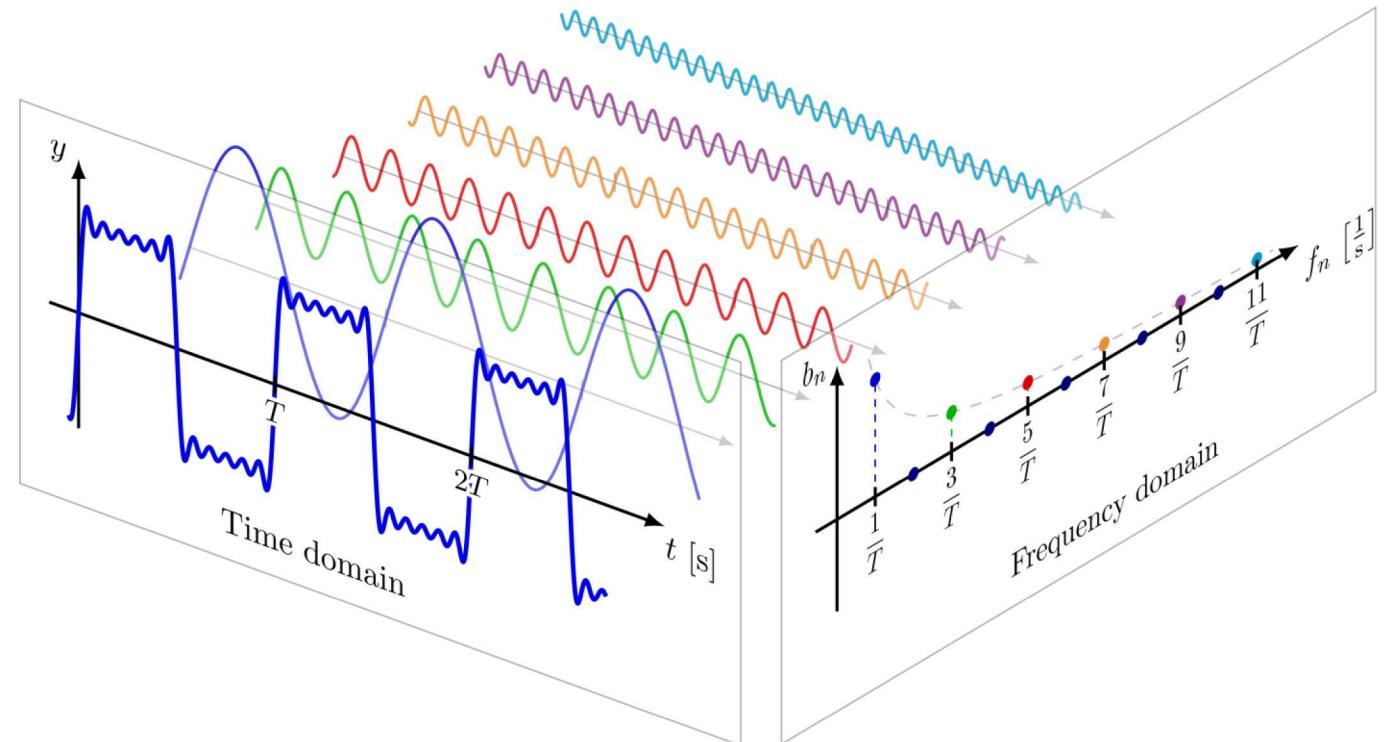


CURS 3 PD

Introducere în limbajul Python

Librăria NumPy



Introducere

Generarea matricelor

Operații matematice

Alte funcționalități

1. Introducere

2. Generarea matricelor

3. Operații matematice

4. Alte funcționalități

Generarea matricelor

Operații matematische

Introducere

Alte funcționalități

Definiție

NumPy (prescurtare de la **Numerical Python**) este o librărie Python care oferă posibilitatea de a defini și utiliza matrice multi-dimensionale, oferind de asemenea și o serie de funcții de nivel înalt pentru a lucra cu acestea (funcții pentru manipularea matricelor, funcții pentru generarea matricelor, funcții de algebră liniară, etc)

