



OpenCV

주요 클래스

수학과 오서영

1. 기본 자료형 클래스

Point 클래스

- : 2차원 평면 위에 있는 점의 좌표를 표현
- 멤버변수 : 2차원 좌표를 나타내는 x와 y

코드 3-1 간략화한 Point_ 클래스 정의와 이름 재정의

```
01 template<typename Tp> class Point
02 {
03 public:
04     Point();
05     Point(Tp _x, _Tp y);
06     Point(const Point& pt);
```

```
08     Point& operator = (const Point_& pt);
09
10     Tp dot(const Point& pt) const;
11     double ddot(const Point& pt) const;
12     double cross(const Point& pt) const;
13     bool inside(const Rect_<_Tp>& r) const;
14     ...
```

```
16     Tp x, y;
17 };
18
19 typedef Point<int>    Point2i;
20 typedef Point<int64>  Point2l;
21 typedef Point<float>  Point2f;
22 typedef Point_<double> Point2d;
23 typedef Point2i       Point;
```

내적(dot product)을 계산하여 반환
내적을 실수형으로 계산하여 double 자료형으로 반환
외적(cross product)을 반환
점의 좌표가 사각형 r 영역 안에 있으면 true를 반환

1. 기본 자료형 클래스

Point 클래스

-> 2차원 정수 좌표계에서 좌표를 표현하는 자료형

- Point2i 클래스를 사용 : 정수형 int 자료형으로 점의 좌표를 표현
- Point2f 클래스를 사용 : float 자료형

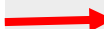
```
Point pt1;           // pt1 = (0, 0)
pt1.x = 5; pt1.y = 10; // pt1 = (5, 10)
Point pt2(10, 30);    // pt2 = (10, 30)
```

Point 연산자를 이용한 좌표연산

```
// pt1 = [5, 10], pt2 = [10, 30]
Point pt3 = pt1 + pt2;    // pt3 = [15, 40]
Point pt4 = pt1 * 2;      // pt4 = [10, 20]
int d1 = pt1.dot(pt2);    // d1 = 350
bool b1 = (pt1 == pt2);   // b1 = false
```

Point 객체 좌표 출력

```
cout << "pt1: " << pt1 << endl;
cout << "pt2: " << pt2 << endl;
```



```
pt1: [5, 10]
pt2: [10, 30]
```

1. 기본 자료형 클래스

Size 클래스

: 영상 또는 사각형 영역의 크기를 표현

- 멤버변수 : 사각형 영역의 가로와 세로 크기를 나타내는 width와 height

코드 3-2 간략화한 Size_ 클래스 정의와 이름 재정의

```
01 template<typename Tp> class Size
02 {
03 public:
04     Size();
05     Size(Tp _width, _Tp height);
06     Size(const Size& sz);
```

```
08     Size& operator = (const Size_& sz);
09
10     _Tp area() const;
11     bool empty() const;
12
13     Tp width, height;
14 };
```

사각형 크기에 해당하는 면적(width×height)을 반환
유효하지 않은 크기이면 true를 반환

1. 기본 자료형 클래스

Size 클래스

-> 2차원 정수 좌표계에서 좌표를 표현하는 자료형

- Size2i 클래스를 사용 : 사각형 영역의 가로 및 세로 크기를 int 자료형으로 표현
- float 자료형을 사용 : Size2f 클래스를 사용

```
Size sz1, sz2(10, 20);      // sz1 = [0 x 0], sz2 = [10 x 20]
sz1.width = 5; sz1.height = 10; // sz1 = [5 x 10]
```

사칙연산으로 크기조절

```
// sz1 = [5 x 10], sz2 = [10 x 20]
Size sz3 = sz1 + sz2;  // sz3 = [15 x 30]
Size sz4 = sz1 * 2;    // sz4 = [10 x 20]
int area1 = sz4.area(); // area1 = 200
```

Size 객체 출력

```
cout << "sz3: " << sz3 << endl;
cout << "sz4: " << sz4 << endl;
```

sz3: [15 x 30]
sz4: [10 x 20]

1. 기본 자료형 클래스

Rect 클래스

: 사각형의 위치와 크기 정보를 표현

- 멤버변수 : 사각형의 좌측 상단 점의 좌표를 나타내는 x, y
사각형의 가로 및 세로 크기를 나타내는 width, height

코드 3-3 간략화한 Rect_ 클래스 정의와 이름 재정의

```
12 Point<Tp> tl() const;  
13 Point<Tp> br() const;  
14 Size<_Tp> size() const;  
15 Tp area() const;  
16 bool empty() const;  
17 bool contains(const Point<_Tp>& pt) const;
```

