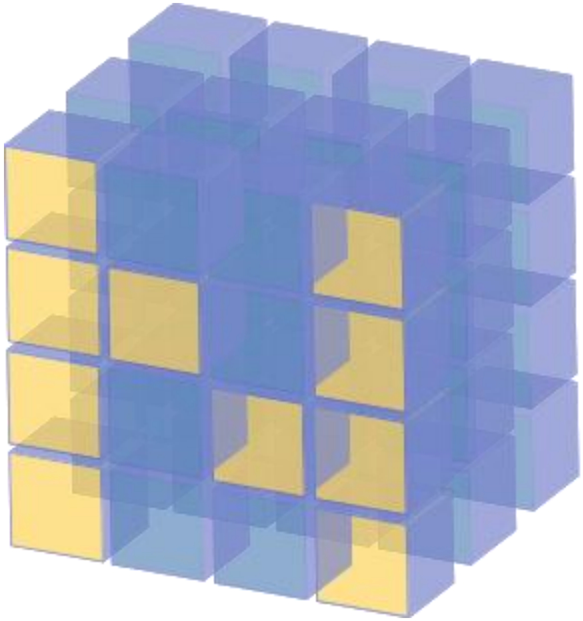


NumPy

NumPy

- Numerical Python
- Modul/library yang berisi fungsi-fungsi matematis dan saintifik
- Merupakan library yang “wajib”
- Digunakan oleh beberapa library yang lain: scikit-learn, Pandas, OpenCV, dll
- Instalasi (via command prompt): **`pip install numpy`**

Impor modul NumPy

```
import numpy as np
```

Membuat dan Mencetak Array NumPy

```
a = np.array([1, 2, 3])
b = np.zeros(5)
c = np.ones(5)
d = np.arange(5)

print(a)  # [1, 2, 3]
print(b)  # [0. 0. 0. 0. 0. ]
print(c)  # [1. 1. 1. 1. 1.]
print(d)  # [0 1 2 3 4]
```

Array Python dan NumPy

- Array Numpy berbeda dengan array Python
- Array NumPy lebih baik dalam hal ukuran data, kecepatan, dan fungsionalitas

Array Python dan NumPy

```
# Array Python
a = [1, 2, 3]

# Array NumPy
b = np.array([1, 2, 3])

print(type(a))
# <class 'list'>

print(type(b))
# <class 'numpy.ndarray'>
```

Properti Array NumPy

```
a = np.array([1, 2, 3])

print(a.size)
# 3

print(a.shape)
# (3,)

print(a.ndim)
# 1

print(a.dtype)
# int32
```

Tipe Data

- Tipe data pada suatu array NumPy harus sama/seragam
- Beberapa tipe data yang tersedia di NumPy: **`bool`**, **`int8`**, **`int16`**, **`int32`**, **`int64`**, **`uint8`**, **`uint16`**, **`uint32`**, **`uint64`**, **`float16`**, **`float32`**, **`float64`**, **`complex64`**, **`complex128`**

Menentukan Tipe Data

```
a = np.array([1, 2, 3], np.uint8)

print(a.dtype)
# uint8
```

Mengubah Tipe Data

```
a = a.astype(np.int32)

print(a.dtype)
# int32
```

Operasi Aritmetik

```
a = np.array([1, 2, 3])
b = np.zeros(a.shape)

# Dengan perulangan
for i in range(a.size):
    b[i] = a[i] + 1

print(b)
# [2 3 4]
```

Operasi Aritmetik

```
a = np.array([1, 2, 3])

b = a + 1
# [2 3 4]

c = a - 1
# [0 1 2]

d = a * 3
# [2 4 6]
```

Operasi Aritmetik

```
e = a / 2
# [0.5 1 1.5]

f = a // 2
# [0 1 1]

g = a ** 2
# [1 4 9]
```

Statistik

```
a = np.arange(11, 21)
# [11 12 13 14 15 16 17 18 19 20]

print(np.mean(a))
# 15.5

print(np.var(a))
# 8.25

print(np.std(a))
# 2.8722813232690143
```

Statistik

```
print(np.sum(a))
# 155

# Peak-to-peak (max-min)
print(np.ptp(a))
# 9
```