- Librăria **NumPy** necesită instalare (așa cum s-a văzut la primul curs).
- Pentru a o putea utiliza , va trebui importată scriind la începutul programului instrucțiunea `import` urmată de numele librăriei (numpy).
- Funcțiile și constantele definite în această librărie pot fi accesate scriind numele librăriei (numpy), simbolul punct și apoi numele funcției sau constantei dorite.
- Pentru a economisi timp, putem să utilizăm un alias mai scurt pentru numele librăriei, completând instrucțiunea import cu un alt cuvânt cheie, `as`, urmat de alias-ul dorit.
- În mod uzual, pentru librăria **NumPy** se folosește alias-ul `np`.

Importare cu alias

```
import numpy as np
```

- O matrice **NumPy** n-dimensională este o grilă de valori de același tip
- Ea se poate inițializa utilizând funcția `np.array` căreia îi dăm ca parametru liste imbricate (ex.: o matrice bidimensională se poate inițializa cu o listă de liste).
- Accesarea elementelor se face utilizând tot paranteze drepte între care se pune indexul / indecesii elementului, numărarea indecesilor începând de la 0

EXEMPLU	
main.py	rezultat
<pre>import numpy as np # matrice unidimensională X = np.array([3, 2, 1]) print(X) print(X[1]) # matrice bidimensională Y = np.array([[3, 2, 1], [6, 5, 4]]) print(Y) print(Y[1, 0])</pre>	[3 2 1] 2
	[[3 2 1] [6 5 4]] 6

- Un al doilea parametru care se poate da funcției `np.array` este tipul de date al elementelor.
- Dacă acesta lipsește, **NumPy** îl va stabili valoarea ca fiind tipul de date cu dimensiune minimă ce poate memora toate elementele date.
- Tipul de date al elementelor poate lua una din valorile:

Data Type	Descriere
?	Boolean
b	Signed byte
B	Unsigned byte
i	Signed integer
u	Unsigned integer
f	Floating point
c	Complex floating point
U	Unicode string

- Tipurile i, u, f și c pot fi urmate și de numărul de byți utilizati la stocarea elementelor (ex: `i8` este *unsigned integer* pe 8 byți – `int64`).
- Pentru o matrice **NumPy** deja existentă, tipul de date se poate obține accesând proprietatea `dtype`.
- Utilizând metoda `astype`, valorile unei matrice deja existentă se pot pune într-o altă matrice de alt tip

Exemplu
main.py
<pre>import numpy as np x = np.array([1, 2, 3]) print(f"{x} - {x.dtype}") x = np.array([1, 2, 3], 'f8') print(f"{x} - {x.dtype}") y = x.astype('c8') print(f"{y} - {y.dtype}")</pre>
rezultat
<pre>[1 2 3] - int32 [1. 2. 3.] - float64 [1.+0.j 2.+0.j 3.+0.j] - complex64</pre>

- Indexarea elementelor unei matrice *unidimensionale* **NumPy** se poate face în mai multe moduri:

Indexare	Descriere
X[i]	selectează elementul cu indexul i din matricea X
X[i:j]	selectează elementele cu indecsii de la i la j - 1 din matricea X
X[:i]	selectează elementele cu indecsii de la 0 la i - 1 din matricea X
X[j:]	selectează elementele cu indecsii de la j la ultimul din matricei X
X[[i, j, k]]	selectează elementele cu indecsii i, j și k din matricea X
X[:, :]	selectează toate elementele din matricea X

- În cazul matricelor *multidimensionale*, indecsii aferenți fiecărei dimensiuni se vor pune între aceleasi paranteze pătrate, separați de virgulă. Pentru indecsii fiecărei dimensiuni se pot utiliza toate metodele prezentate în tabelul anterior.
- Spre exemplu, pentru o matrice bidimensională X, dacă dorim să selectăm elementele care se află pe primele 3 linii (toate coloanele), se va utiliza: A[:3, :].
- Pe lângă indexarea utilizând indecsii elementelor, **NumPy** mai permite și indexarea booleană, cu ajutorul căreia se pot selecta elementele unei matrice care satisfac o anumită condiție.