Rect::**tl()** : 사각형의 좌측 상단 점의 좌표를 반환
Rect::**br()** : 사각형의 우측 하단 점의 좌표를 반환
Rect::**size()** : 사각형의 크기 정보를 반환
Rect::**area()** : 사각형의 면적(width×height)을 반환
Rect::**empty()** : 유효하지 않은 사각형이면 true를 반환
Rect::**contains()** : 인자로 전달된 pt 점이
사각형 내부에 있으면 true를 반환

1. 기본 자료형 클래스

Rect 클래스

-> 2차원 정수형 좌표계에서의 사각형 정보를 표현

- Size2i 클래스를 사용 : 사각형 영역의 가로 및 세로 크기를 int 자료형으로 표현
- float 자료형을 사용 : Size2f 클래스를 사용

```
Rect rc1;                // rc1 = [0 x 0 from (0, 0)]
Rect rc2(10, 10, 60, 40); // rc2 = [60 x 40 from (10, 10)]
```

객체의 크기 및 위치를 변경하는 코드

```
// rc1 = [0 x 0 from (0, 0)], rc2 = [60 x 40 from (10, 10)]
Rect rc3 = rc1 + Size(50, 40); // rc3 = [50 x 40 from (0, 0)]
Rect rc4 = rc2 + Point(10, 10); // rc4 = [60 x 40 from (20, 20)]
```

사각형의 위치가 (10, 10)만큼 이동

1. 기본 자료형 클래스

RotatedRect 클래스

: 회전된 사각형을 표현하는 클래스

- 멤버변수 : 회전된 사각형의 중심 좌표를 나타내는 center
사각형의 가로 및 세로 크기를 나타내는 size
회전 각도 정보를 나타내는 angle

코드 3-4 간략화한 RotatedRect 클래스 정의

```
01 class RotatedRect
02 {
03 public:
04     RotatedRect();
05     RotatedRect(const Point2f& _center, const Size2f& _siz
06     RotatedRect(const Point2f& point1, const Point2f& po
07
08     void points(Point2f pts[]) const;
09     Rect boundingRect() const;
10     Rect<float> boundingRect2f() const;
```

회전된 사각형은 네 꼭지점 좌표를 pts 인자에 저장
회전된 사각형을 포함하는 최소 크기의 사각형 정보를 반환 (정수)
회전된 사각형을 포함하는 최소 크기의 사각형 정보를 반환(실수)

1. 기본 자료형 클래스

RotatedRect 클래스

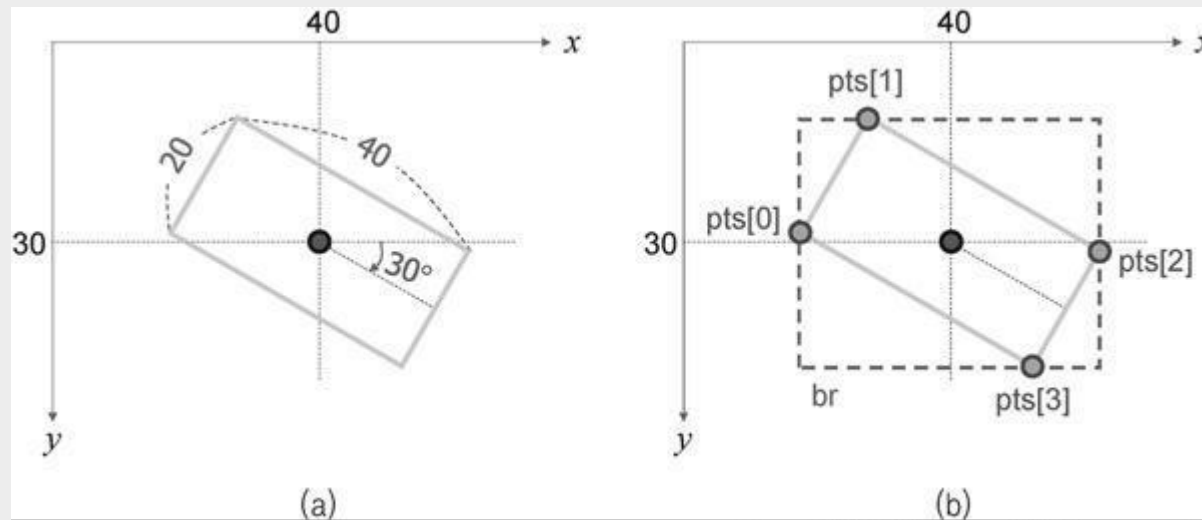
-> 모든 정보를 float 자료형을 사용하여 표현

중심 좌표가 (40, 30), 크기는 40×20, 시계 방향으로 30°만큼 회전된 사각형 객체

```
RotatedRect rr1(Point2f(40, 30), Size2f(40, 20), 30.f);
```

