Introduzione a NumPy

NumPy è l'abbreviazione di **Numerical Python**: un'estensione del linguaggio pensata per il calcolo **algebrico** e **matriciale**.

Consente di lavorare con **vettori** e **matrici** in maniera più efficiente e veloce rispetto alle liste Python standard.

Moduli Essenziali per l'Analisi Dati

NumPy

Calcoli numerici e operazioni su array multidimensionali

Pandas

Caricamento e manipolazione dei dati strutturati

Matplotlib

Visualizzazione e creazione di grafici

Seaborn

Visualizzazioni statistiche avanzate

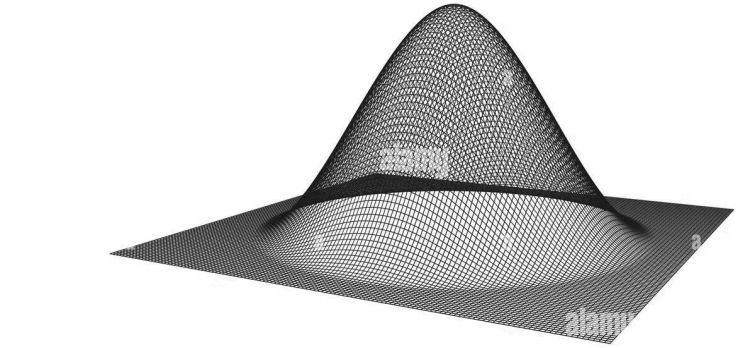
Gli Array NumPy (ndarray)

NumPy fornisce un nuovo tipo di dato: l'array **N-dimensionale (ndarray)**.

È un oggetto array multidimensionale caratterizzato da proprietà specifiche che lo rendono superiore alle liste Python standard.

Vantaggi degli ndarray

- Maggiore efficienza computazionale
- Operazioni vettoriali ottimizzate
- Supporto per calcoli matematici complessi
- Integrazione con librerie scientifiche



Proprietà Fondamentali degli Array

Dimensioni

Gli **ndarray** possono avere da una a un numero arbitrario di dimensioni, definite come "**assi**". Possono essere unidimensionali (vettori), bidimensionali (matrici), tridimensionali (cubi), e così via.

Tipo di Dato

Tutti gli elementi devono avere lo stesso tipo (float, int, bool, string), a differenza delle liste Python che non sono omogenee.

Forma (Shape)

Indica le dimensioni dell'array, cioè il numero di elementi per ogni asse. Ad esempio, (3, 4) significa 3 righe e 4 colonne.

Indicizzazione

Supportano **indicizzazioni avanzate** oltre a quelle standard Python, permettendo selezioni complesse di elementi.

Terminologia Essenziale

Termine	Definizione	Esempio	Valore
Size	Numero totale di elementi	Array 3x5	15
Rank	Numero di assi/dimensioni	Matrice	2
Shape	Dimensioni per ogni asse	Array 4x3x5	(4,3,5)

Creazione di Array

Il modo più semplice per creare un **ndarray** è convertire una lista Python.

NumPy offre anche funzioni specializzate per generare array con caratteristiche specifiche.