EXEMPLU

main.py	rezultat
<pre>import numpy as np A = np.array([[8, 7, 6, 1], [5, 4, 3, 7], [2, 1, 0, 4]]) print(A[A > 2])</pre>	[8 7 6 5 4 3 7 4]

1. Introducere

2. Generarea matricelor

3. Operații matematice

4. Alte funcționalități

Generarea matricelor

Operații matematische

Introducere

Alte funcționalități

1. Introducere

2. Generarea matricelor

3. Operații matematice

4. Alte funcționalități

Introducere

Operații matematice

Generarea matricelor

Alte funcționalități

- **NumPy** oferă trei funcții utile pentru generarea de secvențe de numere sub forma unor matrice unidimensionale:

Metoda	Descriere
<code>np.arange(amin, amax, step)</code>	generează o matrice unidimensională ale cărei valori vor reprezenta o secvență de numere de la <code>amin</code> la <code>amax</code> cu pasul <code>step</code> . Dacă <code>step</code> lipsește atunci pasul va fi implicit 1. Atenție: elementul cu valoarea <code>amax</code> nu va fi inclus în secvență
<code>np.linspace(amin, amax, N)</code>	generează o matrice unidimensională ale cărei valori vor reprezenta o secvență de <code>N</code> numere egal distanțate în intervalul $[amin, amax]$
<code>np.logspace(amin, amax, N)</code>	generează o matrice unidimensională ale cărei valori vor reprezenta o secvență de <code>N</code> numere egal distanțate pe scara logaritmică în intervalul $[10^{amin}, 10^{amax}]$

EXEMPLU	
main.py	rezultat
<pre>import numpy as np print(np.arange(1, 11, 2)) print(np.linspace(0, 10, 3)) print(np.logspace(0, 3, 4))</pre>	<pre>[1 3 5 7 9] [0. 5. 10.] [1.e+00 1.e+01 1.e+02 1.e+03]</pre>

- **NumPy** oferă o serie de funcții utile pentru generarea de matrice multidimensionale uzuale:

Metoda	Descriere	Exemplu	Rezultat
<code>np.zeros(t)</code>	generează o matrice cu toate elementele având valoarea 0, ale cărei dimensiuni sunt specificate de tuplul t	<code>np.zeros((2,2))</code>	<code>[[0. 0.]</code> <code>[0. 0.]]</code>
<code>np.ones(t)</code>	generează o matrice cu toate elementele având valoarea 1, ale cărei dimensiuni sunt specificate de tuplul t	<code>np.ones((1,2))</code>	<code>[[1. 1.]]</code>
<code>np.full(t,z)</code>	generează o matrice cu toate elementele având valoarea z, ale cărei dimensiuni sunt specificate de tuplul t	<code>np.full((3,2),7)</code>	<code>[[7 7]</code> <code>[7 7]</code> <code>[7 7]]</code>
<code>np.eye(x)</code>	generează o matrice identitate cu x linii și x coloane	<code>np.eye(2)</code>	<code>[[1. 0.]</code> <code>[0. 1.]]</code>
<code>np.diag(x)</code>	dacă x este unidimensional, va genera o matrice bidimensională cu valorile din x pe diagonala principală, și restul elementelor 0	<code>np.diag([3,2])</code>	<code>[[3. 0.]</code> <code>[0. 2.]]</code>

