

## פעילות 2 - מערכים תחת NumPy Arrays

NumPy היא הספרייה המתמטית של Python. ספרייה זו משתמשת במבנה נתונים ייחודי בשם ndarray העוזר לנו לבצע סדרה גדולה של פעולות מתמטיות. בפעילות זו נעשה הכרות עם מבנה נתונים זה. בהמשך המדריך כאשר נפתח רשות נירונים המבוססת על פרספטרונים נעשה שימוש משמעותי במבנה נתונים זה ובמערך הפעולות שהספרייה NumPy מספקת לנו עבורו. גם ספריות אחרות כדוגמת Matplotlib ו-scikit-learning עושות שימוש במבנה נתונים זה. מקורות:

<http://cs231n.github.io/python-numpy-tutorial/#python-containers>

<https://www.pythoninformer.com/python-libraries/numpy/index-and-slice/>

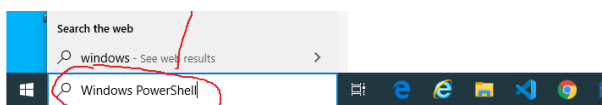
<https://jakevdp.github.io/PythonDataScienceHandbook/02.02-the-basics-of-numpy-arrays.html>

<https://www.programiz.com/python-programming/matrix>

<https://machinelearningmastery.com/index-slice-reshape-numpy-arrays-machine-learning-python/>

### משימה 1: התקנת הספרייה NumPy

נפתח את חלון הפקודות Windows PowerShell.



נבדוק תחילה שהמפרש של Python מותקן במחשב, נעשה זאת על ידי ההוראה:

```
> python --version
```

התקנת הספרייה NumPy תבצע על ידי ההוראה הבאה:

```
> python -m pip install -U numpy
```

נקבל את הפלט הבא:

```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\gadi> python --version
Python 3.7.5
PS C:\Users\gadi> python -m pip install -U numpy
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/a9/38/fe
/numpy-1.18.1-cp37-cp37m-win_amd64.whl (12.8MB)
    |-----| 12.8MB 656kB/s
Installing collected packages: numpy
  Found existing installation: numpy 1.17.4
  Uninstalling numpy-1.17.4:
    Successfully uninstalled numpy-1.17.4
```

## משימה 2: מערך חד מימדי

נכתוב את הקוד הבא:

```
import numpy as np

x = np.array([2, 4, 6, 8, 10])
print(type(x))
print(x.shape)
print(x[0], x[1], x[4])
x[0] = 5
print(x)
```

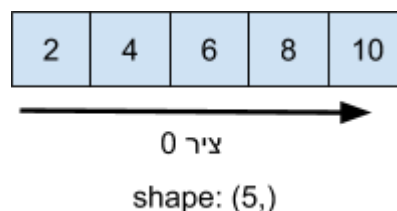
נקבל את הפלט הבא:

```
<class 'numpy.ndarray'>
(5,)
 2  4 10
[ 5  4  6  8 10]
```

נפרש את הוראות התכנית:

<code>x = np.array([2, 4, 6, 8, 10])</code>	הגדרת מערך חד מימדי מטיפוס <code>numpy.ndarray</code> בשם <code>x</code> הכולל 5 איברים.
<code>print(type(x))</code> <code>print(x.shape)</code>	נציג על המסך את סוג טיפוס הנתונים ואת מבנה מערך הנתונים (כלומר מערך חד מימדי הכולל 5 איברים מטיפוס <code>numpy.ndarray</code> ).
<code>print(x[0], x[1], x[4])</code> <code>x[0] = 5</code> <code>print(x)</code>	נציג על המסך חלק מאיברי המערך (כאשר האיבר הראשון הוא איבר מספר 0 והאחרון הוא 4). נשנה את ערכו של האיבר הראשון במערך ל-5 ונציג את כל איברי המערך.

