CLASE 06 - ANALISIS DE DATOS CON PYTHON

COMISION 25261 - PALAZZESI/SOSA

Resumen general

Comando	¿Qué hace?
`pd.read_csv("datos.csv")`	Lee un CSV en un DataFrame.
`df.columns`	Índice con los nombres de columnas.
`df.keys()`	Alias de `df.columns` .
`df.columns.to_list()`	Convierte los nombres de columnas a lista de Python.

Índices

Comando	¿Qué hace?
`df.index`	Muestra el **objeto índice** (etiquetas de filas).
`df.index.name`	Nombre del índice (puede ser `None`).
`df.index.names`	Lista de nombres si es **MultiIndex**.
`df.set_index(["col1","col2"])`	Usa esas columnas como **índice** (dejan de estar en `df.columns`).
`df.reset_index()`	Trae el/los índice(s) a columnas normales y crea un nuevo índice 0..n−1.

Columnas

Comando	¿Qué hace?
`df["Columna"]`	Selecciona **una** columna y devuelve una **Series**.
`df[["Columna"]]`	Devuelve un **DataFrame** con esa columna.
`df["Columna"].to_list()`	Convierte la columna a **lista** de Python.
`df["Columna"].unique()`	Devuelve un array de NumPy con los valores distintos que aparecen en la columna Sex, en orden de aparición (no ordena). Puede incluir NaN si los hay.
`df["Columna"].value_counts()`	Devuelve una Serie con la frecuencia de cada valor distinto en Age, ordenada de mayor a menor. Por defecto excluye NaN (dropna=True).

Filas

Comando	¿Qué hace?
`df.loc[n]`	Selecciona por etiqueta de índice la(s) fila(s) cuyo index == n.

Comando	¿Qué hace?
`df.iloc[n]`	Selección de la **fila n** por **posición** (0-based), sin importar cuál sea la etiqueta del índice.
`df.iloc[[n1,n2,n3]]`	Selecciona por posición las filas (índices 0-based) y devuelve un DataFrame con ellas.
`df.iloc[n]["Columna"]`	Devuelve el valor de la columna "Columna" en la fila n por posición. *(Mejor evitar chained indexing: ver notas abajo)*

Seleccionar más de una fila/columna

Comando	¿Qué hace?
`df[["Col1","Col2"]]`	Selecciona dos columnas por nombre y devuelve un DataFrame con esas columnas (en ese orden) y todas las filas
`df.loc[:, "Col1":"ColN"]`	Selecciona un subconjunto de columnas contiguas por nombre, desde Col1 hasta Coln
`df.loc[n1:n2, ["Col1","Col2","Col3"]]`	Selecciona un subconjunto de filas y columnas por etiquetas. Toma todas las filas cuyo índice va desde n1 hasta n2, inclusive, y sólo esas columnas, en ese orden. Devuelve un DataFrame.

Selección condicional / Búsqueda condicional

Comando	¿Qué hace?
`df.loc[df["Col"] > 60]`	Filas donde `Col` **> 60** (todas las columnas).
`df.loc[df["Col"] > 60, ["Name","Sex","Age"]]`	Igual, pero solo esas **3 columnas**.
`df.loc[df["Col1"].isin(["A","B"]), ["Col2","Col3"]]`	Filas donde `Col1` **es** "A" **o** "B""; devuelve `Col2` y `Col3` .
`df.loc[df["Col1"].between(10,50), ["Col2","Col3"]]`	Filas con `Col1` en **[10,50]** (inclusivo).
`df.loc[df["Col1"].str.contains("texto", case=False, na=False), ["Col2","Col3","Col4"]]`	Crea una máscara booleana que vale

Comando	¿Qué hace?
	True en las filas donde Col1 contiene la cadena "texto". Devuelve un DataFrame con solo las filas que coincidieron y solo las columnas Col2, Col3 y Col4 (en ese orden) .Por defecto contains interpreta el patrón como regex (regex=False para buscar el texto literal)

Transformaciones

Comando	¿Qué hace?
`(df["Fare"] * 1.10).round(2)`	Toma la columna numérica Fare, le aplica un +10% y redondea a 2 decimales. Devuelve una pd.Series.
`df.drop(columns=["Col1"])`	Elimina la columna Col1 y devuelve un nuevo DataFrame (no modifica df a menos que inplace=True)
`df.drop(index=2)`	Elimina la fila cuyo índice (label) es 2 y devuelve un nuevo DataFrame (no modifica df a menos que inplace=True)
`df.reset_index(drop=True)`	reconstruye el índice del DataFrame a un RangeIndex consecutivo 0..n-1. Mueve el índice actual a una columna y crea uno nuevo. drop=True evita crear esa columna: descarta el índice viejo.Devuelve un nuevo DataFrame (a menos que inplace=True).
`df.transpose()`	Intercambia filas y columnas. El índice actual de df se convierte en nombres de columnas del transpuesto, y df.columns pasa a ser el índice. (`df.T`).

Notas útiles

- Para evitar *chained indexing*, usar `df.loc[...]` o `df.iat/at` al acceder a un escalar (ej.: `df.iat[0, df.columns.get_loc("Age")]`).
- En `loc`, los slices por **label** son **inclusivos**; en `iloc` (posiciones) son como Python (`stop` excluido).
- `isin` verifica pertenencia exacta (sensible a mayúsc/minúsc). Para "contiene", usar `str.contains`.