1. Introducere

2. Generarea matricelor

3. Operații matematice

4. Alte funcționalități

Introducere

Operații matematice

Generarea matricelor

Alte funcționalități

1. Introducere

2. Generarea matricelor

3. Operații matematice

4. Alte funcționalități

Introducere

Generarea matricelor

Operații matematice

Alte funcționalități

- Operațiile aritmetice uzuale (adunare, scădere, înmulțire, împărțire și ridicare la putere) se pot efectua:
 1. Între două matrice **NumPy** – operația se va efectua element cu element; **atenție**: este necesar ca matricele să aibă aceleași dimensiuni, altfel se va returna o eroare.
 2. Între o matrice **NumPy** și un scalar – operația se va realiza între fiecare element al matricei și scalarul respectiv

Exemplu	
main.py	rezultat
<pre>import numpy as np A = np.array([[1,2],[3,4]]) B = np.array([[5,6],[7,8]]) print(A + B) # adunare element cu element print(A - B) # scadere element cu element print(A * B) # inmultire element cu element print(A / B) # impartire element cu element print(A ** B) # ridicare la patrat element cu element print(A + 5) # se adaugă valoarea 5 fiecarui element print(A * 5) # se multiplică cu 5 fiecare element</pre>	$\begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$ $\begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix}$ $\begin{bmatrix} 5 & 12 \\ 21 & 32 \end{bmatrix}$ $\begin{bmatrix} 0.2 & 0.33333333 \\ 0.42857143 & 0.5 \end{bmatrix}$ $\begin{bmatrix} 1 & 64 \\ 2187 & 65536 \end{bmatrix}$ $\begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix}$ $\begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}$

- funcțiile matematice disponibile în librăria **math** care au fost prezentate în primul curs se pot folosi doar pentru scalari
- **NumPy** pune la dispoziție propriul set de funcții matematice care se aplică asupra tuturor elementelor unei matrice:

Constante matematice	
np.pi	Constanta matematică π
np.e	Constanta matematică e
Funcții pentru reprezentarea numerelor	
np.ceil(X)	Rotunjirea tuturor elementelor la cel mai mic întreg mai mare
np.floor(X)	Rotunjirea tuturor elementelor la cel mai mare întreg mai mic
np.round(X, d)	Rotunjirea tuturor elementelor matricei X la d zecimale
np.absolute(X)	Modulul tuturor elementelor matricei
Funcții exponentiale și logaritmice	
np.exp(X)	e la puterea fiecărui element din X
np.log(X)	Logaritmul natural al fiecărui element al matricei
np.log2(X)	Logaritmul în bază 2 al fiecărui element al matricei
np.log10(X)	Logaritmul în bază 10 al fiecărui element al matricei
np.sqrt(X)	Rădăcina pătrată a fiecărui element al matricei
Funcții de conversie unghiulară	
np.rad2deg(X)	Convertește valorile matricei din radiani în grade
np.deg2rad(X)	Convertește valorile matricei din grade în radiani

- funcțiile matematice disponibile în librăria **math** care au fost prezentate în primul curs se pot folosi doar pentru scări
- **NumPy** pune la dispoziție propriul set de funcții matematice care se aplică asupra tuturor elementelor unei matrice:

Funcții trigonometrice	
<code>np.sin(X)</code>	Sinusul fiecărui element al matricei (în radiani)
<code>np.cos(X)</code>	Cosinusul fiecărui element al matricei (în radiani)
<code>np.tan(X)</code>	Tangenta fiecărui element al matricei (în radiani)
<code>np.arcsin(X)</code>	Arc sinusul în radiani al fiecărui element al matricei
<code>np.arccos(X)</code>	Arc cosinusul în radiani al fiecărui element al matricei
<code>np.arctan(X)</code>	Arc tangenta în radiani a fiecărui element al matricei (rezultatul între $-\pi/2$ și $\pi/2$)
<code>np.arctan2(B, A)</code>	Arc tangenta în radiani a fiecărui element al matricei B/A (rezultatul între $-\pi$ și π)
Alte funcții	
<code>np.minimum(X, Y)</code>	Returnează o matrice calculând minimul valorilor din X și Y element cu element
<code>np.maximum(X, Y)</code>	Returnează o matrice calculând maximul valorilor din X și Y element cu element
<code>np.amin(X, axis)</code>	Dacă parametrul <code>axis</code> lipsește va returna valoarea minimă din matrice. Altfel, va returna o matrice cu valorile minime pe axa <code>axis</code> a matricei (<code>axis = 0</code> – minimul fiecărei coloane; <code>axis = 1</code> – minimul fiecărui rând)
<code>np.amax(X, axis)</code>	Dacă parametrul <code>axis</code> lipsește va returna valoarea maxima din matrice. Altfel, va returna o matrice cu valorile maxime pe axa <code>axis</code> a matricei
<code>np.sum(X, axis)</code>	Dacă parametrul <code>axis</code> lipsește va returna suma tuturor elementelor matricei. Altfel, va returna o matrice cu suma elementelor de pe axa <code>axis</code> a matricei
<code>np.prod(X, axis)</code>	Dacă parametrul <code>axis</code> lipsește va returna produsul tuturor elementelor matricei. Altfel, va returna o matrice cu produsul elementelor de pe axa <code>axis</code> a matricei

