

## Contents

1. Tạo mảng n chiều với các phần tử bằng 1	1
2. Tạo mảng n chiều với các phần tử bằng 0	2
3. Mảng 0 được đóng khung bởi các phần tử 1	3
4. Đổi các giá trị 1 thành 0 và 0 thành 1 trong mảng	4
5. Đảo mảng nhị phân n chiều	6
6. Mặt nạ nhị phân	6
7. Số nguyên từ 0-100	9
8. Số chẵn dương từ 2-1000	10
9. Nhập 1 số để tạo mảng 2 chiều	10
10. Chuyển mảng 1 chiều thành hàng trong mảng 2 chiều	11
11. Đổi từ mảng 1 chiều sang cột trong mảng 2 chiều	12
12. Tạo mảng hàng n chiều	<b>Error! Bookmark not defined.</b>
13. Tạo mảng 2 chiều	15
14. Lọc mảng tạo thành mảng mới	17

Tên	MSSV
Trần Nguyễn Quang Lâm	20139040
Tô Gia Huy	20139003

### 1. Tạo mảng n chiều với các phần tử bằng 1

Tạo mảng 2 chiều với mỗi phần tử bằng 1

Thư viện NumPy có hàm `np.ones()` để dùng cho việc tạo mảng này, dữ liệu đầu vào là định dạng mảng và số phần tử tương ứng, tạo ra mảng theo dữ liệu đã cấp với mỗi phần tử bằng 1

Tạo một mảng với 7 hàng và 5 cột, các phần tử đều bằng 1

```
print(np.ones(5))
x= np.ones((7,5))
print(x)
```

Kết quả

```
>>> Type "help", "copyright",
===== RESTART: C:\Use
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
>>>
```

Code tạo ra một mảng 2 chiều với số cột là 7, số hàng là 5 và các phần tử bằng 1

Giải thích : import numpy as np: gọi đến thư viện numpy và lưu nó dưới tên np

gọi hàm ones với cú pháp np.ones() và truyền vào đối số 7,5 , hàm sẽ hiểu là 7 cột và 5 hàng nên tạo ra mảng có giá trị như hình ở trên

## 2. Tạo mảng n chiều với các phần tử bằng 0

Tạo mảng 2 chiều với tất cả phần tử bằng 0

Thư viện NumPy có hàm *np.zeros()* để dùng cho việc tạo mảng này, dữ liệu đầu vào là định dạng mảng và số phần tử tương ứng, tạo ra mảng theo dữ liệu đã cấp với mỗi phần tử bằng 0

Ví dụ:

```
fix.py - C:\Users\CCLaptop\Documents\pYTHON
File Edit Format Run Options Window
import numpy as np
x = np.zeros((5,))
print(x)
```

```
[0. 0. 0. 0. 0.]
```

Tạo ra mảng 2 chiều với số cột là 4 và số hàng là 6, tất cả phần tử bằng 0

```
fix.py - C:\Users\CCLaptop\Documents\pYTHON_PR\Week2\fix.py (3.10.6)
File Edit Format Run Options Window Help
import numpy as np
x = np.zeros((4,6))
print(x)
```

```

                                Kết quả
===== RESTART: C:\Users\CCLaptop\
[0. 0. 0. 0. 0.]
>>>
===== RESTART: C:\Users\CCLaptop\
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
>>>
```

Giải thích : `import numpy as np`: gọi đến thư viện numpy và lưu nó dưới tên `np`

Gọi hàm `ones` với cú pháp `np.zeros()` và truyền vào đối số 4, 6, hàm sẽ hiểu là 6 cột và 4 hàng nên tạo ra mảng có giá trị như hình ở trên

### 3. Mảng 0 được đóng khung bởi các phần tử 1

Triển khai hàm `framed_zeros()`. Với dữ liệu đầu vào là số cột và hàng, hàm này tạo ra mảng 2 chiều tương ứng với số cột hàng đã cấp, trong đó các phần tử bằng 0, ngoại trừ các phần tử ở hàng và cột ngoài cùng bằng 1