תאר גרפית את המבנה של מערך חד מימדי:



### משימה 3: הגדרת מערך דו מימדי

נכתוב את הקוד הבא:

```
import numpy as np

x2 = np.array([[1, 2, 3, 4, 5],[6, 7, 8, 9, 10]])
print("\nprint all:\n",x2)
print("\ntype: ", type(x2))
print("\nshape: ", x2.shape)
print("\ndata from row0: ",x2[0,0], x2[0,1], x2[0,4])
print("\ndata from row1: ",x2[1,0], x2[1,1], x2[1,4])
print("\nall row0: ",x2[0])
print("\nall row1: ",x2[1])
x2[0,0] = 500
x2[1,4] = 100
print("\nall new array:\n",x2)
```

נקבל את הפלט הבא:

```
print all:
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]

type: <class 'numpy.ndarray'>

shape: (2, 5)

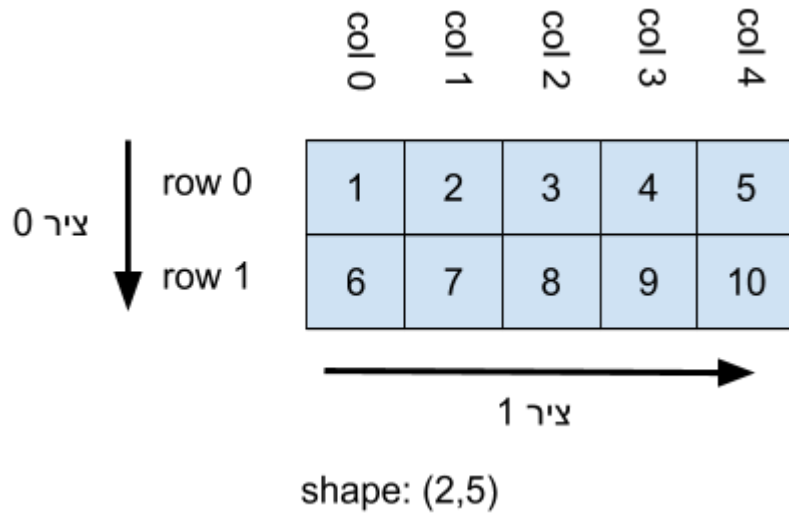
data from row0: 1 2 5
data from row1: 6 7 10

all row0: [1 2 3 4 5]
all row1: [ 6  7  8  9 10]

all new array:
[[500  2  3  4  5]
 [ 6  7  8  9 100]]
```

בדומה למערך חד מימדי גם כאן ניתן לגשת לכל אחד מאיברי המערך, לשנות אותם ולהציג אותם. אך הפעם כל איבר במערך מקבל 2 אינדקסים במבנה הבא:

[row,col]



ניתן לגשת לשורה במערך על ידי ההוראה:

```
print("\nall row0: ",x2[0])
print("\nall row1: ",x2[1])
```

#### משימה 4: מערך שלוש מימדים

נכתוב את הקוד הבא:

```
import numpy as np

x3 = np.array([ [ 1, 2, 3, 4],
                [ 5, 6, 7, 8]
              ],
              [
                [10, 20, 30, 40],
                [50, 60, 70, 80]
              ]
            ])

print("\nprint all:\n",x3)
print("\ntype: ", type(x3))
print("\nshape: ", x3.shape)
print("\nx3[0,0,0]: ",x3[0,0,0])
print("\nx3[1,1,3]: ",x3[1,1,3])
print("\nall row 0,0: ",x3[0,0])
print("\nall row 0,1: ",x3[0,1])
```

```
print("\nall row 1,0: ",x3[1,0])
print("\nall row 1,1: ",x3[1,1])
x3[0,0,0] = 500
x3[1,1,3]=100
print("\nprint all:\n",x3)
```

