Clase 018 — NumPy: boolean masks y fancy indexing

Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 2, §§ 2.6–2.7 Comparisons, Masks, and Boolean Logic + Fancy Indexing. · Duración estimada: 75 min.

Clase 018 — NumPy: boolean masks y fancy indexing

> *Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 2, §§ 2.6–2.7 Comparisons, Masks, and Boolean Logic + Fancy Indexing. · Duración estimada: 75 min.*

Objetivo

Que el alumno seleccione, filtre y modifique sub-arrays de tres formas: slicing (visto), máscaras booleanas (arr[arr > 0]) y fancy indexing (arr[[0, 3, 5]]). Saber cuál devuelve vista vs copia y cuándo cada uno es la herramienta correcta.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Filtrar elementos con máscaras booleanas: arr[arr > 0], arr[(a > 0) & (a < 10)].
2. Combinar máscaras con &, |, ~ — NO con and/or (no vectorizan).
3. Seleccionar por índices con fancy indexing: arr[[0, 3, 5]] o arr[idx_array].
4. Modificar in-place con máscara: arr[arr < 0] = 0 (clipping).
5. Diferenciar vista vs copia: slicing es vista; fancy indexing y máscara son copia.

Temas

#	Tema	Por qué importa	
1	Comparaciones elementwise → a	arr > 0 no devuelve un bool, devu	
2	np.count_nonzero, np.sum sobre	Cuenta cuántos True.	
3	Combinar máscaras con &, `	, ~`	Operadores bitwise — no and/or.
4	Fancy indexing con array de índic	Selección no contigua.	
5	Vista vs copia	Slicing = vista; mask/fancy = copi	
6	np.where(cond) (sin alternativas)	Devuelve índices donde se cumpl	

Definiciones y características

Boolean mask

: Array de bools del mismo shape que el original. arr[mask] extrae solo los elementos donde mask es True. Devuelve un nuevo array (copia, no vista).

Fancy indexing

: Indexar con array de enteros (índices arbitrarios, posiblemente no contiguos). arr[[0, 3, 5]] selecciona esas 3 posiciones. Devuelve copia.

Vista (view) vs copia (copy)

: Slicing (arr[:5]) → vista (mismo storage, mutarla muta el original). Mask / fancy → copia (storage independiente). Fuente del 70% de los bugs sutiles.

Operadores bitwise vs lógicos

: Para combinar masks: &, |, ~ (bitwise, vectorizados, elementwise). NO uses and, or, not — son escalares Python y dan ValueError: truth value of an array is ambiguous.

np.where(cond) 1-arg

: Sin alternativas, devuelve tupla de arrays de índices donde se cumple la condición. Diferente a np.where(cond, a, b) (ternario).

Dataset / recursos

Sintético: array de precipitación diaria (365 valores). Sin descarga.

Ejercicios

1. Cuenta días lluviosos. Dado array de 365 días con precipitación (mm), cuenta cuántos tuvieron >5mm.

2. Estadísticos por máscara. Calcula precipitación media solo en días lluviosos (>0mm).

3. AND/OR combinados. Días entre 1 y 10 mm. Días <1 o >50 mm.

4. Clipping. Reemplaza valores negativos por 0 in-place (arr[arr < 0] = 0).

5. Vista vs copia. Demuestra con un experimento que arr[:5] modifica el original pero arr[arr > 0] no.

Homework verificable

Notebook con array sintético de 365 días de precipitación generado con seed. Calcula: (a) días lluviosos y su media, (b) días extremos (>50mm), (c) demo de vista vs copia, (d) clipping in-place, (e) índices del top 10 días más lluviosos con argsort.

Criterio de aceptación: Resultados reproducibles. Demo vista/copia muestra comportamiento opuesto.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar	
ValueError: The truth value of an array wi	Usaste and/or/not con arrays. Fix: &/`	/~ con paréntesis: (a > 0) & (a < 10)` (lo
Modifico arr[mask] y el original no cambia	Mask devuelve copia. Fix: para modificar i	
arr[idx1, idx2] con fancy indexing me da a	Si idx1 e idx2 son arrays del mismo length	
Slice modifica el original sin querer	subset = arr[:10]; subset[0] = 99 modifica	
Mask con shape distinto al array	Lanza IndexError: boolean index did not ma	

Preguntas frecuentes

¿Mask o fancy indexing?

Mask cuando la selección viene de una condición sobre los valores (arr[arr > 0]). Fancy cuando ya tienes los índices específicos (de un argsort, por ejemplo, o de business logic).

¿Cómo modifico múltiples elementos con mask?

arr[arr < 0] = 0 (clipping). Funciona para asignar escalar a todos los True, o array del mismo tamaño que la

cantidad de Trues: arr[arr < 0] = -arr[arr < 0] (abs solo donde negativo).

¿arr[idx] con idx como booleano o entero?

NumPy distingue: bool array del mismo shape → mask; int array → fancy indexing. Lista Python de bools también funciona, pero ojo con [0, 1, 0, 1] que puede interpretarse como ints (índices 0 y 1) o bools — usa np.array(...) explícito si hay duda.

¿np.where(cond) o np.nonzero(cond)?

Idénticos cuando where se llama con un solo argumento. nonzero es más explícito del intent ("dónde NO es 0/False").

¿Cómo combino mask con np.where ternario?

np.where(cond, valor_si_true, valor_si_false) — ternario vectorizado. La diferencia con arr[cond] = X: where construye array nuevo; mask + asignación modifica in-place.

Referencias

- VanderPlas, cap. 2 §§ 2.6, 2.7.
- NumPy indexing user guide

Siguiente clase

Clase 019 — NumPy: ordenamiento y búsqueda

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

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

Archivos complementarios

- notebook.ipynb