Đầu tiên khai báo hàm `framed_zeros()` với 2 đối số:

+`num_rows`: số hàng của mảng

+`num_cols`: số cột của mảng

Triển khai hàm để nó trả về mảng 2 chiều với kích thước đã cho, các phần tử bằng 0 ngoại trừ hàng và cột ngoài cùng:

+Dùng hàm `np.ones()` để khởi tạo tất cả các giá trị bằng 1

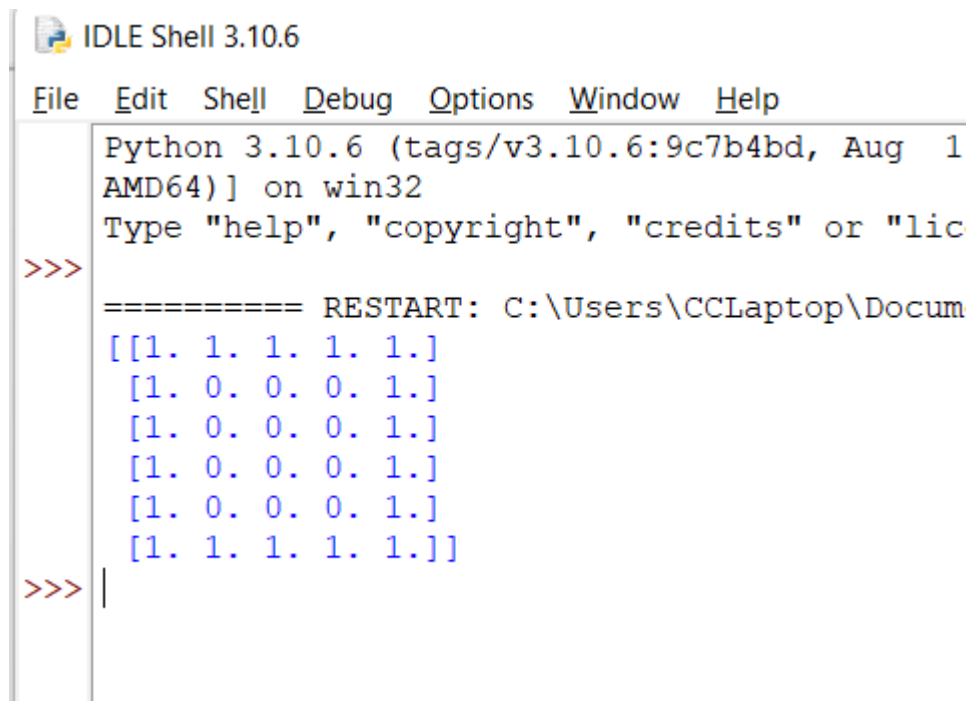
+Đổi tất cả giá trị ở trong mảng về 0 : hàng và cột kế ngoài cùng đều có vị trí từ 1:-1

fix.py - C:\Users\CCLaptop\Documents\pYTHON\_PR\Week2\fix.py (3.10.6)

File Edit Format Run Options Window Help

```
import numpy as np
def framed_zeros(num_rows, num_cols):
    x=np.ones((num_rows, num_cols))
    x[1:-1, 1:-1] = 0
    return x
x = framed_zeros(6,5)
print(x)
```

Kết quả:



```
IDLE Shell 3.10.6
File Edit Shell Debug Options Window Help
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug 1
AMD64) on win32
Type "help", "copyright", "credits" or "lic
>>>
===== RESTART: C:\Users\CCLaptop\Docum
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
>>> |
```

Giải thích : import numpy as np: gọi đến thư viện numpy và lưu nó dưới tên np

Khai báo hàm framed\_zeros() với 2 đối số là cột và hàng, khởi tạo mảng với kích thước đã cho, tất cả phần tử bằng 1. **x= np.ones(6,5)**

-Đổi tất cả các phần tử từ hàng 2 cột 2 đến hàng và cột kế cuối (vị trí là [1:-1,1:-1] thành 0 : **x[1:-1,1:-1]=0**, sau đó trả về x

