

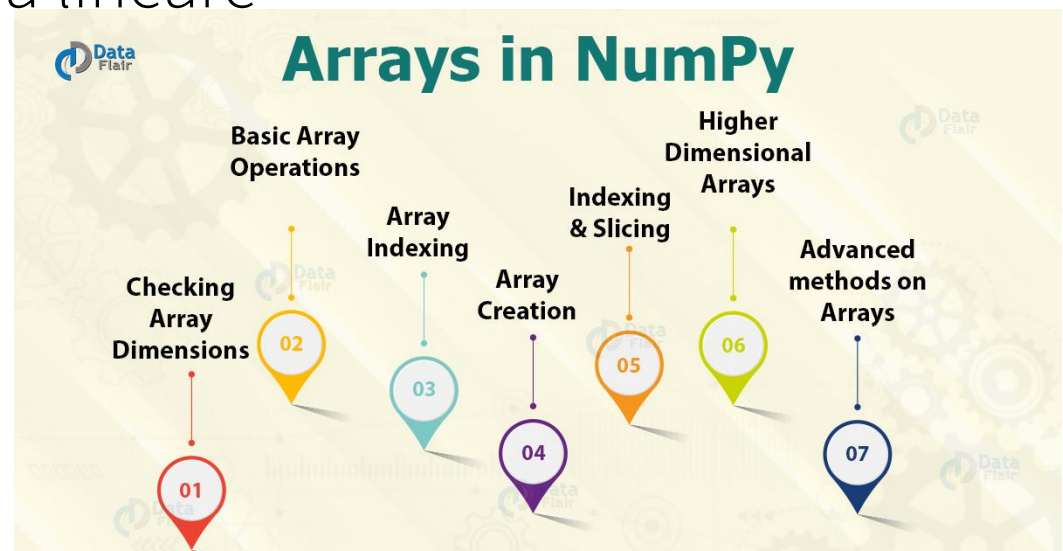
NUMPY

Introduzione

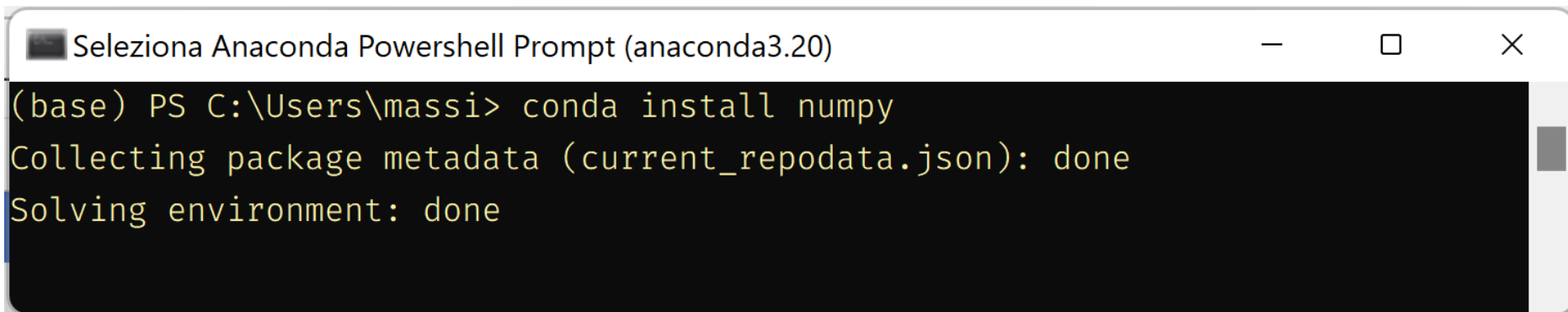
- **numpy** è una libreria Python che aggiunge un importante supporto a *grandi matrici e array multidimensionali*
- Introduce una vasta collezione di funzioni matematiche di alto livello per poter operare efficientemente su queste strutture dati
- Molto utile nell'ambito della *datascience*
- Alla base di molte altre librerie che useremo in questo corso