01

Array da Lista

np.array([1, 2, 3, 4, 5, 6]) crea un vettore con 6 elementi

02

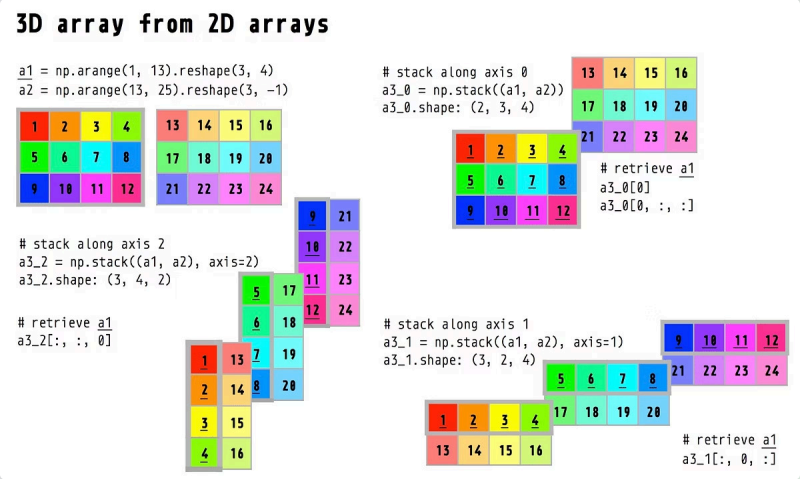
Array 2D

np.array([[1, 2, 3], [4, 5, 6]]) crea una matrice 2x3

03

Funzioni Specializzate

arange(), linspace(), zeros(), ones() per array con pattern specifici



Funzioni per Creare Array

np.arange()

Crea sequenze numeriche con incrementi specificati

Esempio: np.arange(2, 9, 2) → [2, 4, 6, 8]

np.linspace()

Genera punti equidistanti tra due estremi

Esempio: np.linspace(0, 10, 5) → [0, 2.5, 5, 7.5, 10]

np.zeros() / np.ones()

Crea array riempiti con zeri o uni

Esempio: np.zeros(5) → [0, 0, 0, 0, 0]

Indicizzazione e Accesso agli Elementi

L'indicizzazione degli array NumPy è **potente e flessibile,** permettendo di accedere a singoli elementi, righe, colonne o sottomatrici con sintassi intuitive.

Array 1D

a[0] → primo elemento

a[2] → terzo elemento

a[-1] → ultimo elemento

Array 2D

a[0, 2] → elemento riga 0, colonna 2

a[1] → intera seconda riga

a[:, 1] → intera seconda colonna

Operazioni Aritmetiche sugli Array

NumPy permette di eseguire operazioni algebriche elemento per elemento su interi array, automatizzando calcoli che normalmente richiederebbero cicli espliciti.

1

Dati Individuali

Altezza: 1.62m

Peso: 55.4kg

2

Array NumPy

m = [1.62, 1.75, 1.55, 1.74]

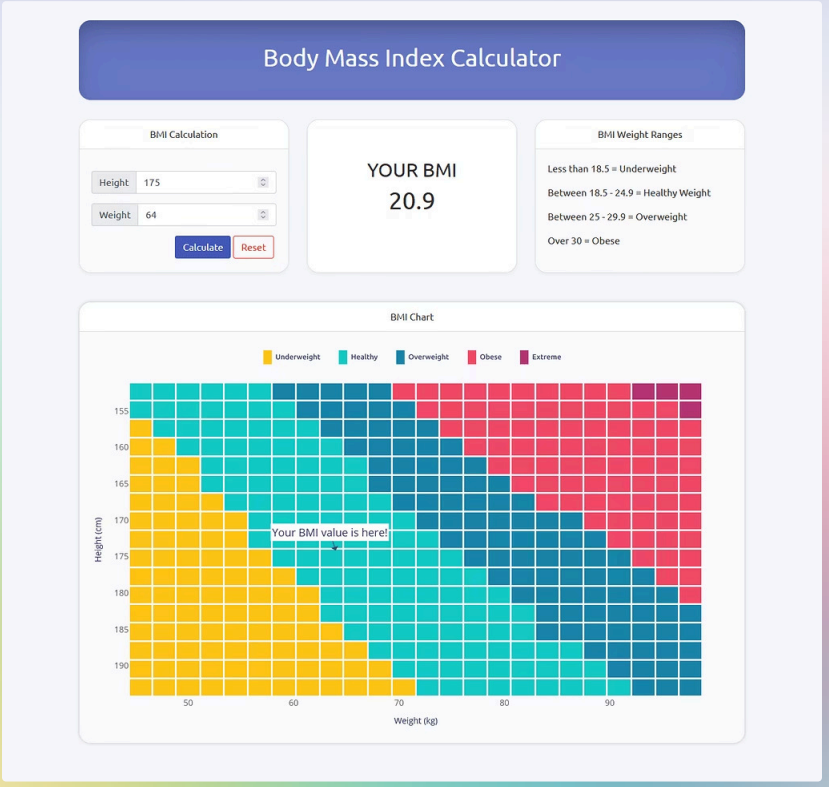
kg = [55.4, 73.6, 57.1, 59.5]

3

Calcolo BMI

bmi = kg / m**2

Risultato per tutti gli elementi



Broadcasting

Il **broadcasting** è un meccanismo potente che consente a NumPy di eseguire operazioni tra array di diverse dimensioni o tra un array e uno scalare, estendendo automaticamente le dimensioni quando necessario.