#### 4. Đổi các giá trị 1 thành 0 và 0 thành 1 trong mảng

Tập triển khai hàm **alternate\_ones\_zeros()**, dữ liệu đầu vào là số hàng và cột, tạo ra mảng 2 chiều với kích cỡ đã cho, giá trị các phần tử lần lượt là 1, 0

Đầu tiên triển khai hàm **alternate\_ones\_zeros()** với 2 đối số truyền vào

-**num\_rows**: số hàng

-**num\_cols**: số cột

Triển khai hàm để nó trả về mảng 2 chiều với kích thước đã cho, các phần tử lần lượt bằng 0 và 1:

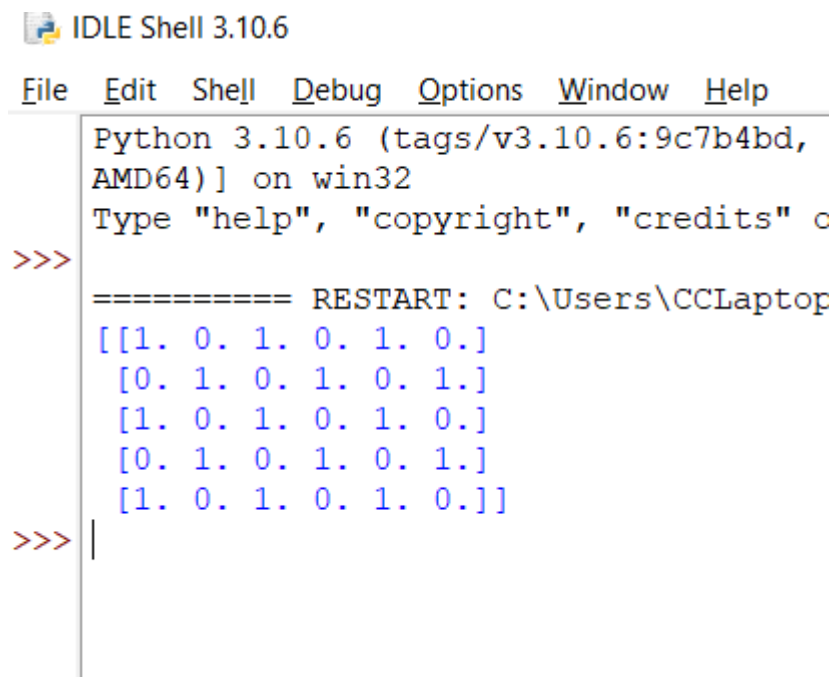
-Khởi tạo mảng với tất cả các giá trị bằng 0 hoặc bằng 1

-Cho các giá trị xen kẽ nhau với khoảng cách là 2 (::2) =1 hoặc bằng 0 nếu khởi tạo mảng với giá trị 1

Code:

```
def alternate_ones_zeros(num_rows, num_cols):  
    x = np.zeros((num_rows, num_cols))  
    x[::2, ::2]=1  
    x[1::2, 1::2]=1  
    return x  
x = alternate_ones_zeros(5,6)  
print(x)
```

Kết quả



```
IDLE Shell 3.10.6  
File Edit Shell Debug Options Window Help  
Python 3.10.6 (tags/v3.10.6:9c7b4bd, AMD64) on win32  
Type "help", "copyright", "credits" c  
>>>   
===== RESTART: C:\Users\CCLaptop  
[[1. 0. 1. 0. 1. 0.]  
 [0. 1. 0. 1. 0. 1.]  
 [1. 0. 1. 0. 1. 0.]  
 [0. 1. 0. 1. 0. 1.]  
 [1. 0. 1. 0. 1. 0.]]  
>>> |
```

Giải thích: Đề yêu cầu tạo ra một mảng với các giá trị 1, 0 xen kẽ nên ta chỉ cần

-Tạo mảng toàn bộ giá trị = 0: **x=np.zeros(5,6)**

-Bắt đầu chạy từ hàng 0, cột 0 và cách mỗi hàng mỗi cột sẽ thay đổi giá trị thành 1 qua lệnh **x[::2,::2]=1** VD: x[0,0]=1 , x[0,2]=1, x[0,4]=1, ....x[2,0], x[2,2]=1 và tương tự. Xử lý hết các hàng và cột chẵn