Histogram

```
a = np.random.randint(0, 10, 20)
# [7 4 9 5 3 5 8 3 1 3 7 1 6 5 7 9 3 3 1 0]

hist = np.histogram(a)[0]

print(hist)
# [3 2 3 3 1 0 0 3 2 3]
```

Pengurutan

```
a = np.random.randint(0, 10, 20)
# [7 4 9 5 3 5 8 3 1 3 7 1 6 5 7 9 3 3 1 0]

print(np.sort(a))
# [0 1 1 1 3 3 3 3 3 4 5 5 5 6 7 7 7 8 9 9]
```

Memotong Array

```
a = np.arange(11, 21)
```

Indeks	0	1	2	3	4	5	6	7	8	9
Nilai	11	12	13	14	15	16	17	18	19	20

Memotong Array

```
print(a[2:5])
# [13 14 15]
```

Indeks	0	1	2	3	4	5	6	7	8	9
Nilai	11	12	13	14	15	16	17	18	19	20

Memotong Array

```
print(a[:5])
# [11 12 13 14 15]
```

Indeks	0	1	2	3	4	5	6	7	8	9
Nilai	11	12	13	14	15	16	17	18	19	20

Memotong Array

```
print(a[5:])
# [16 17 18 19 20]
```

Indeks	0	1	2	3	4	5	6	7	8	9
Nilai	11	12	13	14	15	16	17	18	19	20

Memotong Array

```
print(a[:-1])
# [11 12 13 14 15 16 17 18 19]
```

Indeks	0	1	2	3	4	5	6	7	8	9
Nilai	11	12	13	14	15	16	17	18	19	20

Memotong Array

```
print(a[2:7:2])
# [13 15 17]
```

Indeks	0	1	2	3	4	5	6	7	8	9
Nilai	11	12	13	14	15	16	17	18	19	20

Memotong Array

```
idx = [1, 4, 8]

print(a[idx])
# [12 15 19]
```

Indeks	0	1	2	3	4	5	6	7	8	9
Nilai	11	12	13	14	15	16	17	18	19	20

Bantuan

```
np.info(np.mean)
 mean(a, axis=None, dtype=None, out=None, keepdims=<no
value>)

Compute the arithmetic mean along the specified axis.

Returns the average of the array elements.  The average
is taken over
the flattened array by default, otherwise over the
specified axis.
`float64` intermediate and return values are used for
integer inputs.
...
```

Matriks

Membuat Matriks

```
A = np.array([[11, 12, 13, 14],
              [15, 16, 17, 18],
              [19, 20, 21, 22],
              [23, 24, 25, 26]])
```

11	12	13	14
15	16	17	18
19	20	21	22
23	24	25	26

Membuat Matriks

```
A = np.arange(11, 27).reshape(4, 4)
```

11	12	13	14
15	16	17	18
19	20	21	22
23	24	25	26

Membuat Matriks

```
B = np.zeros((4, 4), dtype=np.int8)
```

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Membuat Matriks

```
C = np.ones((4, 4), dtype=np.int8)
```

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Membuat Matriks

```
D = np.full((4, 4), 5, dtype=np.int8)
```

5	5	5	5
5	5	5	5
5	5	5	5
5	5	5	5

Properti Matrix

```
print(A.ndim)
# 2

print(A.shape)
# (4, 4)

h, w = A.shape

print(A.size)
# 16
```

Transpose Matriks

```
E = A.T
```

11	12	13	14
15	16	17	18
19	20	21	22
23	24	25	26

11	15	19	23
12	16	20	24
13	17	21	25
14	18	22	26

Determinan Matriks

```
F = np.arange(2, 6).reshape(2, 2)

print(F)
# [[2 3]
   [4 5]]

print(np.linalg.det(F))
# -2.0
```

Eigenvalue dan Eigenvector

```
F = np.arange(2, 6).reshape(2, 2)

print(F)
# [[2 3]
   [4 5]]

print(np.linalg.eig(F))
# (array([-0.27491722,  7.27491722]),
   array([[-0.79681209, -0.49436913],
          [ 0.60422718, -0.86925207]]))
```