נקבל את הפלט הבא:

```
print all:
[[[ 1  2  3  4]
 [ 5  6  7  8]]

 [[10 20 30 40]
 [50 60 70 80]]]

type: <class 'numpy.ndarray'>
shape: (2, 2, 4)

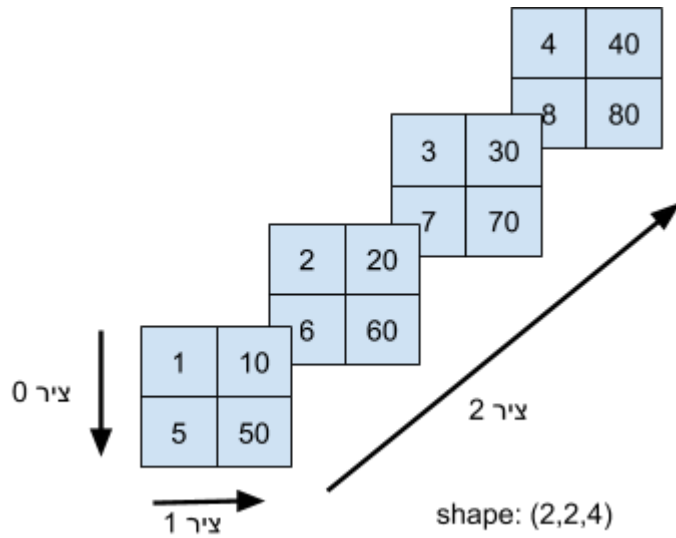
[0,0,0]: 1
[1,1,3]: 80

all row 0,0: [1 2 3 4]
all row 0,1: [5 6 7 8]
all row 1,0: [10 20 30 40]
all row 1,1: [50 60 70 80]

print all:
[[[500  2  3  4]
 [ 5  6  7  8]]

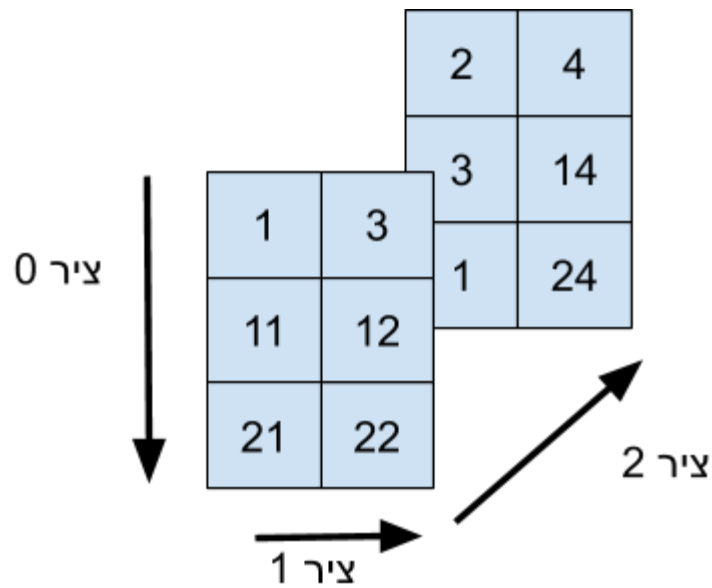
 [[ 10  20  30  40]
 [ 50  60  70 100]]]
```

להלן מבנה המערך X3:



### תרגיל

ממש את המערך הבא בקוד:



1. הציגו את כל אחד מאיברי המערך
2. הציגו את איברי המערך כאשר כל איבר מוכפל ב-2
3. הציגו את כל איברי המערך כאשר מוסיפים לכל איבר 10

### פתרון:

```
import numpy as np
```

```
x3 = np.array([[ [1 , 2 ],[3 , 4 ]],
```

```
[[11, 12],[13, 14] ],
 [[21, 22],[23, 24] ] ] )
print("\n shape: ", x3.shape)

print("\n all:\n",x3)
print("\n all+10: \n",x3+10)
print("\n all*2: \n",x3*2)
```

### משימה 5: אתחול מערך ללא צורך בהכנסת ערכים מראש

פעמים רבות יעלה הצורך ליצור מערכים הכוללים ערכים כבר בשלב האתחול. לא פעם הצורך יהיה ליצור מערך נתונים הכולל מספרים אקראיים או ערכים קבועים שונים לדוגמה מערך שכולו 1 או 0. בחלק של הפעילות נלמד לעשות זאת.

נכתוב את הקוד הבא:

```
import numpy as np

a = np.zeros((4,4))
print("\n print all:\n",a)

b = np.ones((1,8))
print("\n print all:\n",b)

c = np.full((2,2), 7)
print("\n print all:\n",c)

d = np.eye(5,5)
print("\n print all:\n",d)

e = np.random.uniform(size=[3,5])
```

```
print("\n print all:\n",e)
```

```
f = np.random.uniform(-1,1,size=[2,4])
```

```
print("\n print all:\n",f)
```