-Đổi tiếp : **x[1::2,1::2]=1**. Hàng bắt đầu từ 1 và Cột bắt đầu từ 1. Ta xử lý hết các hàng và cột lẻ

## 5. Đảo mảng nhị phân n chiều

Triển khai hàm `invert_binary()`, dữ liệu đầu vào một mảng, trả về mảng có phần tử = 0 tại vị trí phần tử = 1 và trả về phần tử = 0 tại vị trí phần tử bằng 1. Hay nói đơn giản là thay 0 thành 1, thay 1 thành 0.

Các bước triển khai hàm `invert_binary()`:

- Khai báo hàm với một đối số

- + x mảng chứa các phần tử 1 và 0

- Triển khai hàm để trả về kết quả 1 tại vị trí phần tử 0 và trả về 0 tại vị trí phần tử 1: lấy phần bù  $x[i,j] = 1 - x$

```
def invert_binary(x):  
    return 1-x  
x=np.array([  
    [1,0,0,1],  
    [0,0,1,1],  
    [0,0,1,1]  
])  
print(invert_binary(x))
```

Kết quả:

```
>>>   
===== RESTART: C:\Users\  
[[1. 0. 1. 0. 1. 0.]  
 [0. 1. 0. 1. 0. 1.]  
 [1. 0. 1. 0. 1. 0.]  
 [0. 1. 0. 1. 0. 1.]  
 [1. 0. 1. 0. 1. 0.]  
>>>
```

Giải thích thuật toán: khởi tạo hàm `invert_binary` để đổi giá trị các phần tử trong mảng, lấy phần bù để từ 1 ra 0 và từ 0 ra 1 (return 1-x)

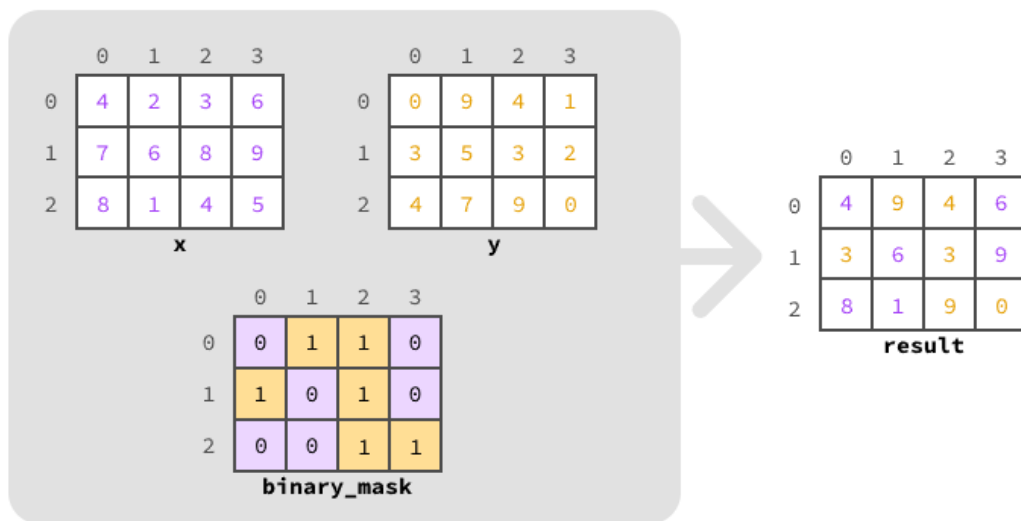
Khởi tạo một mảng x với các phần tử 0 và 1. Gọi hàm `invert_binary` đã tạo và truyền vào đối số chính là mảng x, kết quả trả về là mảng x với các phần tử đã được biến đổi lấy phần bù.

