CHAPTER 03
파이썬 기초 실습3

김 찬, 윤 영 선

ckim.esw@gmail.com, ysyun@hnu.kr

정보통신공학과

3. 파이썬의 자료구조

3.3 numpy

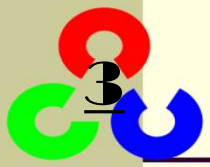
◆ 행렬 원소로 지정값 생성하기

```
# example1
import numpy as np

zeros = np.zeros((2, 5), np.int32)      # 0으로 된 2행 5열 행렬 생성
ones = np.ones((3, 1), np.uint8)        # 1로 된 3행 1열 행렬 생성
emptys = np.empty((1, 5), np.float64)   # 비어있는 1행 5열 행렬 생성
fulls = np.full(5, 15, np.float32)      # 15로 된 1차원 행렬 생성

print(zeros)
print(ones)
print(emptys)
print(fulls)
```

```
[[0 0 0 0 0]
 [0 0 0 0 0]]
[[1]
 [1]
 [1]]
[[3.9155e-313 1.8399e+222 1.6758e+243 8.8241e+199 4.3385e-313]]
[15. 15. 15. 15. 15.]
```



3. 파이썬의 자료구조

3.3 numpy

◆ 행렬 원소로 임의값 생성하기

```
# example2
import numpy as np

np.random.seed(10)                          # 랜덤시드 10으로 설정
a = np.random.rand(2, 3)                    # 2행 3열로 된 랜덤 행렬 생성
b = np.random.randint(1, 100, 6)            # 1 ~ 100 랜덤 행렬 6개 생성
print(a)
print(b)
b = b.reshape(2,-1)                         # 1차원 행렬을 2행으로 변경
print(b)
```

```
[[0.77132064 0.02075195 0.63364823]
 [0.74880388 0.49850701 0.22479665]]
[ 9 74  1 41 37 17]
[[ 9 74  1]
 [41 37 17]]
```

3. 파이썬의 자료구조

3.3 numpy 실습1

◆조건

- 입력
  - list1 = np.array([1, 2, 3, 4, 5, 6, 7, 8])
  - list2 = np.array([10, 20, 30, 40, 50, 60, 70, 80])

- 조건
  - 위 두개의 배열을 (2, 4) 형태로 바꾸고, 두 배열의 합과 곱을 출력하라.

- 출력 예시

```
[[11 22 33 44]
 [55 66 77 88]]
[[ 10  40  90 160]
 [250 360 490 640]]
```

3. 파이썬의 자료구조

3.3 numpy 실습1

- 1. 위 두개의 배열을 (2, 4) 형태로 바꾸기.
- 2. 두 배열의 합과 곱을 출력하기.

```
list1 = np.array([1, 2, 3, 4, 5, 6, 7, 8])
list2 = np.array([10, 20, 30, 40, 50, 60, 70, 80])
```

```

```

```
print(          )
print(          )
```

- *출력 예시*

```
[[11 22 33 44]
 [55 66 77 88]]
[[ 10  40  90 160]
 [250 360 490 640]]
```

3. 파이썬의 자료구조

3.3 numpy 실습2

◆ 조건

- 입력
  - 자유

- 조건
  - 실수형 원소 10개를 갖는 ndarray 행렬을 랜덤으로 선언해서 전체 원소와 합과 평균을 구하라.

