Clase 014 — NumPy: tipos, creación, atributos

Parte: 0 — Prerrequisitos · Fuente: VanderPlas, Python Data Science Handbook, cap. 2 — Introduction to NumPy, §§ 2.1–2.2. · Duración estimada: 75 min.

Clase 014 — NumPy: tipos, creación, atributos

> *Parte: 0 — Prerrequisitos · Fuente: VanderPlas, Python Data Science Handbook, cap. 2 — Introduction to NumPy, §§ 2.1–2.2. · Duración estimada: 75 min.*

Objetivo

Que el alumno entienda el modelo mental de un ndarray — bloque contiguo de memoria con shape, dtype y strides — y sepa crear arrays de las 6 formas más útiles (array, zeros, arange, linspace, random, desde lista). Sin este modelo, todo el rendimiento de NumPy parece magia.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Explicar por qué ndarray es 50–100× más rápido que list (memoria contigua + dtype fijo + sin overhead Python).
2. Crear arrays con np.array, np.zeros, np.ones, np.full, np.arange, np.linspace.
3. Inspeccionar un array con shape, dtype, ndim, size, nbytes, itemsize.
4. Cambiar dtype explícitamente con astype y entender promociones implícitas (int + float = float).
5. Generar arrays aleatorios reproducibles con np.random.default_rng(seed).

Temas

#	Tema	Por qué importa
1	ndarray: memoria contigua + dtype fijo	Lo que lo hace rápido.
2	Creación: array, zeros, arange, linspace	Las 6 formas más usadas.
3	dtype: int8/16/32/64, float32/64, bool	Memoria y precisión.
4	Atributos: shape, dtype, ndim, size, nbyte	Diagnóstico instantáneo.
5	astype y promoción de tipos	El bug clásico de overflow int8.
6	random moderno: default_rng(seed)	El API legacy np.random.seed está deprecat

Definiciones y características

ndarray

: Estructura de datos central de NumPy. Bloque contiguo de memoria con dtype fijo + metadata (shape, strides). 50–100× más rápido que list por evitar el overhead de PyObject por elemento y permitir vectorización SIMD.

dtype

: Tipo de elemento del array: int8/16/32/64, uint8 (imágenes RGB), float32/64, bool, complex64/128. Determina memoria por elemento (itemsize) y precisión/rango.

shape

: Tupla con tamaño de cada dimensión: (10,) 1D, (3, 4) matriz, (2, 3, 4) tensor 3D. len(shape) = ndim.

prod(shape) = size (total elementos).

strides

: Bytes que el array salta para moverse 1 paso en cada dimensión. Permite vistas eficientes sin copiar memoria (transpose, slicing). Detalle interno pero útil para entender por qué algunas operaciones son gratis.

Generator (random)

: API moderno para aleatoriedad: rng = np.random.default_rng(seed). Reemplaza al legacy np.random.seed() + funciones globales. Características: PCG64 (mejor algoritmo), múltiples generadores independientes, API consistente.

Promoción de tipo

: Cuando operas arrays de dtypes distintos, NumPy promueve al más amplio: int + float = float, int8 + int16 = int16. La regla evita pérdida silenciosa pero puede generar overflow en otros casos (clase: overflow int8).

Dataset / recursos

Sintético: arrays generados en el notebook (escalares, matrices, aleatorios reproducibles). Sin descarga.

Ejercicios

1. Memoria. Crea list(range(1_000_000)) y np.arange(1_000_000). Compara sys.getsizeof y arr.nbytes. Calcula el ratio.

2. Las 6 formas. Crea: vector 100 ceros, matriz 5×5 unos, vector 0..1 con 50 puntos equiespaciados, matriz 3×3 de 7s, vector de 100 aleatorios uniformes [0,1).

3. Bug de dtype. Crea np.array([100, 200, 50], dtype=np.int8) y suma 200 a cada elemento. Observa el resultado y explica.

4. Diagnóstico. Dado un array, escribe una función que imprima shape, dtype, ndim, size, nbytes y memoria humana (KB/MB).

5. Random reproducible. Genera 1000 normales N(0,1) con seed=42. Calcula media y std. Repite — debe dar exactamente lo mismo.

Homework verificable

Notebook que: (a) compara memoria list vs ndarray para N=1M con tabla; (b) crea las 6 formas y reporta dtype default de cada una; (c) reproduce el bug de overflow int8 con explicación; (d) función info(arr) con diagnóstico completo.

Criterio de aceptación: El ratio memoria list/ndarray es >5×. La función info reporta todos los atributos.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
OverflowError silencioso con int8/uint8	NumPy no lanza excepción — hace wrap-aroun
np.array([1, 2, 'x']) queda como dtype='<U	NumPy promueve TODO a string para ser homo
np.empty(3) con basura en vez de ceros	empty NO inicializa — más rápido que zeros

Reproduzco un experimento con seed y da re	Estás usando el API legacy (np.random.seed
np.arange(0.1, 1.0, 0.1) no incluye 1.0	Floats — el último step da 0.9999... por i

Preguntas frecuentes

¿np.array([1,2,3]) o np.asarray([1,2,3])?

asarray no copia si ya es ndarray (más eficiente); array siempre copia por default. Usa asarray cuando aceptas cualquier 'array-like' y no necesitas garantizar copia.

¿float32 o float64?

Default es float64 — máxima precisión. float32 cuando trabajas con redes neuronales en GPU (la mitad de memoria, suficiente para gradientes), imágenes, o necesitas duplicar la velocidad de IO.

¿Cuándo int32 y cuándo int64?

Default depende del OS (int64 Unix/macOS, int32 Windows pre-numpy 2.0). En 2026, NumPy 2+ usa int64 en todas las plataformas. Solo bajes a int32 si memoria es crítica y sabes que tus valores caben.

¿np.random.seed(42) ya no se usa?

Funciona pero está deprecated en favor del nuevo API. Razones: estado global (peligroso), Mersenne Twister (lento), no permite múltiples streams independientes. Usa default_rng(seed).

¿Por qué arr.nbytes no coincide con sys.getsizeof(arr)?

nbytes cuenta solo los datos (size * itemsize). getsizeof cuenta también el header del objeto ndarray (~100 bytes). Para arrays grandes la diferencia es despreciable.

Referencias

- VanderPlas, cap. 2, §§ 2.1–2.2 Understanding Data Types + The Basics of NumPy Arrays.
- NumPy user guide — Array creation
- NumPy dtypes
- Generator random API

Siguiente clase

Clase 015 — NumPy: ufuncs y vectorización

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
import sys
import numpy as np
print('numpy:', np.__version__)
```

Archivos complementarios

- notebook.ipynb