Caratteristiche

- Alte prestazioni
- Codice base C/C++
- Gestione dati multidimensionali
- Funzioni broadcasting
- Funzione base di algebra lineare



Installazione



The image shows a screenshot of a Windows PowerShell window titled "Seleziona Anaconda Powershell Prompt (anaconda3.20)". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The command prompt shows the following text:

```
(base) PS C:\Users\massi> conda install numpy
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

Importazione

```
[1]: import numpy as np
```

```
[2]: np.version.full_version
```

```
[2]: '1.20.3'
```

```
[3]: np.version.version
```

```
[3]: '1.20.3'
```

Array in numpy

- `numpy` ha una speciale classe denominata `numpy.ndarray` che *assomiglia* alla classe `list` di `python`
- Ma:
 - ammette solo tipi omogenei (ma strutturati)
 - è orientata al calcolo numerico vettoriale
- Per creare un oggetto di classe `numpy.ndarray` dobbiamo usare il relativo costruttore:

```
numpy.array(object,  
            dtype=None,  
            *,  
            copy=True,  
            order='K',  
            subok=False,  
            ndmin=0,  
            like=None)
```

Prima

```
file_wh = open("weight-height.csv")  # apro il file
sex = [] # inizializzo 4 liste
weight = []
height = []
bmi = []
lb_to_kg = 0.453592 # costanti di conversione
in_to_mt = 0.0254

file_wh.readline() # skip prima riga (header)
for line in file_wh:
    fields = line.split(",") # separo la riga
    sex.append(fields[0].strip('"')) # aggiungo il record
    height.append(float(fields[1]) * in_to_mt)
    weight.append(float(fields[2]) * lb_to_kg)
    bmi.append(weight[-1] / height[-1]**2) # calcolo il bmi
file_wh.close()
```

Dopo

```
import numpy as np

dataset = np.genfromtxt('weight-height.csv',
                        dtype=('U10', 'f', 'f'),
                        skip_header=1,
                        delimiter=",",
                        converters={0: lambda s:s[1:-1]},
                        names=["Sesso", "Altezza", "Peso"])

bmi = dataset["Peso"] * lb_to_kg / (dataset["Altezza"] * in_to_mt)**2
```


genfromtxt

numpy.genfromtxt