Estensione Automatica

NumPy allinea automaticamente le dimensioni degli array per rendere possibili le operazioni

Array-Scalare

Operazioni tra un array e un singolo numero vengono applicate a tutti gli elementi

Codice Compatto

Rende il codice più leggibile ed evita cicli espliciti per allineare le dimensioni

Funzioni Predefinite per Array

Statistiche

mean(), std(), min(), max() per analisi statistiche rapide

Aggregazione

sum(), prod() per operazioni di aggregazione sui dati

Conversione

astype() per convertire il tipo di dati degli elementi

Proprietà

shape, size, ndim, dtype per informazioni sull'array

Implementazione di Formule Matematiche

NumPy rende semplice l'implementazione di formule matematiche complesse. Esempio: calcolo della deviazione standard.

$$s = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}}$$

01

Calcola la Media

np.mean(x) produce uno scalare rappresentante la media

02

Sottrai la Media

(x - np.mean(x)) usando il broadcasting

03

Eleva al Quadrato

(x - np.mean(x))**2 per ogni elemento

04

Somma e Radice

np.sqrt(np.sum(...)/np.size(x)) per il risultato finale

Slicing: Selezione di Porzioni

Lo slicing permette di selezionare porzioni di array multidimensionali usando la sintassi [start:stop:step]. È uno strumento potente per l'analisi e manipolazione dei dati.

Sintassi Base

arr[start:stop:step] per selezioni lineari

arr[:, 1] per colonne specifiche

arr[0, :] per righe specifiche

Esempi Pratici

a[:2, 1:3] → prime 2 righe, colonne 1-2

a[::2, ::2] → ogni seconda riga e colonna

Attenzione: Viste vs Copie

Importante: Uno slice di un array NumPy è una vista degli stessi dati. Modificare lo slice modifica l'array originale!

Vista (View)

b = a[:2, 1:3] crea una vista. Modifiche a b influenzano a

Copia Indipendente

c = a.copy() crea una copia indipendente. Modifiche a c non influenzano a

Esercizi Pratici - Parte 1

1 **Vettore con Elemento Specifico**

Creare un vettore nullo di 10 elementi con il quinto valore uguale a 1

2 **Sequenza Numerica**

Creare un vettore con valori compresi tra 10 e 49

3 **Inversione Vettore**

Invertire un vettore (primo elemento diventa ultimo)

Esercizi Pratici - Parte 2

Elementi Non Zero

Trovare gli indici degli elementi non zero in [1, 2, 0, 0, 4, 0]

Soluzione: np.nonzero()

Array Casuali

Creare array 10x10 con valori casuali e trovare min/max

Soluzione: np.random.random()

Valore Medio

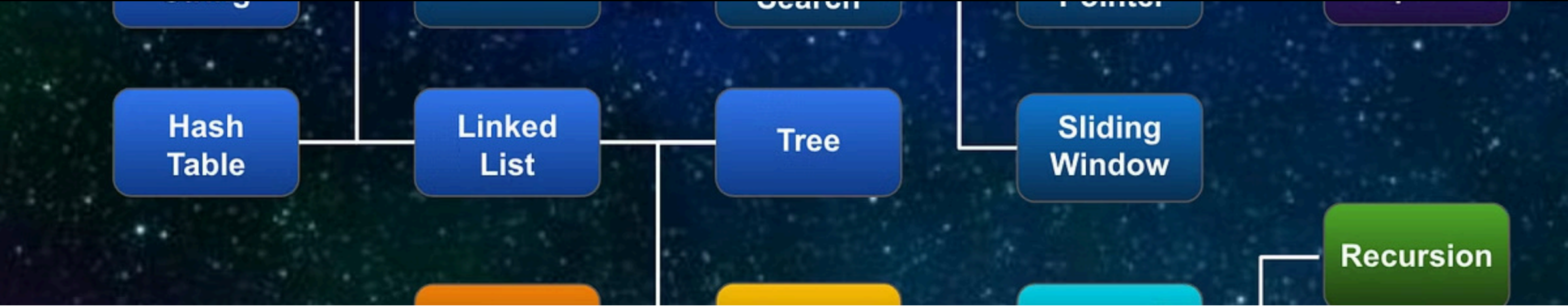
Creare vettore casuale di 30 elementi e calcolare la media

