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

Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 3 § 3.3 Data Indexing and Selection. · Duración estimada: 75 min.

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

> *Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 3 § 3.3 Data Indexing and Selection. · Duración estimada: 75 min.*

Objetivo

Que el alumno domine los 4 indexers de pandas y elija el correcto según el caso. El bug "SettingWithCopyWarning" y el bug del slicing por label inclusivo nacen aquí — saber qué indexer usar evita ambos.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Usar .loc[row_label, col_label] para acceso por etiqueta (inclusivo en slicing).
2. Usar .iloc[row_pos, col_pos] para acceso por posición entera (exclusivo, como Python).
3. Usar .at / .iat para acceso a un único valor (más rápido que loc/iloc).
4. Evitar SettingWithCopyWarning usando .loc para asignar en una vista.
5. Filtrar filas con boolean mask dentro de .loc: df.loc[df['edad'] > 30, 'nombre'].

Temas

#	Tema	Por qué importa
1	[] directo: shortcut con quirks	Columnas → Series; filas → KeyError.
2	.loc: por label, slicing inclusivo	El indexer principal del 80% del tiempo.
3	.iloc: por posición, slicing exclusivo (co	Cuando no te importa el label.
4	.at / .iat: single value	Optimizado para 1 celda — útil en loops.
5	Mask + loc para filtros con asignación	df.loc[mask, 'col'] = valor.
6	SettingWithCopyWarning: qué es y cómo evit	Usar .loc para asignar.

Definiciones y características

.loc[fila, col]

: Acceso por etiqueta. Slicing es inclusivo en ambos extremos (df.loc['a':'c'] incluye 'c'). Acepta booleano: df.loc[df['x'] > 0, 'col'].

.iloc[fila, col]

: Acceso por posición entera. Slicing es exclusivo del extremo (estilo Python). df.iloc[0:5] da 5 filas (índices 0..4).

.at[fila, col] / .iat[fila, col]

: Versiones para acceder/asignar un único valor. ~10× más rápidos que .loc/.iloc cuando estás en loops. Mismo patrón label vs posición.

Indexer chaining (df[cond][col] = x)

: Encadenar dos [] operaciones de acceso. Crea ambigüedad: ¿es vista o copia? Origen del SettingWithCopyWarning. Evítalo siempre.

SettingWithCopyWarning

: Aviso de pandas: "estoy haciendo algo ambiguo, puede que tu asignación se pierda". Causado por chaining o asignación sobre subset no-explícito. Solución universal: .loc[cond, col] = x en una sola operación.

Dataset / recursos

Palmer Penguins desde URL (mismo de clase 022) o el sintético si no hay internet.

Ejercicios

1. Acceso simple. Carga penguins. Obtén la columna species con los 3 métodos: df.species, df['species'], df.loc[:, 'species'].

2. loc inclusivo vs iloc exclusivo. Con index 0..N por default, compara df.loc[0:5] vs df.iloc[0:5]. ¿Cuántas filas devuelve cada uno?

3. Filtro + columnas seleccionadas. Pingüinos Adelie machos con bill_length > 40: df.loc[(df.species=='Adelie') & (df.sex=='male') & (df.bill_length_mm > 40), ['species', 'island', 'bill_length_mm']].

4. Asignación segura. Crea una columna is_big que sea True si body_mass_g > 4500, usando .loc.

5. Provoca y arregla SettingWithCopyWarning. Slicea con df[df.x > 0] y modifica → ve warning. Hazlo con .loc → sin warning.

Homework verificable

Notebook que: (a) muestra los 3 métodos de acceso a columna; (b) compara loc vs iloc en slicing con tabla; (c) filtra Adelie machos con bill_length>40 mostrando 3 columnas; (d) reproduce y arregla SettingWithCopyWarning con explicación.

Criterio de aceptación: Los filtros producen el subset correcto; la versión con .loc no emite warning.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
SettingWithCopyWarning aparece y no sé por	Estás asignando sobre un resultado de slic
df.loc[0:5] da 6 filas no 5	loc es inclusive end. Para 5 filas: df.ilo
KeyError con .loc[5] cuando hice set_index	El index ya no es 0..N — es la columna id.
Asignación a vista no modifica el original	Pandas 3+ va a ser estricto: la vista no s
df.col1 = x no actualiza la columna	Como atributo, asignar no agrega columna n

Preguntas frecuentes

¿loc o iloc?

loc cuando el index es semántico (nombres, fechas, ids). iloc cuando solo importa la posición (top-K por orden, primeros 10, último). Mezclarlos en el mismo código causa confusión.

¿Por qué loc slicing es inclusivo?

Decisión histórica: para labels (strings, fechas), incluir el end es lo intuitivo ('enero':'marzo' debe incluir marzo). Para enteros default queda raro — usa iloc ahí.

¿at vale la pena vs loc para single value?

Solo en hot loops (>10k iteraciones). Para uso interactivo, loc es lo suficientemente rápido y más legible.

¿Cómo selecciono múltiples columnas?

Lista entre brackets: df[['a', 'b', 'c']] (devuelve DataFrame). Notar el [[]]. Un solo [] con string devuelve Series.

¿.loc[mask, cols] o .query() + [cols]?

Ambos válidos. .loc[mask, cols] para máscaras computadas; .query() para filtros declarativos largos. Misma velocidad para datasets <100k.

Referencias

- VanderPlas, cap. 3 § 3.3.
- pandas Indexing user guide
- SettingWithCopyWarning explained

Siguiente clase

Clase 024 — Pandas: operaciones y alineación

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
import numpy as np
import pandas as pd
rng = np.random.default_rng(42)

# DataFrame de demo
df = pd.DataFrame({
    'nombre': ['Ana', 'Bob', 'Cris', 'Dan', 'Eli'],
    'edad'  : [30, 25, 28, 35, 22],
    'nota'  : [7.5, 6.0, 8.2, 5.8, 9.1],
}, index=['a', 'b', 'c', 'd', 'e'])
print(df)
```

Archivos complementarios

- notebook.ipynb