## 6. Mặt nạ nhị phân

Sử dụng một mặt nạ với các phần tử nhị phân sẽ cho thấy những gì ta muốn cho thấy và che đi những phần tử ta không muốn xuất ra kết quả

Triển khai hàm `merge_with_binary_mask()` để kết hợp 2 mảng với cùng kích cỡ, sử dụng binary mask

The following figure illustrates how this works:



Kết quả mảng trả về sẽ là mảng có cùng kích thước với x và y, có phần tử của x tại các vị trí mà binary mask = 0 và có các phần tử của y tại các vị trí mà binary mask = 1

```
x=np.array([
    [4,8,0,9],
    [1,5,6,8],
    [9,7,7,7]])
y=np.array([
    [7,8,4,5],
    [2,3,3,4],
    [8,4,7,6]])
bi=np.array([
    [0,0,0,0],
    [1,0,1,1],
    [1,0,1,1]])
print(merge_with_binary_mask(x,y,bi))
```

```
[[4 8 0 9]
 [2 5 3 4]
 [8 7 7 6]]
```

Các bước triển khai hàm `merge_with_binary_mask()` với 3 đối số

-x: mảng x

-y: mảng y

-binary\_mask mảng nhị phân với kích thước giống như x

Triển khai `merge_with_binary_mask()` để nó trả về phần tử của x tại các vị trí mà phần tử của mảng binary mask này = 0 và phần tử của y tại vị trí các phần tử mảng mask này = 1

```
6_binary_mask.py > ...
1  import numpy as np
2  def merge_with_binary_mask(x,y,binary_mask):
3      return x*(1-binary_mask)+y*(binary_mask)
4  x=np.array([
5      [4,8,0,9],
6      [1,5,6,8],
7      [9,7,7,7]])
8  y=np.array([
9      [7,8,4,5],
10     [2,3,3,4],
11     [8,4,7,6]])
12  bi=np.array([
13     [0,0,0,0],
14     [1,0,1,1],
15     [1,0,1,1]])
16  print(merge_with_binary_mask(x,y,bi))
```

```
[[4 8 0 9]
 [2 5 3 4]
 [8 7 7 6]]
```



## 7. Số nguyên từ 0-100

Tạo mảng 1 chiều chứa số từ 0 - 100

You might want to look at the `numpy.arange()` [function](#). This function can be used to create a ndarray with all numbers between two given values.

Sử dụng `numpy.arange()` function. Function này dùng để tạo mảng với tất cả các số ở giữa 2 số mình nhập.

VD tạo mảng chứa các giá trị từ 3- 10

```
7_array.py > ...  
1  import numpy as np  
2  a=np.arange(3,11)  
3  print(a)
```

```
[ 3  4  5  6  7  8  9 10]
```

Tạo mảng 1 chiều từ 0 -100:

```
7_array.py > ...  
1  import numpy as np  
2  # a=np.arange(3,11)  
3  # print(a)  
4  b=np.arange(0,101)  
5  print(b)
```

```
[ 0  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 26 27 28 29 30 31 32 33 34 35  
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  
54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71  
72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
90 91 92 93 94 95 96 97 98 99 100]
```

## 8. Số chẵn dương từ 2-1000

Tạo mảng 1 chiều với số chẵn từ 2 đến 1000.

Sử dụng `numpy.arange()` [function](#). Chúng ta sẽ thêm bước nhảy cho hàm để tạo mảng theo ý của mình.

VD: tạo mảng số lẻ (1,11):

```
print(np.arange(1,12,2))
```

```
[ 1  3  5  7  9 11]
```

Tạo mảng 1 chiều chứa các số chẵn từ 2 đến 1000

```
even_2_to_1000= np.arange(2,1001,2)
print(even_2_to_1000)
#cách 2
even_2_to_1000= np.array([i for i in range(2,1001,2)])
print(even_2_to_1000)
```

```
[  2   4   6   8  10  12  14  16  18  20  22  24  26  28
 30  32  34  36  38  40  42  44  46  48  50  52  54  56
 58  60  62  64  66  68  70  72  74  76  78  80  82  84
 86  88  90  92  94  96  98 100 102 104 106 108 110 112
114 116 118 120 122 124 126 128 130 132 134 136 138 140
142 144 146 148 150 152 154 156 158 160 162 164 166 168
.....
926 928 930 932 934 936 938 940 942 944 946 948 950 952
954 956 958 960 962 964 966 968 970 972 974 976 978 980
982 984 986 988 990 992 994 996 998 1000]
```