נקבל את הפלט הבא:

```
print all:
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]

print all:
[[1. 1. 1. 1. 1. 1. 1. 1.]]

print all:
[[7 7]
 [7 7]]

print all:
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]

print all:
[[0.78070304 0.37067051 0.42389852 0.36508368 0.24009258]
 [0.32674762 0.22874655 0.56151192 0.48126036 0.33942365]
 [0.02085075 0.1657081 0.64184528 0.33539173 0.20018986]]

print all:
[[ 0.23045777  0.75048065 -0.83702638 -0.37596959]
 [-0.53248042 -0.12681088 -0.78728514 -0.70618097]]
```

דרכים נוספת לבניית מערך מספרים אקראיים בין 0 ל-1:

```
g = np.random.random_sample((5,))
```

```
print("\n print all:\n",g)
```

```
h = np.random.random_sample((5,5))
```

```
print("\n print all:\n",h)
```

נקבל את הפלט הבא:



```

print all:
[0.24531116 0.08234682 0.07271374 0.50329535 0.68550822]

print all:
[[0.19422442 0.61544653 0.82738011 0.25666235 0.45739554]
 [0.55995805 0.85316693 0.72864517 0.4628477 0.39517055]
 [0.25479628 0.68626459 0.30398616 0.34686117 0.08231422]
 [0.12593126 0.74477394 0.48695433 0.03208739 0.01722872]
 [0.57670587 0.83904837 0.02387257 0.29616319 0.46256324]]

```

### משימה 6: חיתוך מערכים חד מימדיים

נכתוב את הקוד הבא:

```

import numpy as np
x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print("\nprint all:\n",x)
print("start=1:stop=7:step=no ",x[1:7])
print("start=5:stop=no:step=no ",x[5:])
print("start=no:stop=5:step=no ",x[:5])
print("start=1:stop=7:step=2 ",x[1:7:2])
print("start=no:stop=no:step=-1 ",x[::-1])
print("start=-3:stop=-6:step=-1 ",x[-3:-6:-1])

```

נקבל את הפלט הבא:

```

print all:
[ 1  2  3  4  5  6  7  8  9 10]
start=1:stop=7:step=no [2 3 4 5 6 7]
start=5:stop=no:step=no [ 6  7  8  9 10]
start=no:stop=5:step=no [1 2 3 4 5]
start=1:stop=7:step=2 [2 4 6]
start=no:stop=no:step=-1 [10 9 8 7 6 5 4 3 2 1]
start=-3:stop=-6:step=-1 [8 7 6]

```

### משימה 7: חיתוך מערכים דו מימדיים

נכתוב את הקוד הבא:

```

import numpy as np

```

```

x2 = np.array([ [1,2,3,4,5],
                [6,7,8,9,10],
                [11,12,13,14,15],
                [16,17,18,19,20],
                [21,22,23,24,25] ])

print("\nprint all:\n",x2)
print("FROM:start=1:stop=4:step=no TO:start=1:stop=4:step=no\n",x2[1:4,1:4])
print("FROM:start=2:stop=no:step=no TO:start=2:stop=no:step=no\n",x2[2:,:])
print("FROM:start=no:stop=no:step=2 TO:start=no:stop=no:step=2\n",x2[:,2::2])
print("FROM:start=no:stop=no:step=-1 TO:start=no:stop=no:step=-1\n",x2[:,::-1])
print("FROM:start=-2:stop=no:step=-1 TO:start=-2:stop=no:step=-1\n",x2[-2::-1,-2::-1])

```

נקבל את הפלט הבא:

```

print all:
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]
FROM:start=1:stop=4:step=no TO:start=1:stop=4:step=no
[[ 7  8  9]
 [12 13 14]
 [17 18 19]]
FROM:start=2:stop=no:step=no TO:start=2:stop=no:step=no
[[13 14 15]
 [18 19 20]
 [23 24 25]]
FROM:start=no:stop=no:step=2 TO:start=no:stop=no:step=2
[[ 1  3  5]
 [11 13 15]
 [21 23 25]]
FROM:start=no:stop=no:step=-1 TO:start=no:stop=no:step=-1
[[25 24 23 22 21]
 [20 19 18 17 16]
 [15 14 13 12 11]
 [10  9  8  7  6]
 [ 5  4  3  2  1]]
FROM:start=-2:stop=no:step=-1 TO:start=-2:stop=no:step=-1
[[19 18 17 16]
 [14 13 12 11]
 [ 9  8  7  6]
 [ 4  3  2  1]]