사각형의 네 꼭지점 좌표가 pts 배열에 저장

```
Point2f pts[4];  
rr1.points(pts);
```



1. 기본 자료형 클래스

Range 클래스

: 범위 또는 구간을 표현하는 클래스

- 멤버변수 : 범위의 시작과 끝을 나타내는 start와 end

코드 3-5 간략화한 Range 클래스 정의

```
01  class Range
02  {
03  public:
04      Range();
05      Range(int _start, int _end);
06
07      int size() const;
08      bool empty() const;
09      static Range all();
10
11      int start, end;
12  };
```

범위 크기(end - start)를 반환
start와 end가 같으면 true를 반환
start = INT_MIN, end = INT_MAX로 설정한
Range 객체를 반환

1. 기본 자료형 클래스

String 클래스

: 문자열을 저장하고 처리

```
String str1 = "Hello";  
String str2 = "world";  
String str3 = str1 + " " + str2; // str3 = "Hello world"
```

객체의 내용을 비교

```
bool ret = (str2 == "WORLD");
```

format() 함수

```
Mat imgs[3];  
for (int i = 0; i < 3; i++) {  
    String filename = format("test%02d.bmp", i + 1);  
    imgs[i] = imread(filename);  
}
```

2. Mat 클래스

Mat 클래스

- 일반적인 2차원 행렬뿐만 아니라 고차원 행렬을 표현할 수 있으며, 한 개 이상의 채널(channel)을 가질 수 있다.
- 정수, 실수, 복소수 등으로 구성된 행렬 또는 벡터(vector)를 저장할 수 있고, 그레이스케일 또는 컬러 영상을 저장가능

행렬의 생성과 초기화

```
Mat img1;      Mat img2(480, 640, CV_8UC1);  // unsigned char, 1-channel
                Mat img3(480, 640, CV_8UC3);  // unsigned char, 3-channels
```

```
Mat::Mat(int rows, int cols, int type);
```

- rows 새로 만들 행렬의 행 개수(영상의 세로 크기)
- cols 새로 만들 행렬의 열 개수(영상의 가로 크기)
- type 새로 만들 행렬의 타입

```
#define CV_8U  0  // uchar, unsigned char
#define CV_8S  1  // schar, signed char
#define CV_16U 2  // ushort, unsigned short
#define CV_16S 3  // signed short
#define CV_32S 4  // int
#define CV_32F 5  // float
#define CV_64F 6  // double
#define CV_16F 7  // float16_t
```

2. Mat 클래스

Mat 클래스

- 일반적인 2차원 행렬뿐만 아니라 고차원 행렬을 표현할 수 있으며, 한 개 이상의 채널(channel)을 가질 수 있다.
- 정수, 실수, 복소수 등으로 구성된 행렬 또는 벡터(vector)를 저장할 수 있고, 그레이스케일 또는 컬러 영상을 저장가능

행렬의 생성과 초기화

```
Mat img4(Size(640, 480), CV_8UC3);    // Size(width, height)
```

```
Mat::Mat(Size size, int type);
```

- size 새로 만들 행렬의 크기. Size(cols, rows) 또는 Size(width, height)
- type 새로 만들 행렬의 타입

2. Mat 클래스

행렬의 복사

```
Mat img1 = imread("dog.bmp");
```

```
Mat img2 = img1; // 복사 생성자(얕은 복사)
```

```
Mat img3;
```

```
img3 = img1; // 대입 연산자(얕은 복사)
```

```
Mat img4 = img1.clone(); // 깊은 복사
```

```
Mat img5;
```

```
img1.copyTo(img5); // 깊은 복사
```

Mat::clone()

: 자기 자신과 동일한 Mat 객체를 새로 만들어서 반환

Mat::copyTo()

: 인자로 전달된 m 행렬에 자기 자신을 복사

img1 영상의 모든 픽셀을 **Scalar(0, 255, 255)**에 해당하는 노란색으로 설정

```
img1.setTo(Scalar(0, 255, 255)); // yellow
```