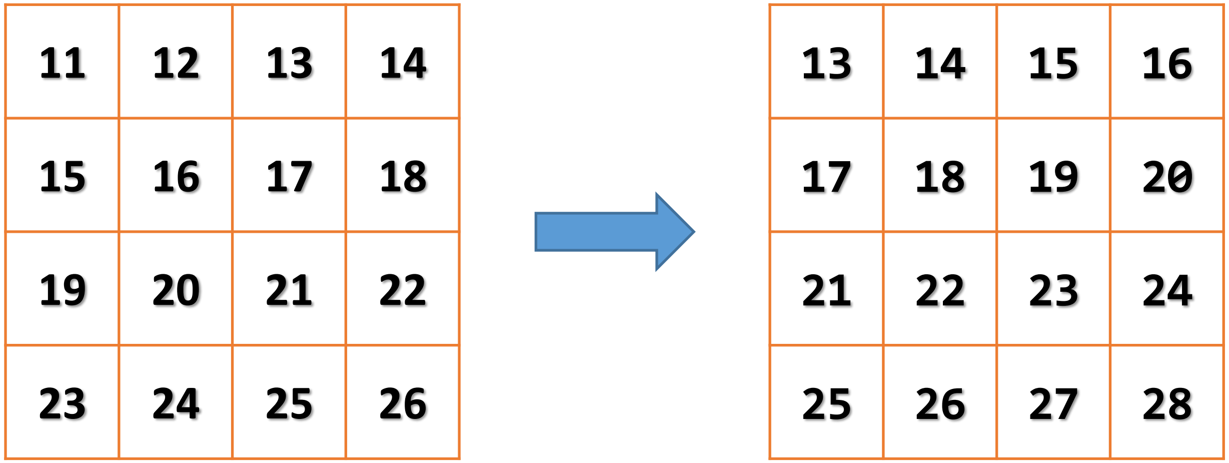
Operasi Matriks

```
G = np.empty(A.shape, A.dtype)
h, w = G.shape

# Dengan perulangan
for y in range(h):
    for x in range(w):
        G[y, x] = A[y, x] + 2
```

13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28

Operasi Matriks

```
H = A + 2
```



11	12	13	14
15	16	17	18
19	20	21	22
23	24	25	26

13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28

Operasi Matriks

```
I = A - 2
```

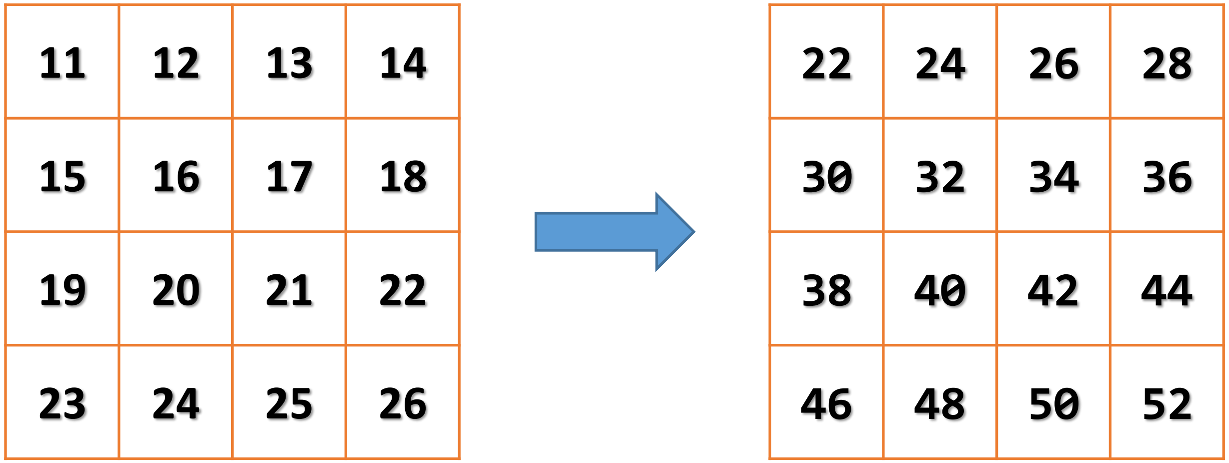
11	12	13	14
15	16	17	18
19	20	21	22
23	24	25	26

→

9	10	11	12
13	14	15	16
17	18	19	20
21	22	23	24

Operasi Matriks

```
J = A * 2
```



11	12	13	14
15	16	17	18
19	20	21	22
23	24	25	26

22	24	26	28
30	32	34	36
38	40	42	44
46	48	50	52

Operasi Matriks

```
K = A / 2
```

11	12	13	14
15	16	17	18
19	20	21	22
23	24	25	26

5.5	6.	6.5	7.
7.5	8.	8.5	9.
9.5	10.	10.5	11.
11.5	12.	12.5	13.