## 9. Nhập 1 số để tạo mảng 2 chiều

Nhập vào 1 số và tạo mảng 2 chiều có số cột và số hàng của số đó

1. định nghĩa một hàm `numbered_table()` với 1 đối số: `size` (là số hàng và cột của mảng 2 chiều ta tạo)
2. Thực hiện hàm đó và trả về mảng (`size, size`) chứa từ 1 đến `size` mũ 2

```
9_numbertable.py > ...
1  import numpy as np
2  def numbered_table(size):
3      return np.arange(1, size*size+1).reshape((size, size))
4  def numbered_table_2(size):
5      return np.array([i for i in range(1, size*size+1)]).reshape(size, size)
6  print(numbered_table_2(5))
7  print(numbered_table(3))
8
```

```

C:\UNIVERSITY\hamsk11\IT_Python\bu
[[ 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]]
[[1 2 3]
 [4 5 6]
 [7 8 9]]

```

Giải thích: thư viện numpy hỗ trợ hàm `reshape`, biến mảng 1 chiều thành mảng n chiều mà ta muốn

## 10. Chuyển mảng 1 chiều thành hàng trong mảng 2 chiều

Chạy hàm `dim1_to_dim2()` làm mảng 1 chiều thành mảng 2 chiều với 1 hàng chứa các giá trị cũ.

```

x=np.array([4,2,7,1])
print(dim1_to_dim2(x))

```

```

[[4 2 7 1]]

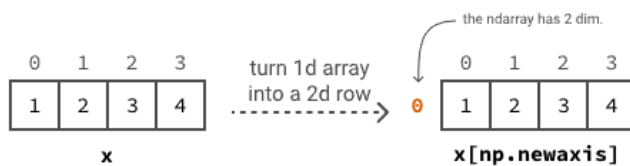
```

Dấu ngoặc kép trong output cho ta biết đó là 1 mảng 2 chiều với 1 hàng duy nhất

Cách 1 để làm là sử dụng hàm `ndarray.reshape()`.

Cách 2 là chuyển mảng 1 chiều vào mảng 2 chiều bằng hàm `np.newaxis` [constant](#):

```
x = x[np.newaxis] # same as doing x = x[np.newaxis, :]
```



Ta có thể biến mảng 1 chiều thành mảng 2 chiều với 1 cột duy nhất ta làm:

```
x=x[:,np.newaxis]  
print(x)
```

Kết quả:

```
[[4]  
 [2]  
 [7]  
 [1]]
```

```
import numpy as np  
def dim1_to_dim2(x):  
    return x.reshape(1,len(x))  
def dim1_to_dim2_cach_2(x):  
    return x[np.newaxis]  
x=np.array([4,2,7,1])  
print(dim1_to_dim2(x))  
print(dim1_to_dim2_cach_2(x))
```

```
[[4 2 7 1]]  
[[4 2 7 1]]
```

## 11. Đổi từ mảng 1 chiều sang cột trong mảng 2 chiều

Chạy hàm `row_to_col()`, hàm cần input mảng 1 chiều và trả về mảng 2 chiều với 1 cột bao gồm giá trị cũ.

Example of use:

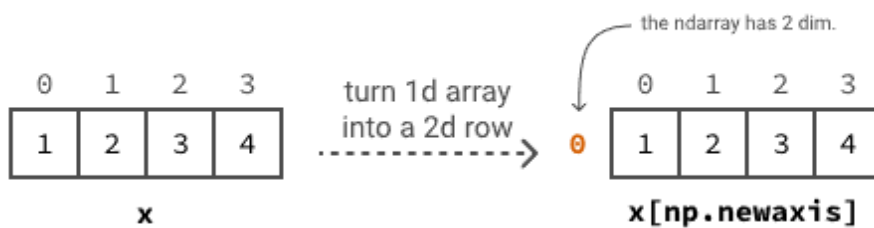
```
x = np.array([4, 2, 7, 1])  
print(row_to_col(x))
```

```
[[4]  
 [2]  
 [7]  
 [1]]
```

Một cách để đạt được điều này là sử dụng `ndarray.reshape()`. Một cách thay thế để biến mảng 1 chiều thành mảng 2 chiều là sử dụng hằng số `np.newaxis`.