- 출력 예시

```
[[0.77132064 0.02075195 0.63364823 0.74880388 0.49850701 0.22479665
  0.19806286 0.76053071 0.16911084 0.08833981]]

4.113872595619601

0.41138725956196015
```

3. 파이썬의 자료구조

■ 3.3 numpy 실습2

◆ numpy + for

```
# solution1
import numpy as np

np.random.seed(10)
np_a = np.random.rand(10)
sum_a = 0
for a in np_a:
    sum_a += a
print(np_a)
print(sum_a)
print(sum_a/10)
```

```
[[0.77132064 0.02075195 0.63364823 0.74880388 0.49850701 0.22479665
  0.19806286 0.76053071 0.16911084 0.08833981]]

4.113872595619601

0.41138725956196015
```

◆ numpy + 내장함수

```
# solution2
import numpy as np

np.random.seed(10)
np_a = np.random.rand(1, 10)
print(np_a)
print(np_a.sum())
print(np_a.mean())
```

3. 파이썬의 자료구조

3.3 numpy 실습3

◆ 조건

- 입력
  - 자유
- 조건
  - 0~50 사이의 임의의 원소(정수형, 중복가능)를 500개 만들어서 가장 많이 나온 **원소값**과 **중복횟수**로 출력하라.
- 출력 예시

```
가장 많이 나온 원소: 102
중복횟수: 4회
```

3. 파이썬의 자료구조

3.3 numpy 실습3

◆ 가장 많이 중복 된 원소값 + 중복횟수 출력

```
import numpy as np

np.random.seed(10)
result = {}
arr = np.random.randint(0, 50, 500)

# 딕셔너리에 중복값을 담는 과정
```

3. 파이썬의 자료구조

3.3 numpy 실습3

◆ 가장 많이 중복 된 원소값 + 중복횟수 출력

- **For-loop로 가장 많이 나온 원소 값 찾기**

```
# solution1
## 중복이 가장 많은 값만 변수에 저장
max_key, max_value = 0, 0
for k, v in result.items():
    if v > max_value:
        max_value = v
        max_key = k

print(f'가장 많이 나온 원소: {max_key}')
print(f'중복횟수: {max_value}회')
```

- ***출력 예시***

```
가장 많이 나온 원소: 102
중복횟수: 4회
```

3. 파이썬의 자료구조

3.3 numpy 실습3

◆ 가장 많이 중복 된 원소값 + 중복횟수 출력

- 익명함수(람다, lambda)

```
# solution2
## 중복이 많은 횟수대로 정렬
result = sorted(result.items(), key=lambda x:x[1], reverse=True)
print(result)

print(f'가장 많이 나온 원소: {result[0][0]}')
print(f'중복횟수: {result[0][1]}회')
```

- *출력 예시*

```
가장 많이 나온 원소: 102
중복횟수: 4회
```

- 2번째, 3번째로 많이 중복 된 원소값도 찾기 쉬워짐

CHAPTER 04
인터페이스 기초

김 찬, 윤 영 선

ckim.esw@gmail.com, ysyun@hnu.kr

정보통신공학과

4. 인터페이스 기초

4.1 윈도우 제어

◆4.1.1 윈도우 생성

```
import numpy as np
import cv2

image = np.zeros((200, 400), np.uint8) # 0으로 된 200 x 400 행렬 생성
print(image)

title1 = 'Position1'                   # Window 이름 변수
cv2.namedWindow(title1)                # Window 이름 지정
cv2.imshow(title1, image)              # Window 이름과 이미지 출력
cv2.waitKey(0)                         # 키 입력 대기
cv2.destroyAllWindows()                # 실행 되어있는 창 모두 닫기
```

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

RGB 색상에서 0은 검은색을 의미, 255는 흰색을 의미한다.

4. 인터페이스 기초

4.1 윈도우 제어

◆4.1.2 윈도우 색상 변경

```
import numpy as np
import cv2

image = np.zeros((200, 400), np.uint8) # 0으로 된 200 x 400 행렬 생성
image[:] = 200                         # 변경하고 싶은 색 RGB 값 입력
print(image)

title1 = 'Changed'                     # Window 이름 변수
cv2.namedWindow(title1)                # Window 이름 지정
cv2.imshow(title1, image)              # Window 이름과 이미지 출력
cv2.waitKey(0)                         # 키 입력 대기
cv2.destroyAllWindows()                # 실행 되어있는 창 모두 닫기
```