Operasi Matriks

```
# Element-wise product
L = A * E
```

11	12	13	14
15	16	17	18
19	20	21	22
23	24	25	26

13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28

143	168	195	224
255	288	323	360
399	440	483	528
575	624	675	728

Operasi Matriks

```
# Matrix product
M = A @ E
```

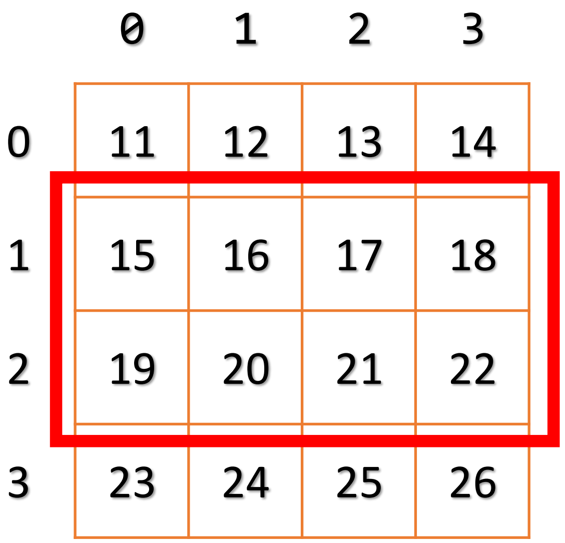
11	12	13	14
15	16	17	18
19	20	21	22
23	24	25	26

13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28

970	1020	1070	1120
1274	1340	1406	1472
1578	1660	1742	1824
1882	1980	2078	2176

Memotong Matriks

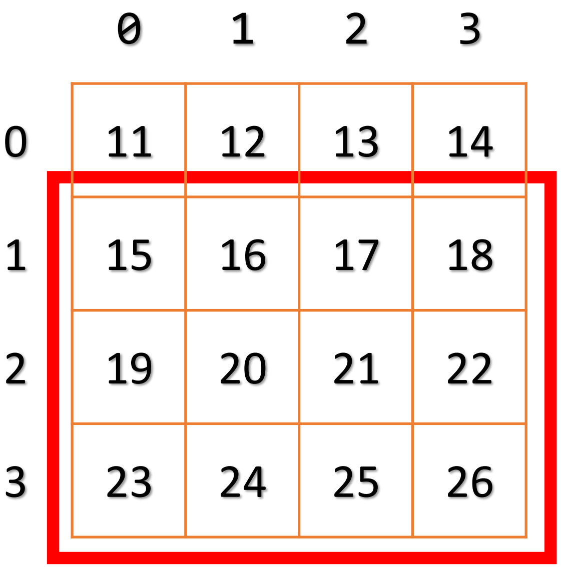
```
N = A[1:3, :]
N = A[1:3]

print(N)
# [[15 16 17 18]
#  [19 20 21 22]]
```



	0	1	2	3
0	11	12	13	14
1	15	16	17	18
2	19	20	21	22
3	23	24	25	26

Memotong Matriks

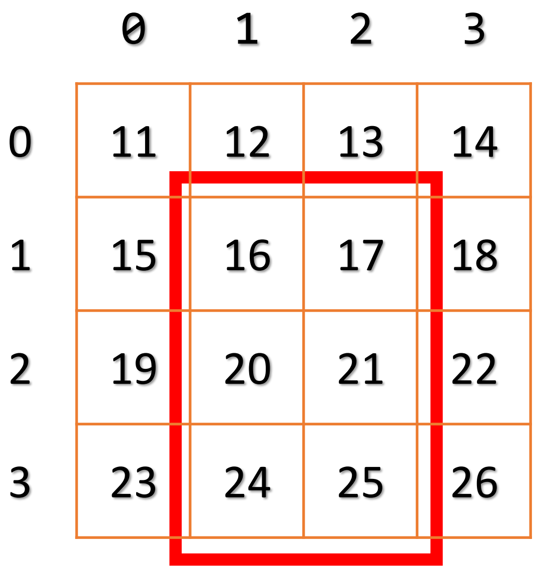
```
O = A[1:]

print(O)
# [[15 16 17 18]
#  [19 20 21 22]
#  [23 24 25 26]]
```