- funcțiile matematice disponibile în librăria **math** care au fost prezentate în primul curs se pot folosi doar pentru scalari
- **NumPy** pune la dispoziție propriul set de funcții matematice care se aplică asupra tuturor elementelor unei matrice.

Exemplu	
calculăm cosinusul tuturor elementelor unei matrice unidimensionale ce conține 10 valori de la 0 la $\frac{\pi}{2}$	
main.py	rezultat
<pre>import numpy as np A = np.linspace(0, np.pi / 2, 10) print(np.sin(A))</pre>	[0. 0.17364818 0.34202014 0.5 0.64278761 0.76604444 0.8660254 0.93969262 0.98480775 1.]

- librăria **NumPy** permite utilizarea matricelor în rezolvarea problemelor de algebră liniară
- putem determina transpusa unei matrice accesând proprietatea **T**
- putem înmulți două matrice în sens algebric utilizând operatorul **@**

Exemplu	
main.py	<pre>import numpy as np X = np.array([[1, 2, 3], [4, 5, 6]]) print(X @ X.T) # X înmultit cu transpusa acestuia</pre>
rezultat	<pre>[[14 32] [32 77]]</pre>

- librăria **NumPy** permite utilizarea matricelor în rezolvarea problemelor de algebră liniară
- submodulul **linalg** furnizează o serie de funcții utile dedicate algebrei liniare:

Funcția	Descriere
<code>np.linalg.det(X)</code>	Calculează determinantul matricei X
<code>np.linalg.inv(X)</code>	Calculează inversa matricei X
<code>np.linalg.matrix_rank(X)</code>	Calculează rangul matricei X
<code>np.linalg.svd(X)</code>	Descompunerea pe valori singulare a matricei X
<code>np.linalg.eig(X)</code>	Calculează valorile și vectorii proprii ai matricei X
<code>np.linalg.solve(A, B)</code>	Calculează soluțiile sistemelor de ecuații scrise în formă matriceală: $A \cdot x = B$

Exemplu

calcul al determinantului și al inversei unei matrice

main.py

```
import numpy as np

X = np.array([[1,2],[3,4]])

print(np.linalg.det(X))
print(np.linalg.inv(X))
```

rezultat

```
-2.0000000000000004
[[ -2.   1. ]
 [ 1.5 -0.5]]
```

- librăria **NumPy** permite utilizarea matricelor în rezolvarea problemelor de algebră liniară
- submodulul **linalg** furnizează o serie de funcții utile dedicate algebrei liniare:

Funcția	Descriere
...	...
np.linalg.solve(A, B)	Calculează soluțiile sistemelor de ecuații scrise în formă matriceală: $A \cdot x = B$

Exemplu

Rezolvarea sistemului de ecuații liniare $\begin{cases} 2 \cdot x_1 - 3 \cdot x_2 = 5 \\ -x_1 + 2 \cdot x_2 = 8 \end{cases}$ utilizând algebra liniară