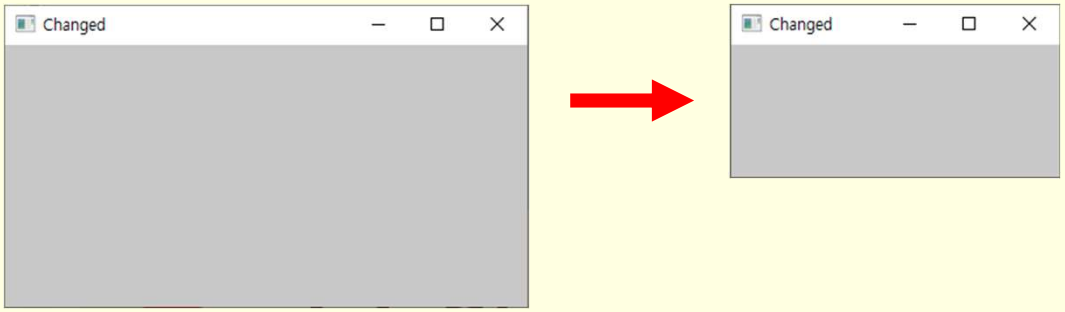
```
[[200 200 200 ... 200 200 200]
 [200 200 200 ... 200 200 200]
 [200 200 200 ... 200 200 200]
 ...
 [200 200 200 ... 200 200 200]
 [200 200 200 ... 200 200 200]
 [200 200 200 ... 200 200 200]]
```

RGB 색상에서 0은 검은색을 의미, 255는 흰색을 의미한다.

4. 인터페이스 기초

4.1 윈도우 제어

◆ 4.1.2 윈도우 크기 변경




```
import numpy as np
import cv2

image = np.zeros((200, 400), np.uint8) # 0으로 된 200 x 400 행렬 생성
image[:] = 200                         # 변경하고 싶은 색 RGB 값 입력

title1 = 'Changed'                     # Window 이름 변수
cv2.namedWindow(title1)                # Window 이름 지정
                                       # Window 이름과 변환할 사이즈 입력
cv2.imshow(title1, image)              # Window 이름과 이미지 출력
cv2.waitKey(0)                         # 키 입력 대기
cv2.destroyAllWindows()                # 실행 되어있는 창 모두 닫기
```

RGB 색상에서 0은 검은색을 의미, 255는 흰색을 의미한다.

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.1 키 이벤트 사용

```
import numpy as np
import cv2

image = np.zeros((200, 300), np.uint8)      # 0으로 된 200 x 300 행렬 생성
title1 = "Keyboard Event"                   # Window 이름 변수
cv2.namedWindow(title1)                     # Window 이름 지정
cv2.imshow(title1, image)                   # Window 이름과 이미지 출력

while True:                                 # 무한반복
    key = cv2.waitKeyEx(100)                # 100ms마다 키 입력 대기
    print(key)

cv2.destroyAllWindows()                     # 실행 되어있는 창 모두 닫기
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.1 키 이벤트 사용

```
switch_case = {                                  # 딕셔너리 형태로 키 이벤트 지정
    97: 'a키 입력',                               # ASCII 형태로 입력 받을 때
    ord('b'): 'b키 입력',                         # 98 == ASCII 형태의 'b'와 같다.
    0x41: 'A키 입력',                             # 16진수 형태로 A를 입력 받을 때
    int('0x42', 16): 'B키 입력',                  # 16진수로 받은 입력을 10진수로 변환
    2424832: "왼쪽 화살표 키 입력",
    2490368: "윗쪽 화살표 키 입력",
    2555904: "오른쪽 화살표 키 입력",
    2621440: "아래쪽 화살표 키 입력"
}
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.1 키 이벤트 사용

```
while True:                                 # 무한반복
    key = cv2.waitKeyEx(100)                # 100ms마다 키 입력 받아오기

    if key == 27:                           # esc 키 이벤트가 감지되면
        break                               # while문 탈출

    try:                                    # 일단 실행
        result = switch_case[key]           # 입력한 키가 딕셔너리에 있는 값이면
        print(result)                       # 어떤 키 입력인지 출력

    except KeyError:                        # 딕셔너리에 없는 값이라면
        result = -1                         # -1 반환 (미 입력 시 어차피 -1)
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

◆ 조건

- 입력
  - 자유

- 조건
  - 1. Apple을 키 입력으로 출력하라.
  - 2. 잘못 입력했다면 “Del” 키 이벤트를 통해 모두 지워라.
  - 2. 종료 버튼은 esc가 아닌 키보드에 있는 “End” 키 이벤트로 종료 하도록 변경하라.

- 출력 예시