파이썬 실습

Func1 : imread, type, shape, imshow

```
import numpy as np
import cv2

def func1():
    img1 = cv2.imread('cat.bmp', cv2.IMREAD_GRAYSCALE)

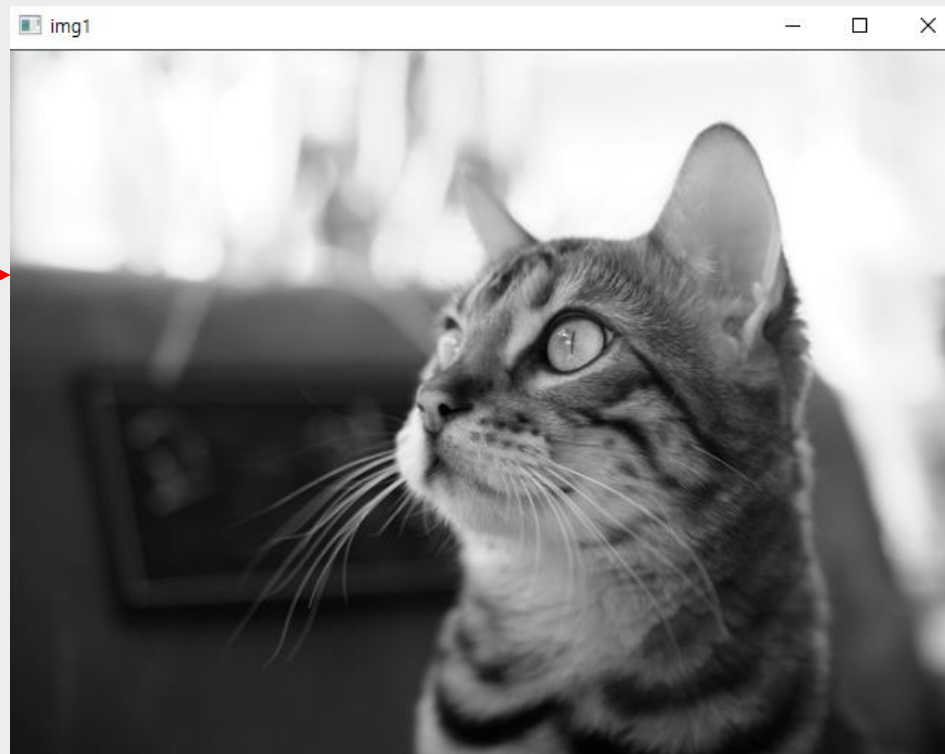
    if img1 is None:
        print('Image load failed!')
        return

    print('type(img1):', type(img1))
    print('img1.shape:', img1.shape)

    if len(img1.shape) == 2:
        print('img1 is a grayscale image')
    elif len(img1.shape) == 3:
        print('img1 is a truecolor image')

    cv2.imshow('img1', img1)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

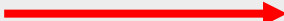
```
type(img1): <class 'numpy.ndarray'>
img1.shape: (480, 640)
img1 is a grayscale image
```



파이썬 실습

Func2 : empty, zeros, ones, full with numpy

```
def func2():  
    img1 = np.empty((480, 640), np.uint8)      # grayscale image  
    img2 = np.zeros((480, 640, 3), np.uint8)    # color image  
    img3 = np.ones((480, 640), np.int32)        # 1's matrix  
    img4 = np.full((480, 640), 0, np.float32)   # Fill with 0.0  
  
    mat1 = np.array([[11, 12, 13, 14],  
                    [21, 22, 23, 24],  
                    [31, 32, 33, 34]]).astype(np.uint8)  
  