Soluzione: array.mean()

Operazioni Condizionali

NumPy permette operazioni sofisticate basate su condizioni, inclusa la negazione selettiva di elementi e la sostituzione di valori basata su criteri specifici.

1	2	3
Condizione	**Operazione**	**Risultato**
Z[(3 < Z) & (Z < 8)]	*= -1	[0,1,2,3,-4,-5,-6,-7,8,9,10]
Seleziona elementi tra 3 e 8	Moltiplica per -1 (nega)	Solo elementi 4,5,6,7 negati



Ricerca e Sostituzione

Trovare il Massimo

Z.argmax() restituisce l'indice dell'elemento massimo

Z[Z.argmax()] = 0 sostituisce il massimo con 0

Valore Più Vicino

np.abs(Z - v).argmin() trova l'indice dell'elemento più vicino al valore v

Utile per ricerche approssimate in dataset

Conversioni di Tipo

Float → Integer

a.astype("int") converte numeri decimali in interi, troncando la parte decimale

1

Controllo Tipo

array.dtype mostra il tipo corrente degli elementi nell'array

3

2

Binario → Booleano

a.astype("bool") converte 0 in False e qualsiasi altro valore in True

Operazioni su Array Multipli

NumPy eccelle nelle operazioni che coinvolgono più array, permettendo confronti, unioni e manipolazioni complesse con sintassi semplice.

Unione Orizzontale

np.hstack((a1, a2)) unisce array orizzontalmente, creando matrici più larghe mantenendo il numero di righe

Confronto Elementi

np.where(a == b) trova gli indici dove gli elementi di due array corrispondono esattamente

Moltiplicazione

np.multiply(a, b) moltiplica elemento per elemento due array delle stesse dimensioni

Generazione e Ripetizione

Ripetizione Pattern

np.tile(array, n) ripete un array n volte, utile per creare pattern ripetitivi o dataset di test

Numeri Casuali

np.random.randint(0, 10, size=(5,5)) genera array di interi casuali con dimensioni specificate