Sistemul de ecuații în forma matriceală: $\begin{bmatrix} 2 & -3 \\ -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$ deci $A = \begin{bmatrix} 2 & -3 \\ -1 & 2 \end{bmatrix}$ și $B = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$

main.py

```
import numpy as np

A = np.array([[2,-3],[-1,2]])
B = np.array([[5],[8]])

print(np.linalg.solve(A, B))
```

rezultat

```
[[34.]
 [21.]]
```

- librăria **NumPy** permite utilizarea matricelor în rezolvarea problemelor de algebră liniară
- O altă funcție utilă definită de **NumPy** este funcția np.roots(P) care calculează rădăcinile polinomului ai cărui coeficienți sunt definiți de matricea unidimensională P

Exemplu	
calcularea rădăcinilor polinomului $x^2 - 5 \cdot x + 6 \quad P = [1, -5, 6]$	
main.py	rezultat
<pre>import numpy as np print(np.roots(np.array([1, -5, 6])))</pre>	[3. 2.]

1. Introducere

2. Generarea matricelor

3. Operații matematice

4. Alte funcționalități

Introducere

Generarea matricelor

Operații matematice

Alte funcționalități

1. Introducere

2. Generarea matricelor

3. Operații matematice

4. Alte funcționalități

Introducere

Generarea matricelor

Operații matematice

Alte funcționalități



- pentru a determina dimensiunea unei matrice, se va accesa proprietatea **shape** a acesteia
 - această proprietate este un **tuplu** ce specifică dimensiunile matricei
 - funcția **np.reshape** ne permite să schimbăm formă unei matrice
 - această funcție primește doi parametri:
 - matricea ce se dorește redimensionată
 - noua dimensiune specificată ca un **tuplu**
- și returnează matricea redimensionată

Exemplu	
main.py	rezultat
<pre>import numpy as np A = np.array([[1, 2, 3, 4, 5, 6], [7, 8, 9, 10, 11, 12]]) print(A) print(A.shape) B = np.reshape(A, (3, 4)) print(B) print(B.shape)</pre>	<p>[[1 2 3 4 5 6] [7 8 9 10 11 12]] (2, 6)</p> <p>[[1 2 3 4] [5 6 7 8] [9 10 11 12]] (3, 4)</p>

- concatenarea a două sau mai multe matrice se va face cu funcțiile:
 - `np.hstack` (concatenare pe orizontală a matricelor)
 - `np.vstack` (concatenarea pe verticală a matricelor)
- aceste funcții vor primi ca parametru un **tuplu** ce conține matricele ce se doresc concatenate.

Exemplu	
main.py	rezultat
<pre>import numpy as np X = np.array([1, 2]) Y = np.array([3, 4]) A = np.hstack((X, Y)) print(A) B = np.vstack((X, Y)) print(B)</pre>	<p>[1 2 3 4]</p> <p>[[1 2] [3 4]]</p>

- salvarea matricelor **NumPy** într-un fișier **CSV** se face utilizând funcția **np.savetxt**
- încărcarea unei matrice **NumPy** dintr-un fișier **CSV** se face utilizând funcția **np.loadtxt**.
- aceste două funcții **NU** implică deschiderea prealabilă a fișierului cu ajutorul structurii **with**.

Exemplu	
main.py	rezultat
<pre>import numpy as np X = np.array([[1, 2, 3], [4, 5, 6]]) # scriere in fisier np.savetxt("test.csv", X, delimiter = ",") # citire din fisier Y = np.loadtxt("test.csv", delimiter = ",") print(Y)</pre>	<pre>[[1. 2. 3.] [4. 5. 6.]]</pre>

1. Introducere

2. Generarea matricelor

3. Operații matematice

4. Alte funcționalități

Introducere

Generarea matricelor

Operații matematice

Alte funcționalități

1. Introducere

2. Generarea matricelor

3. Operații matematice

4. Alte funcționalități

Introducere

Generarea matricelor

Operații matematice

Alte funcționalități