    mat1[0, 1] = 100    # element at x=1, y=0  
    mat1[2, :] = 200  
  
    print(mat1)
```

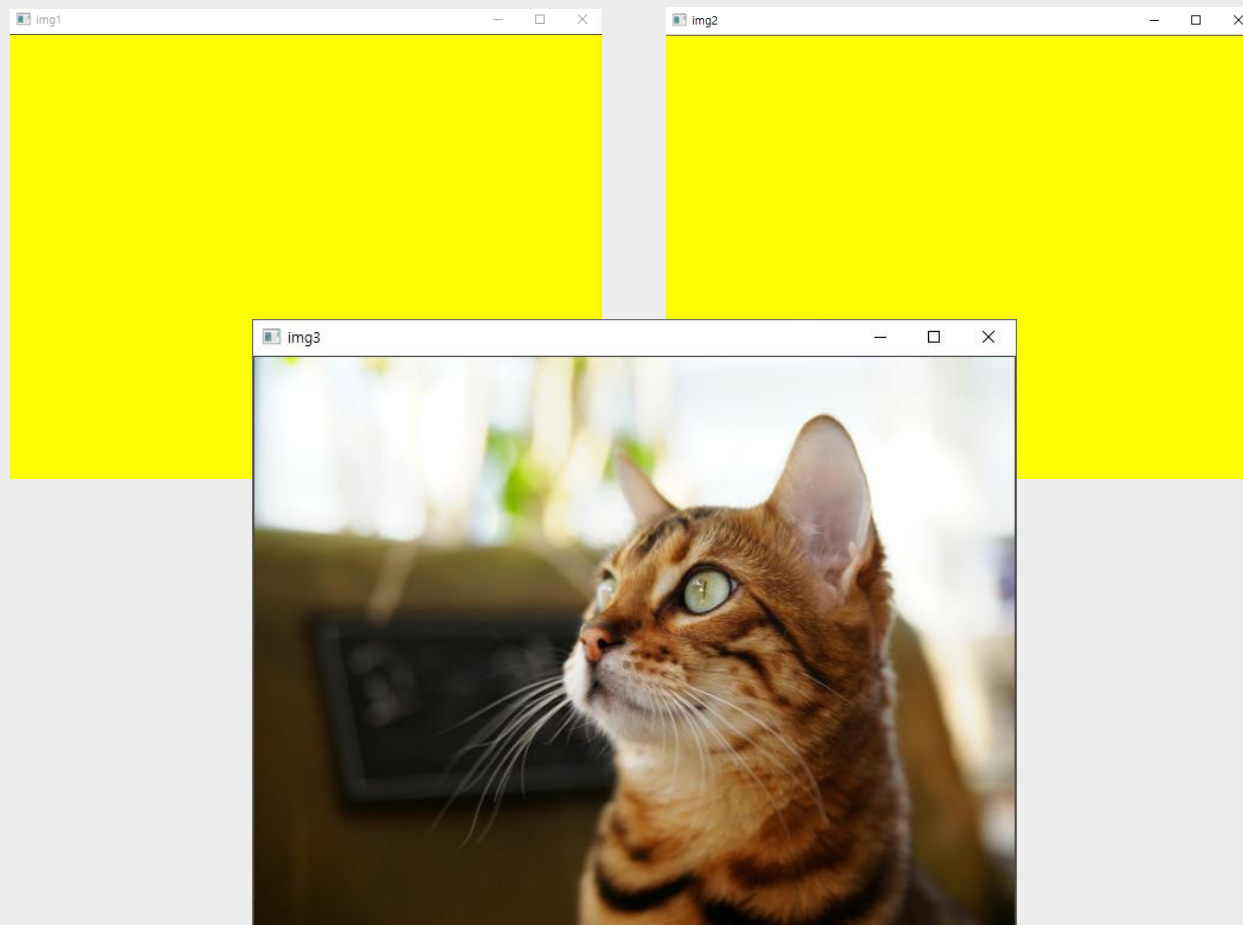
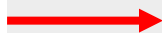