```
a
ap
app
appl
apple
A
Ap
App
Appl
Apple
```

소문자로 apple을 입력해서
Del 버튼으로 초기화 시킨 후
다시 입력한 모습

Hint
- ord(str) = 문자를 아스키코드로 변경
- chr(int) = 아스키코드를 문자로 변경

4. 인터페이스 기초

4.2 키보드 이벤트 제어

```
result = ""                          # 입력한 키 저장할 변수 선언
while True:                          # 무한반복
    key = cv2.waitKeyEx(100) # 100ms마다 키 입력 받아오기

    # del 키 이벤트가 감지되면
    # while문 탈출
    # end 키 이벤트가 감지되면
    # 초기화

    # 일단 실행
    # 입력한 키를 result에 저장
    # 어떤 키 입력인지 출력
    # 아스키코드에 없는 값이라면
    # 무시

cv2.destroyAllWindows()      # 반복 문 탈출 시 모든 창 닫기
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.2 마우스 이벤트 사용

```
import numpy as np
import cv2

def onMouse(event, x, y, flags, param):      # onMouse 함수에서 제공하는 이벤트
    print(event, x, y, flags, param)         # 출력을 통해 어떤 값이 나오나 보기

image = np.full((200, 300), 255, np.uint8)  # 255로 채워진 200 x 300 행렬 생성

title1 = “Mouse Event”
Cv2.imshow(title1, image)
cv2.setMouseCallback(title1, onMouse)        # 마우스 이벤트 콜백 함수
Cv2.waitKey(0)
Cv2.destroyAllWindows()
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.2 마우스 이벤트 사용

```
def onMouse(event, x, y, flags, param):

# flags
1  = 왼쪽 버튼 누르기
2  = 오른쪽 버튼 누르기
4  = 중간 버튼 누르기
8  = [Ctrl]키 누르기
16 = [Shift]키 누르기
32 = [Alt]키 누르기
```

```
# event
0  = 마우스 움직임
1  = 왼쪽 버튼 클릭
2  = 오른쪽 버튼 클릭
3  = 휠 버튼 누르기
4  = 왼쪽 버튼 떼기
5  = 오른쪽 버튼 떼기
6  = 휠 버튼 떼기
7  = 왼쪽 버튼 더블클릭
8  = 오른쪽 버튼 더블클릭
9  = 중간 버튼 더블클릭
10 = 마우스 휠
11 = 마우스 가로 휠
```

조건:

- 마우스 왼쪽, 오른쪽, 휠 클릭에 대한 이벤트 출력
  - ex) if ~~:

        print("마우스 왼쪽 버튼 누르기")