Để làm mảng 1 chiều thành mảng 2 chiều ta có thể làm

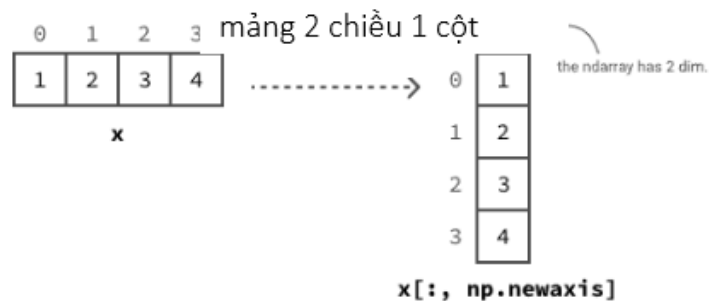
```
x = x[np.newaxis] # same as doing x = x[np.newaxis, :]
```



Để biến mảng 1 chiều thành mảng 2 chiều với một cột duy nhất, chúng ta có thể thực hiện:

```
x=x[:, np.newaxis]
```

Biến mảng 1 chiều thành



1\_col.py > ...

```
import numpy as np
def row_to_col(x):
    return x.reshape(len(x),1)
def row_to_col_cach_2(x):
    return x[:,np.newaxis]
x=np.array([4,2,7,1])
print(row_to_col(x))
print(row_to_col_cach_2(x))
```

```
[[4]
 [2]
 [7]
 [1]]
[[4]
 [2]
 [7]
 [1]]
```

## 12. Tạo mảng hằng n chiều

Tạo mảng 10x10 mỗi phần tử bằng 5:

Dùng `numpy.full()` [function](#) để làm:

Assign to a variable `fives`, an ndarray with shape `(10, 10)` whose entries are all equal to 5.

```
12_array2.py > ...  
1  import numpy as np  
2  fives=np.full((10,10),5)  
3  print(fives)
```

```
[[5 5 5 5 5 5 5 5 5 5]  
 [5 5 5 5 5 5 5 5 5 5]  
 [5 5 5 5 5 5 5 5 5 5]  
 [5 5 5 5 5 5 5 5 5 5]  
 [5 5 5 5 5 5 5 5 5 5]  
 [5 5 5 5 5 5 5 5 5 5]  
 [5 5 5 5 5 5 5 5 5 5]  
 [5 5 5 5 5 5 5 5 5 5]  
 [5 5 5 5 5 5 5 5 5 5]  
 [5 5 5 5 5 5 5 5 5 5]]
```

### 13. Tạo mảng 2 chiều

Tạo 1 mảng như hình ở dưới :

	0	1	2	3	4	5	6	7	8	9
0	1	1	1	1	1	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2	2
2	1	1	1	1	1	2	2	2	2	2
3	1	1	1	1	1	2	2	2	2	2
4	1	1	1	1	1	2	2	2	2	2
5	3	3	3	3	3	4	4	4	4	4
6	3	3	3	3	3	4	4	4	4	4
7	3	3	3	3	3	4	4	4	4	4
8	3	3	3	3	3	4	4	4	4	4
9	3	3	3	3	3	4	4	4	4	4

Tạo chương trình ndarray trong sơ đồ. Gán kết quả cho một biến x.

```

13_2D_array.py > ...
1  import numpy as np
2  x=np.ones((9,9))
3  x[0:4,5:9]=2
4  x[5:9,0:4]=3
5  x[5:9,5:9]=4
6  print(x)