```
[[ 11 100  13  14]  
 [ 21  22  23  24]  
 [200 200 200 200]]
```


파이썬 실습

Func3 : 행렬의 복사 – 얇은복사, 깊은복사

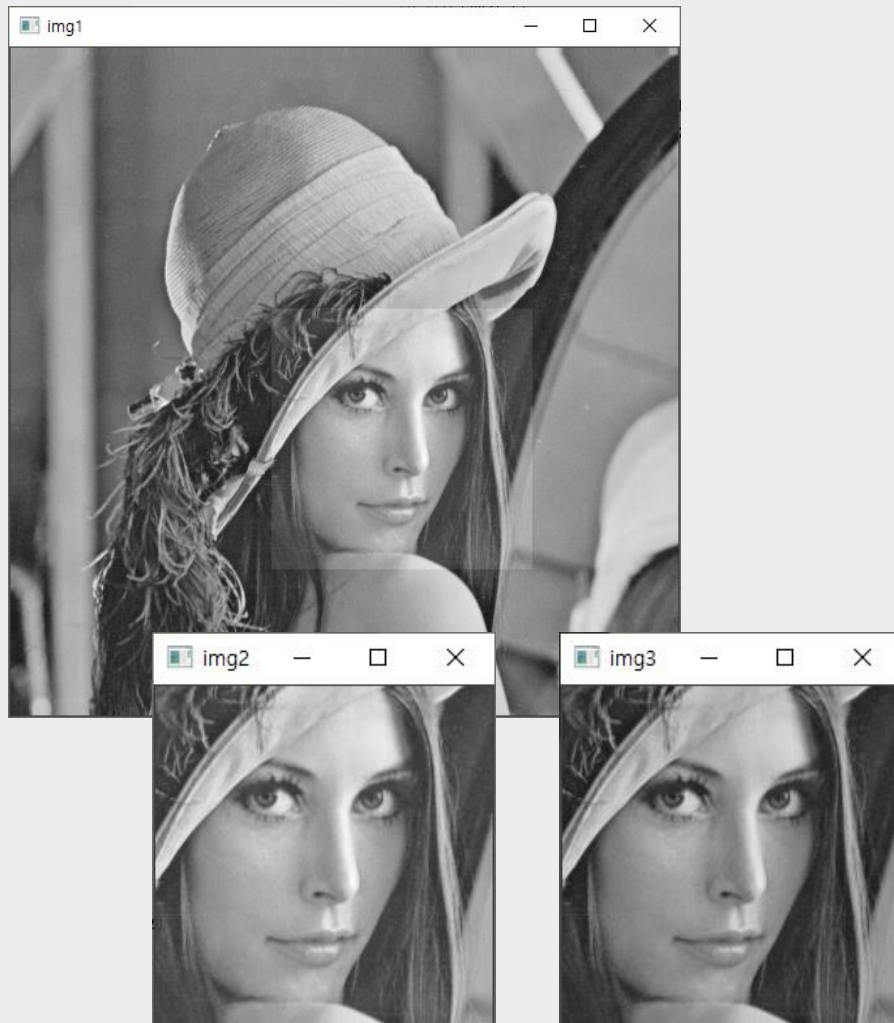
```
def func3():  
    img1 = cv2.imread('cat.bmp')  
  
    img2 = img1  
    img3 = img1.copy()  
  
    img1[:, :] = (0, 255, 255) # yellow  
  
    cv2.imshow('img1', img1)  
    cv2.imshow('img2', img2)  
    cv2.imshow('img3', img3)  
    cv2.waitKey()  
    cv2.destroyAllWindows()
```



파이썬 실습

Func4 : 행렬의 복사 – 얇은복사, 깊은복사

```
def func4():  
    img1 = cv2.imread('lenna.bmp', cv2.IMREAD_GRAYSCALE)  
  
    img2 = img1[200:400, 200:400]  
    img3 = img1[200:400, 200:400].copy()  
  
    img2 += 20  
  
    cv2.imshow('img1', img1)  
    cv2.imshow('img2', img2)  
    cv2.imshow('img3', img3)  
    cv2.waitKey()  
    cv2.destroyAllWindows()
```



파이썬 실습

Func5 : mat with numpy

```
def func5():  
    mat1 = np.array(np.arange(12)).reshape(3, 4)  
  
    print('mat1:')  
    print(mat1)  
  
    h, w = mat1.shape[:2]  
  
    mat2 = np.zeros(mat1.shape, type(mat1))  
  
    for j in range(h):  
        for i in range(w):  
            mat2[j, i] = mat1[j, i] + 10  
  
    print('mat2:')  
    print(mat2)
```

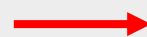


```
mat1:  
[[ 0  1  2  3]  
 [ 4  5  6  7]  
 [ 8  9 10 11]]  
mat2:  
[[10 11 12 13]  
 [14 15 16 17]  
 [18 19 20 21]]
```

파이썬 실습

Func6 : mat with numpy

```
def func6():  
    mat1 = np.ones((3, 4), np.int32)    # 1's matrix  
    mat2 = np.arange(12).reshape(3, 4)  
    mat3 = mat1 + mat2  
    mat4 = mat2 * 2  
  
    print("mat1:", mat1, sep='#n')  
    print("mat2:", mat2, sep='#n')  
    print("mat3:", mat3, sep='#n')  
    print("mat4:", mat4, sep='#n')
```



```
mat1:  
[[1 1 1 1]  
 [1 1 1 1]  
 [1 1 1 1]]  
mat2:  
[[ 0  1  2  3]  
 [ 4  5  6  7]  
 [ 8  9 10 11]]  
mat3:  
[[ 1  2  3  4]  
 [ 5  6  7  8]  
 [ 9 10 11 12]]  
mat4:  
[[ 0  2  4  6]  
 [ 8 10 12 14]  
 [16 18 20 22]]
```