandas as pd
print('pandas:', pd.__version__)
rng = np.random.default_rng(42)
```

Archivos complementarios

- notebook.ipynb