```

משימה 8: עיצוב/שינוי מחדש של מבנה מערך חד מימדי

נכתוב את הקוד הבא:

```
import numpy as np
x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print("\n print all:\n",x)
a = np.reshape(x,(5, 2))
print("\n print all:\n",a)
b = np.reshape(x,(10, 1))
print("\n print all:\n",b)
c = np.reshape(x,(2, 5))
print("\n print all:\n",c)
```

נקבל את הפלט הבא:

```
print all:
 [ 1  2  3  4  5  6  7  8  9 10]

print all:
[[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]]

print all:
[[ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]]

print all:
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
```

**משימה 9: עיצוב/שינוי מחדש של מבנה מערך דו מימדי**

נכתוב את הקוד הבא:

```
import numpy as np
x2 = np.array([[1,2,3,4], [5,6,7,8]])
```

```

d = np.reshape(x2,8)
print("\n print all:\n",d)
e = np.reshape(x2,(8,1))
print("\n print all:\n",e)
#print(x2.shape,x2.shape[0],x2.shape[1])
f = np.reshape(x2,(x2.shape[0]*x2.shape[1], 1))
print("\n print all:\n",f)

```

נקבל את הפלט הבא:

```

print all:
[1 2 3 4 5 6 7 8]

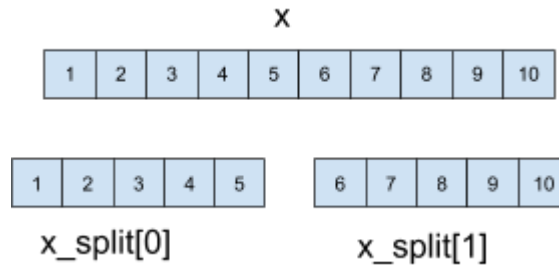
print all:
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]]

print all:
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]]

```

### משימה 10: פיצול מערכים על ידי הפעולה split

הפעולה split המאפשרת לפצל מערך למספר מערכים משנה. באיור הבא ניתן לראות דוגמה גרפית של חלוקת המערך x ל-2 מערכים.



נכתוב את הקוד הבא:

```
import numpy as np
x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
x_split = np.split(x,2)
print("\nprint all:\n",x)
print("\nprint all split:\n",x_split)
print("\nprint array[0]:\n",x_split[0])
print("\nprint array[1]:\n",x_split[1])
```

נקבל את הפלט הבא:

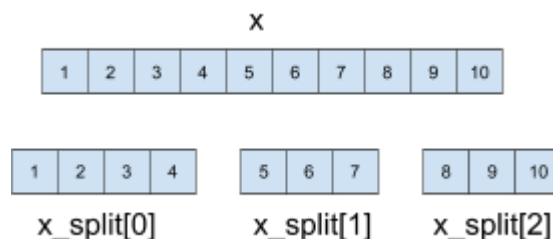
```
print all:
[ 1  2  3  4  5  6  7  8  9 10]

print split all:
[array([1, 2, 3, 4, 5]), array([ 6,  7,  8,  9, 10])]

print array[0]:
[1 2 3 4 5]

print array[1]:
[ 6  7  8  9 10]
```

הפעולה split לא מאפשרת לפצל מערכים בצורה לא סימטרית לדוגמה:



כדי לפצל מערך בצורה כזו ניתן להשתמש בפעולה array\_split נדגים את השימוש בה דרך בקוד הבא:

```
import numpy as np
x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

```
x_split = np.array_split(x,3)
print("\nprint all:\n",x)
print("\nprint all split:\n",x_split)
print("\nprint array[0]:\n",x_split[0])
print("\nprint array[1]:\n",x_split[1])
print("\nprint array[2]:\n",x_split[2])
```

נקבל את הפלט הבא:

```
print all:
[ 1  2  3  4  5  6  7  8  9 10]

print all split:
[array([1, 2, 3, 4]), array([5, 6, 7]), array([ 8,  9, 10])]

print array[0]:
[1 2 3 4]

print array[1]:
[5 6 7]

print array[2]:
[ 8  9 10]
```

**משימה 11: חיבור מערכים על ידי הפעולות vstack ו-hstack**

נדגים חיבור מערכים על ידי דוגמה הבא:

```
import numpy as np
x1 = np.array([1, 2, 3, 4, 5 ])
x2 = np.array([6, 7, 8, 9, 10])
x = np.vstack((x1,x2))
print("\nprint all:\n",x)
y = np.hstack((x1,x2))
print("\nprint all:\n",y)
```

פלט התוכנית יהיה:

```
print all:
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]

print all:
[ 1  2  3  4  5  6  7  8  9 10]
```

**תרגיל**

מה יהיה פלט התוכנית הבא:

```
import numpy as np
x1 = np.array([[1], [2], [3], [4], [5 ]])
```

```
x2 = np.array([[6], [7], [8], [9], [10]])
x = np.vstack((x1,x2))
print("\nprint all:\n",x)
y = np.hstack((x1,x2))
print("\nprint all:\n",y)
```

פתרון:

```
print all:
[[ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]]

print all:
[[ 1 6]
 [ 2 7]
 [ 3 8]
 [ 4 9]
 [ 5 10]]
```

### משימה 12: פעולות אריתמטיות ולוגיות על מערכים

כפי שראינו באחד התרגילים בפעילות זו ניתן לבצע פעולות אריתמטיות על מערך שלם ללא הצורך לעבור על כל אחד מאיברי המערך.

```
import numpy as np
x = np.array([1, 2, 3, 4, 5 ])
x1 = x+10
print("\n x:\n",x)
print("\n x1: \n",x1)
```

נקבל את הפלט הבא:

```
x:
[1 2 3 4 5]

x1:
[11 12 13 14 15]
```

יכולת זו חוסכת לנו את הצורך לכתב הקוד הכולל לולאה שעוברת על כל אחד מאיברי המערך. נדגים זאת על ידי הקוד הבא:

```
import numpy as np
x = np.array([1, 2, 3, 4, 5 ])
x1 = x+10
print("\n x:\n",x)
print("\n x1: \n",x1)

x2 = np.empty([0])
for i in x:
    x2 = np.append(x2, [i+10])
print("\n x1: \n",x2)
```

נדגים פעולות מתמטיות נוספות שניתן לבצע על מערכים:

```
import numpy as np
x1 = np.array([1, 2, 3, 4, 5 ])
x2 = np.array([1, 3, 3, 6, 7 ])
x3 = x1 + x2
x4 = x1 / x2
x5 = x1 * x2
x6 = x1 - x2
print("\n x1 + x2: \n",x3)
print("\n x1 / x2: \n",x4)
print("\n x1 * x2: \n",x5)
print("\n x1 - x2: \n",x6)
```

פלט התוכנית יהיה:

```
x1 + x2:
[ 2  5  6 10 12]

x1 / x2:
[1.          0.66666667 1.          0.66666667 0.71428571]

x1 * x2:
[ 1  6  9 24 35]

x1 - x2:
[ 0 -1  0 -2 -2]
```

ניתן כמובן לבצע גם פעולות לוגיות כמודגם בקוד הבא:

```
import numpy as np
x1 = np.array([1, 2, 3, 4, 5 ])
x2 = np.array([1, 3, 3, 6, 7 ])
```



```
x3 = x1 == x2
x4 = x1 < x2
x5 = x1 > x2
x6 = x1 != x2
print("\n x1 == x2: \n",x3)
print("\n x1 > x2: \n",x4)
print("\n x1 < x2: \n",x5)
print("\n x1 != x2: \n",x6)
```

נקבל את הפלט הבא:

```
x1 == x2:
[ True False  True False False]

x1 > x2:
[False  True False  True  True]

x1 < x2:
[False False False False False]

x1 != x2:
[False  True False  True  True]
```

## תרגיל

מה לדעתכם יהיה פלט הקוד הבא:

```
import numpy as np
x1 = np.array([1, 2, 3, 4, 5 ])
x2 = x1[x1>3]
print("\n x1>3 : \n",x2)
```

## פתרון

הפתרון מדגים דרך מהירה לעבור על כל אחד מהאיברים של המערך x1 ולחפש בו את כל האיברים הגדולים מ-3.

```
x1>3 :
[4 5]
```

## תנאי השימוש

תנאי השימוש במסמך זה הם לפי הסטנדרט הבא:

You are free:

to Share – to copy, distribute and transmit the material  
to Remix – to adapt the material

Under the following conditions:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.