	0	1	2	3
0	11	12	13	14
1	15	16	17	18
2	19	20	21	22
3	23	24	25	26

Memotong Matriks

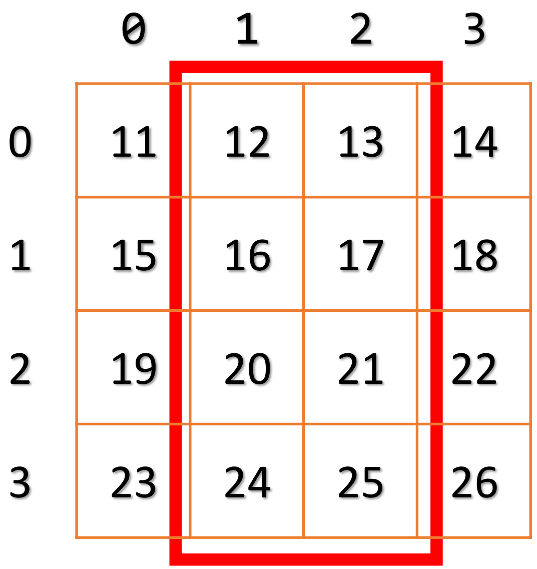
```
P = A[1:, 1:3]

print(P)
# [[16 17]
#  [20 21]
#  [24 25]]
```



	0	1	2	3
0	11	12	13	14
1	15	16	17	18
2	19	20	21	22
3	23	24	25	26

Memotong Matriks

```
Q = A[:, 1:3]

print(Q)
# [[16 17]
#  [20 21]
#  [24 25]]
```