Verifica e Confronto

Verifica Elementi

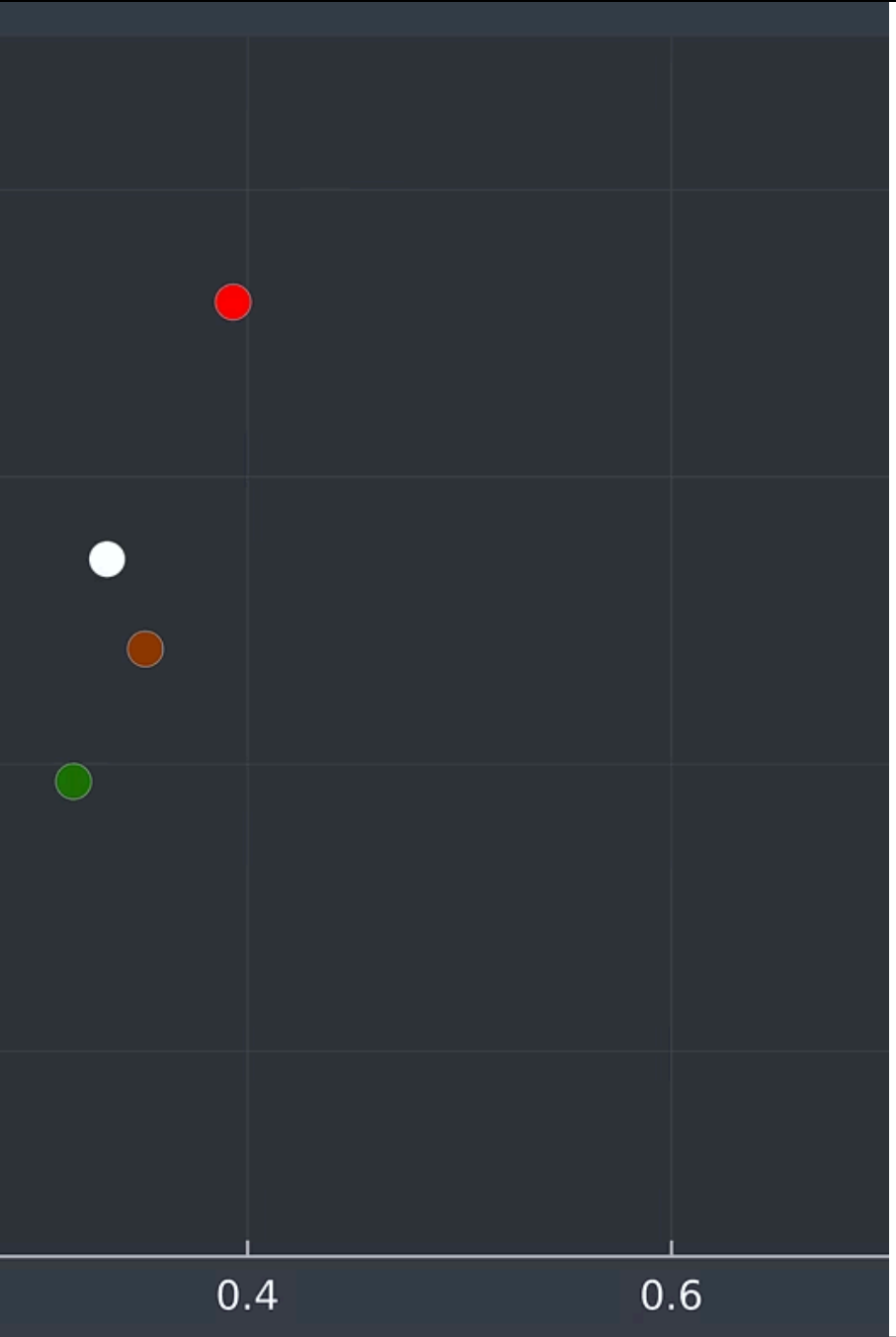
np.any(x) verifica se almeno un elemento è diverso da zero

np.all(x) verifica se tutti gli elementi sono diversi da zero

Confronto con Tolleranza

np.equal(x, y) confronto esatto elemento per elemento

np.allclose(x, y) confronto con tolleranza per errori di precisione



Gestione Dati Mancanti

NumPy fornisce strumenti specifici per identificare e gestire valori mancanti (NaN) nei dataset, essenziale per l'analisi di dati reali.

01

Identificazione

np.isnan(array) crea una maschera booleana che identifica i valori NaN

02

Conteggio

np.sum(np.isnan(array)) conta il numero totale di valori mancanti

03

Filtraggio

array[~np.isnan(array)] estrae solo i valori non-NaN per l'analisi

Filtraggio e Sostituzione Avanzata

1

Estrazione Condizionale

nums[nums < n] estrae tutti gli elementi minori di un valore specificato, creando un nuovo array filtrato

2

Sostituzione Condizionale

np.where(condizione, valore_se_vero, valore_se_falso) sostituisce elementi basandosi su condizioni complesse

3

Operazioni Multiple

Possibilità di combinare più condizioni con operatori logici (&, |, ~) per filtrazioni sofisticate

Riepilogo e Prossimi Passi

NumPy è la base fondamentale per il calcolo scientifico in Python. Abbiamo esplorato array multidimensionali, operazioni vettoriali, broadcasting e manipolazione avanzata dei dati.

25+

Funzioni Principali

Dalla creazione di array alle operazioni statistiche avanzate

100%

Efficienza

Prestazioni superiori rispetto alle liste Python standard

∞

Possibilità

Base per Pandas, Matplotlib, SciPy e tutto l'ecosistema scientifico Python