```



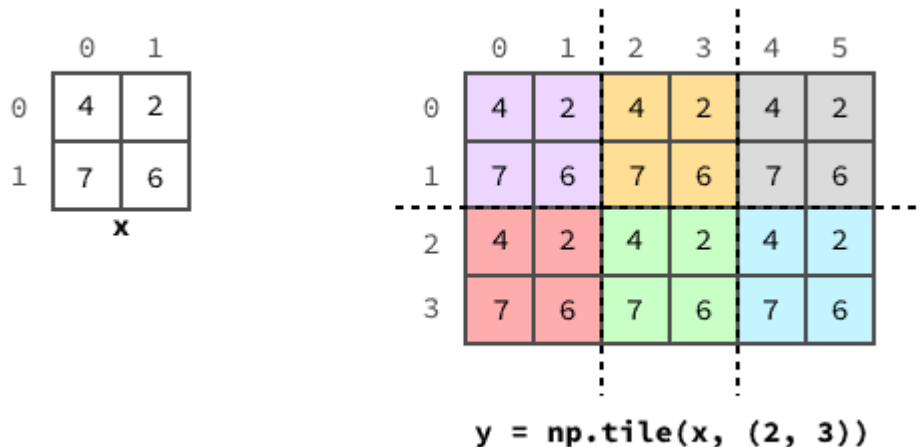
```
[[1. 1. 1. 1. 1. 2. 2. 2. 2.]
 [1. 1. 1. 1. 1. 2. 2. 2. 2.]
 [1. 1. 1. 1. 1. 2. 2. 2. 2.]
 [1. 1. 1. 1. 1. 2. 2. 2. 2.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [3. 3. 3. 3. 1. 4. 4. 4. 4.]
 [3. 3. 3. 3. 1. 4. 4. 4. 4.]
 [3. 3. 3. 3. 1. 4. 4. 4. 4.]
 [3. 3. 3. 3. 1. 4. 4. 4. 4.]]
```

## 14. Lọc mảng tạo thành mảng mới

Tạo 1 mảng như bên dưới:

	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	1	2	3	4	5
1	6	7	8	9	10	6	7	8	9	10
2	11	12	13	14	15	11	12	13	14	15
3	16	17	18	19	20	16	17	18	19	20
4	21	22	23	24	25	21	22	23	24	25
5	1	2	3	4	5	1	2	3	4	5
6	6	7	8	9	10	6	7	8	9	10
7	11	12	13	14	15	11	12	13	14	15
8	16	17	18	19	20	16	17	18	19	20
9	21	22	23	24	25	21	22	23	24	25

Một hàm có thể giúp bạn đạt được điều này là hàm `numpy.tile()`. Nó cho phép ta xây dựng một ndarray bằng cách lặp lại một ndarray đã cho. Sơ đồ sau đây cho thấy một ví dụ trong đó chúng ta sử dụng hàm `numpy.tile()` để xếp một mảng 2 x 2.



Đối số thứ hai trong hàm `numpy.tile ()` là `shape (2, 3)`. Đối số này chỉ định rằng chúng ta muốn sao chép `x` hai lần theo chiều dọc và ba lần theo chiều ngang.

```

14_Narray.py > ...
1  import numpy as np
2  x=np.arange(1,26).reshape(5,5)
3  x=np.tile(x,(2,2))
4  print(x)

```

```

[[ 1  2  3  4  5  1  2  3  4  5]
 [ 6  7  8  9 10  6  7  8  9 10]
 [11 12 13 14 15 11 12 13 14 15]
 [16 17 18 19 20 16 17 18 19 20]
 [21 22 23 24 25 21 22 23 24 25]
 [ 1  2  3  4  5  1  2  3  4  5]
 [ 6  7  8  9 10  6  7  8  9 10]
 [11 12 13 14 15 11 12 13 14 15]
 [16 17 18 19 20 16 17 18 19 20]
 [21 22 23 24 25 21 22 23 24 25]]

```

Giải thích: ta tạo mảng `x` 1 chiều gán các giá trị từ 1 đến 25, sau đó reshape lại thành mảng 2 chiều 5 hàng 5 cột. Cuối cùng là dùng hàm `tile(x,(2,2))` để sao chép `x` 2 lần theo chiều dọc và 2 lần theo chiều ngang