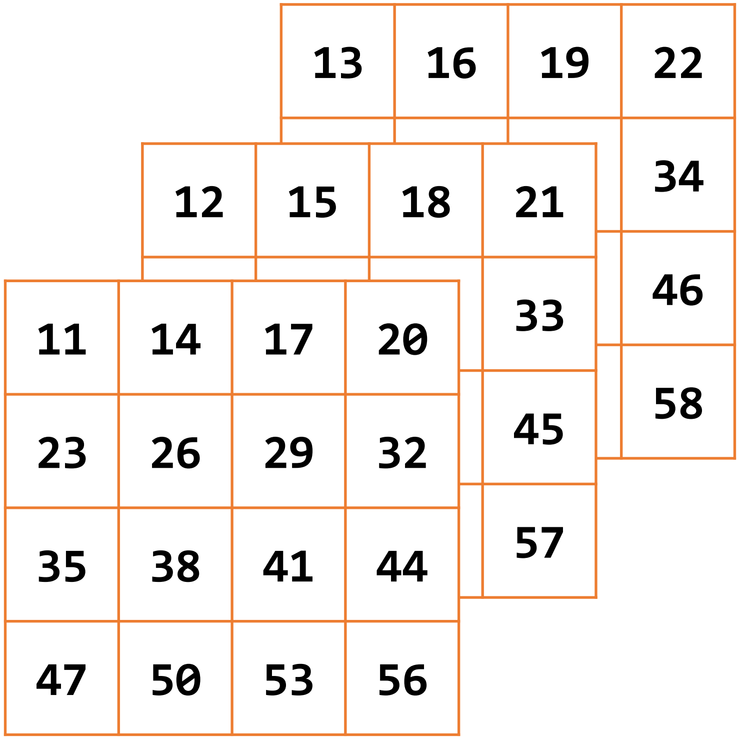


	0	1	2	3
0	11	12	13	14
1	15	16	17	18
2	19	20	21	22
3	23	24	25	26

Array 3-D

Membuat Array 3-D

```
A = np.arange(11, 59).reshape((4, 4, 3))
```



13	16	19	22
			34
			46
			58

12	15	18	21
			33
			45
			57

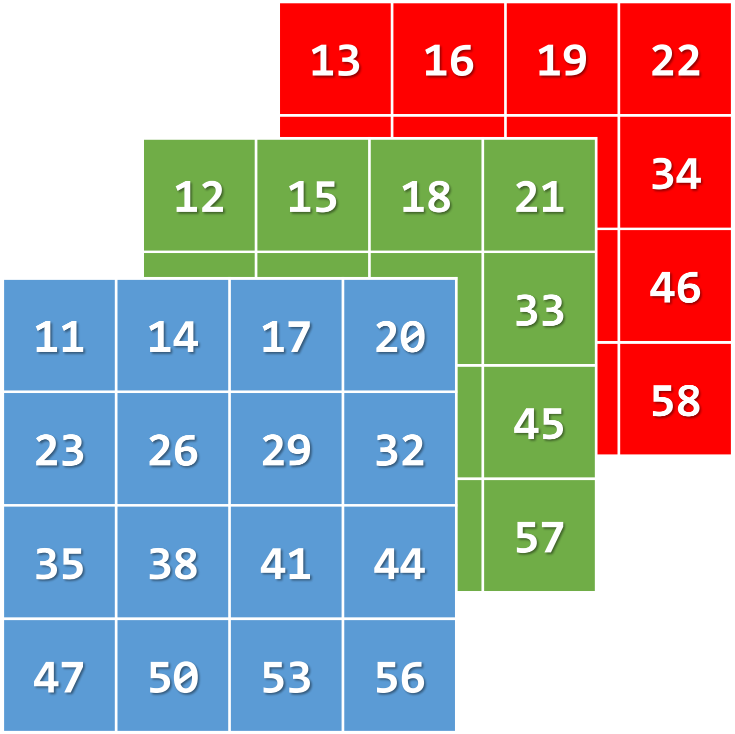
11	14	17	20
23	26	29	32
35	38	41	44
47	50	53	56

Dimensi Array 3-D

```
print(A.ndim)
# 3


print(A.shape)
# (4, 4, 3)

h, w = A.shape[:2]


print(A.size)
# 48
```

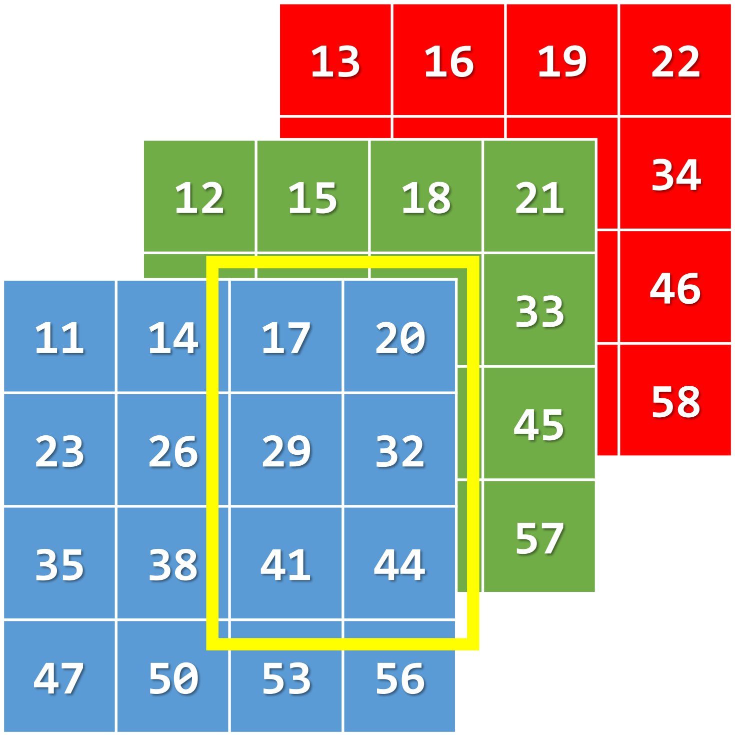
Color Channel

```
B = A[:, :, 0]
B = A[..., 0]

G = A[:, :, 1]
G = A[..., 1]

R = A[:, :, 2]
R = A[..., 2]
```