```
numpy.genfromtxt(fname, dtype=<class 'float'>, comments='#', delimiter=None,  
skip_header=0, skip_footer=0, converters=None, missing_values=None, filling_values=None,  
usecols=None, names=None, excludelist=None, deletechars=" !#$%&'()*+,-.  
:/:;<=>?@[\\]^_{|}~", replace_space='_', autostrip=False, case_sensitive=True,  
defaultfmt='%i', unpack=None, usemask=False, loose=True, invalid_raise=True,  
max_rows=None, encoding='bytes', *, ndmin=0, like=None) #
```

[\[source\]](#)

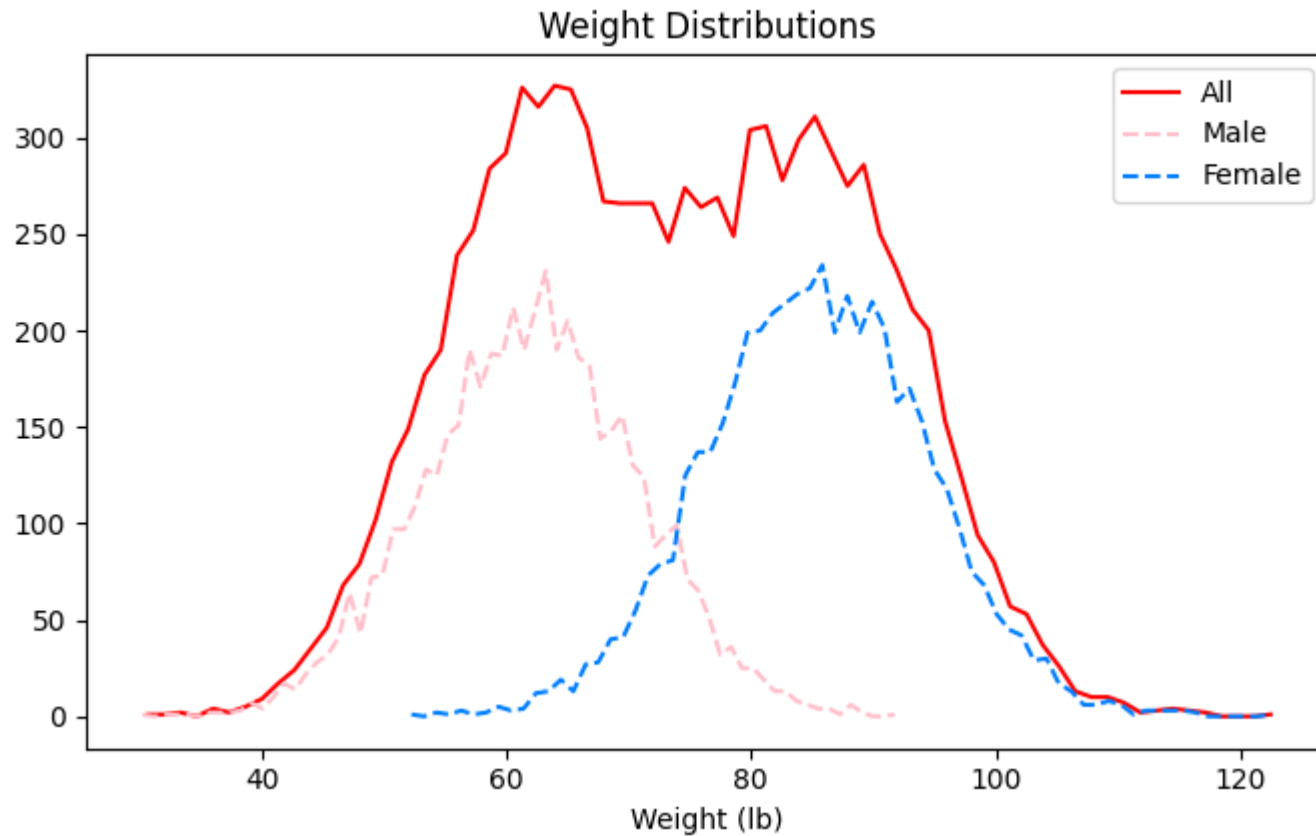
Interazione con *matplotlib*

- La classe `numpy.ndarray` è pienamente supportata da *matplotlib*

```
import matplotlib.pyplot as plt

(values, intervals) = np.histogram(np_weight, bins=70)
(fvalues, fintervals) = np.histogram(np_weight[5000:], bins=70)
(mvalues, mintervals) = np.histogram(np_weight[:5000], bins=70)
plt.figure(figsize=(8,4.5))
plt.plot(intervals[1:], values, 'r')
plt.plot(fintervals[1:], fvalues, '--', c='#FFC0CB')
plt.plot(mintervals[1:], mvalues, '--', c='#007FFF')
plt.legend(["All", "Male", "Female"])
plt.title("Weight Distributions")
plt.xlabel("Weight (lb)")
plt.show()
```

Integrazione con matplotlib - 2



Overload operatori

- Gli operatori aritmetici e booleani hanno per gli array *numpy* un comportamento diverso dalle liste:
 - La somma di due ndarray crea un ndarray che è la concatenazione dei due addendi
- Per *numpy* la somma, la sottrazione, la divisione e tutte le altre operazioni binarie aritmetiche hanno il seguente comportamento:

$$l^{op} = l^1 \oplus l^2 \mid l_i^{op} = l_i^1 \oplus l_i^2 \quad (\oplus = \text{operatore}) \quad \forall i$$

- La stessa cosa vale per gli operatori booleani restituendo, in questo caso un array di booleani
- Ovviamente gli array devono essere della stessa lunghezza.

Esempio