```
마우스 왼쪽 버튼 누르기
마우스 오른쪽 버튼 누르기
마우스 오른쪽 버튼 떼기
```

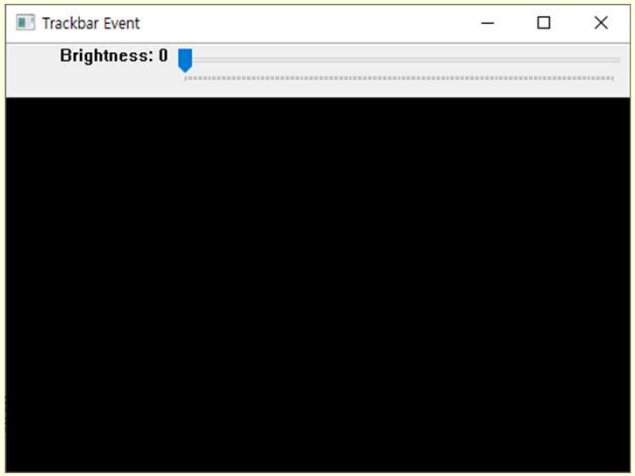
4. 인터페이스 기초

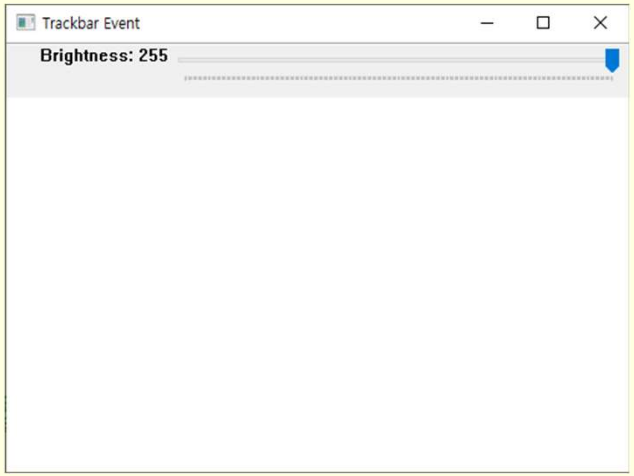
4.2 키보드 이벤트 제어

4.2.3 트랙바 이벤트 사용

```
cv2.createTrackbar(trackbarName, windowName, value, count, onChange)

# trackbarName: 트랙바 이름
# windowName: 트랙바를 생성할 창 이름
# value: 트랙바 위치 초기값
# count: 트랙바 최댓값
# onChange: 트랙바 위치가 변경될 때마다 호출할 콜백 함수
```





4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.3 트랙바 이벤트 사용

```
import numpy as np
import cv2

def onChange(value):
    global image, title
    [                    ]          # 트랙바 제어로 인해 변경되는 이미지 색상 적용
    cv2.imshow(title, image)

image = np.zeros((300, 500), np.uint8)
title = "Trackbar Event"

cv2.imshow(title, image)
cv2.createTrackbar("Brightness", title, image[0][0], 255, onChange)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4. 인터페이스 기초

4.2 키보드 이벤트 제어

4.2.3 트랙바 이벤트 사용

```
import numpy as np
import cv2

def onChange(value):
    global image, title
    image[:] = value
    cv2.imshow(title, image)

image = np.zeros((300, 500), np.uint8)
image[:] = 150                          # 현재 이미지 색상 지정
title = "Trackbar Event"

cv2.imshow(title, image)
cv2.createTrackbar("Brightness", title, image[0][0], 255, onChange)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

지정된 이미지 색상 가져와서
트랙바 위치 시작값으로 지정

4. 인터페이스 기초

4.3 그리기 함수

4.3.1 직선 및 사각형 그리기

```
import numpy as np
import cv2

blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)
image = np.zeros((400, 600, 3), np.uint8)
image[:] = (255, 255, 255)

pt1, pt2 = (50, 50), (250, 150)
pt3, pt4 = (400, 150), (500, 50)
roi = (50, 200, 200, 100)
```

4. 인터페이스 기초

4.3 그리기 함수

◆ 4.3.1 직선 및 사각형 그리기

```
# 사각형 그리기
cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)
cv2.rectangle(image, roi, red, 3, cv2.LINE_8)
cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)

# 직선 그리기
cv2.line(image, pt1, pt2, red)
cv2.line(image, pt3, pt4, cv2.LINE_AA)

cv2.imshow("Line & Rectangle", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```