Cropping

```
crop = A[:3, 2:, 0]
```



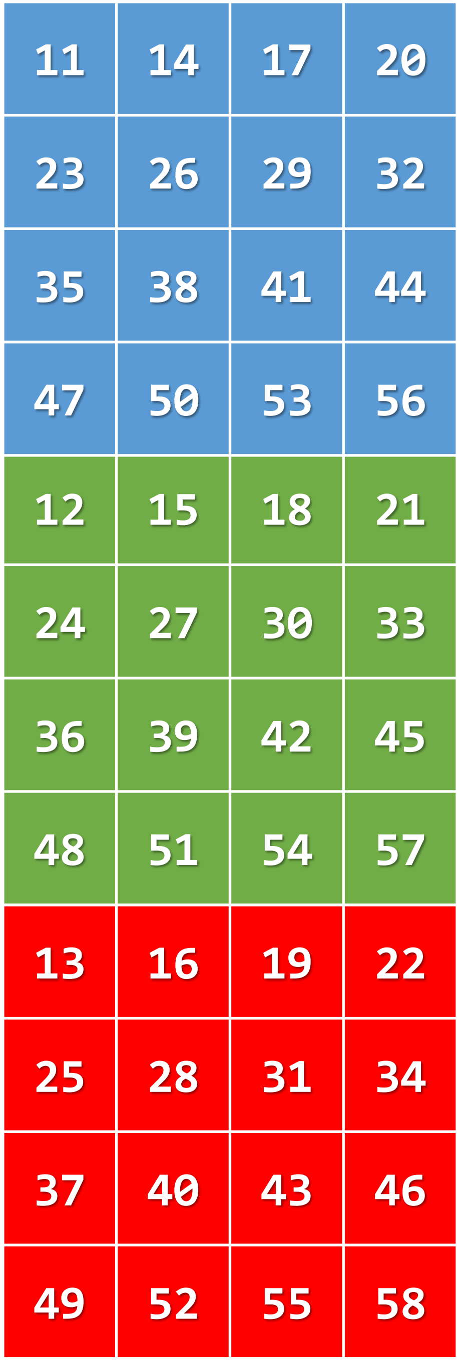
Stack

```
stack = np.hstack((B, G, R))
```

11	14	17	20	12	15	18	21	13	16	19	22
23	26	29	32	24	27	30	33	25	28	31	34
35	38	41	44	36	39	42	45	37	40	43	46
47	50	53	56	48	51	54	57	49	52	55	58

Stack

```
stack = np.hstack((B, G, R))
```



11	14	17	20
23	26	29	32
35	38	41	44
47	50	53	56
12	15	18	21
24	27	30	33
36	39	42	45
48	51	54	57
13	16	19	22
25	28	31	34
37	40	43	46
49	52	55	58

Thresholding

```
th = 40

print(B > th)
# [[False False False False]
#  [False False False False]
#  [False False  True  True]
#  [ True  True  True  True]]
```

11	14	17	20
23	26	29	32
35	38	41	44
47	50	53	56

Thresholding

```
B[B > th] = 255
```

11	14	17	20
23	26	29	32
35	38	255	255
255	255	255	255

Thresholding

```
B[B > th] = 255
B[B <= th] = 0
```

0	0	0	0
0	0	0	0
0	0	255	255
255	255	255	255

Thresholding

```
B = np.where(B > th, 255, 0)
```

0	0	0	0
0	0	0	0
0	0	255	255
255	255	255	255

Brightness

```
h, w = G.shape
add = 220

for y in range(h):
    for x in range(w):
        if G[y, x] + add <= 255:
            G[y, x] += add
        else:
            G[y, x] = 255
```

12	15	18	21
24	27	30	33
36	39	42	45
48	51	54	57

232	235	238	241
244	247	250	253
255	255	255	255
255	255	255	255

Brightness

```
h, w = G.shape
add = 220

for y in range(h):
    for x in range(w):
        G[y, x] = G[y, x] + add if G[y, x] + add <= 255
else 255
```

12	15	18	21
24	27	30	33
36	39	42	45
48	51	54	57

232	235	238	241
244	247	250	253
255	255	255	255
255	255	255	255

Brightness

```
h, w = G.shape
add = 220

G[G > 255 - add] = 255
G[G <= 255 - add] += add
```

12	15	18	21
24	27	30	33
36	39	42	45
48	51	54	57

232	235	238	241
244	247	250	253
255	255	255	255
255	255	255	255

Brightness

```
h, w = G.shape
add = 220

G[G > 255 - add] = 255
G[G <= 255 - add] += add
```

12	15	18	21
24	27	30	33
36	39	42	45
48	51	54	57

232	235	238	241
244	247	250	253
255	255	255	255
255	255	255	255

Brightness

```
h, w = G.shape
add = 220

gray = np.where(gray <= 255 - add, gray + add, 255)
```

12	15	18	21
24	27	30	33
36	39	42	45
48	51	54	57

232	235	238	241
244	247	250	253
255	255	255	255
255	255	255	255

Alhamdulillah.