```
l1 = np.array([7, 2, 6, 4, 10])
l2 = np.array([2, 4, 5, 8, 4])
print(l1 / l2)    # [3.5 0.5 1.2 0.5 2.5]
print(l1 * l2)    # [14  8 30 32 40]
print(l1 % l2)    # [1 2 1 4 2]

print(l1 > l2)    # [ True False True  False True]
print(l1 < l2)    # [False  True False True  False]
```

Selezione (Slicing)

- Le liste Python ammettono alcune forme di selezione (*slicing*) legate alla posizione degli elementi
 - Possiamo ad esempio usando l'operatore parentesi quadra (`[]`) selezionare un sottoinsieme lineare
 - `[a:b:c]` dove `a` è l'elemento iniziale, `b` il finale e `c` lo step
 - `a`, `b`, `c` sono opzionali
- La stessa cosa fa anche l'array di *numpy*
- Ma fa molto di più!
- Ammette come indice una lista

Array Indexing



Esempio

```
# standard slicing  
print(bmi[0:10000:1000])
```

```
# overload operatore > e slicing:  
oversize = bmi > 32 # creo array  
print(bmi[oversize])  
# in breve  
print(bmi[bmi > 32])
```

Metodi

- La classe *ndarray* implementa vari metodi che possiamo dividere in categorie:
 - *Conversione di array*
 - *Manipolazioni di forma*
 - *Selezione e manipolazione di elementi*
 - *Calcolo*
 - *Operatori matematici*
 - *overloaded*
 - *functional*
 - *Costruttori*
 - *Funzioni statistiche, trigonometriche, esponenziali, ...*

Funzioni matematiche

- Come esempio di funzioni vediamo alcune implementazioni di funzioni matematiche che agiscono non su numeri ma su array (Matlab style!)

Trigonometric functions

<code>sin(x, /[, out, where, casting, order, ...])</code>	Trigonometric sine, element-wise.
<code>cos(x, /[, out, where, casting, order, ...])</code>	Cosine element-wise.
<code>tan(x, /[, out, where, casting, order, ...])</code>	Compute tangent element-wise.
<code>arcsin(x, /[, out, where, casting, order, ...])</code>	Inverse sine, element-wise.
<code>arccos(x, /[, out, where, casting, order, ...])</code>	Trigonometric inverse cosine, element-wise.
<code>arctan(x, /[, out, where, casting, order, ...])</code>	Trigonometric inverse tangent, element-wise.

Trattamento dati

Sums, products, differences

prod (a[, axis, dtype, out, keepdims, ...])	Return the product of array elements over a given axis.
sum (a[, axis, dtype, out, keepdims, ...])	Sum of array elements over a given axis.
nanprod (a[, axis, dtype, out, keepdims, ...])	Return the product of array elements over a given axis treating Not a Numbers (NaNs) as ones.
nansum (a[, axis, dtype, out, keepdims, ...])	Return the sum of array elements over a given axis treating Not a Numbers (NaNs) as zero.
cumprod (a[, axis, dtype, out])	Return the cumulative product of elements along a given axis.
cumsum (a[, axis, dtype, out])	Return the cumulative sum of the elements along a given axis.
nancumprod (a[, axis, dtype, out])	Return the cumulative product of array elements over a given axis treating Not a Numbers (NaNs) as one.
nancumsum (a[, axis, dtype, out])	Return the cumulative sum of array elements over a given axis treating Not a Numbers (NaNs) as zero.

Trattamento dati

Text files

<code>loadtxt(fname[, dtype, comments, delimiter, ...])</code>	Load data from a text file.
<code>savetxt(fname, X[, fmt, delimiter, newline, ...])</code>	Save an array to a text file.
<code>genfromtxt(fname[, dtype, comments, ...])</code>	Load data from a text file, with missing values handled as specified.
<code>fromregex(file, regexp, dtype[, encoding])</code>	Construct an array from a text file, using regular expression parsing.
<code>fromstring(string[, dtype, count, like])</code>	A new 1-D array initialized from text data in a string.
<code>ndarray.tofile(fid[, sep, format])</code>	Write array to a file as text or binary (default).
<code>ndarray.tolist()</code>	Return the array as an <code>a.ndim</code> -levels deep nested list of Python scalars.