K-Digital Training

codingOn

NumPy

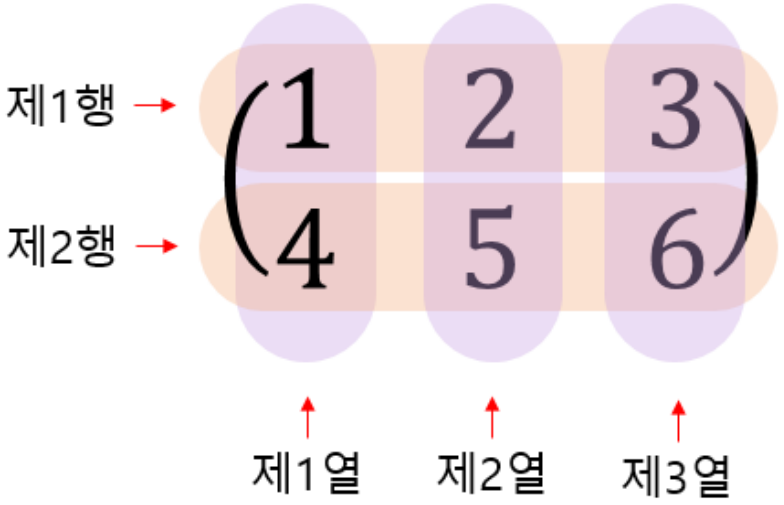
NumPy?

- NumPy는 Python에서 다차원 배열과 수치 계산을 효율적으로 처리하기 위한 라이브러리
- Numerical Python의 줄임말
- NumPy의 중심은 ndarray(n-dimensional array)라는 다차원 배열 객체
- 배열의 형태(shape), 크기(size), 데이터 타입(dtype)을 지정할 수 있음
- 크기가 다른 배열 간에도 연산이 가능하도록 자동으로 크기를 맞춤(브로드캐스팅)
- 공식홈페이지

NumPy?

- 언제 필요한가?
- 크롤링 데이터에서 수치 계산이 필요할 때
- 데이터 배열 연산이나 변환이 필요할 때
- 예)
  - 리스트를 Numpy 배열로 변환
  - 배열 연산 (예: 합계, 평균, 정규화)

행렬



행렬의 성분을
가로로 배열한 줄을 **행**
세로로 배열한 줄을 **열**

→ 2 x 3 행렬 또는 2행 3열의 행렬

NumPy

codingOn

- Ndarray 타입의 배열을 만들 수 있음. 파이썬의 리스트와 다름

[Ndarray 타입이란?]

N-dimension array 의 약자. 다차원 배열을 의미

주의) 배열은 동일한 자료형의 요소로 구성되어야함

```
import numpy as np #numpy 라이브러리 import
x = np.array([3, 1, 2]) #np.array : ndarray 타입의 배열 생성
print(x) # x 성분 출력
type(x) # x의 type 확인
```

```
[3 1 2]
```

```
numpy.ndarray
```

NumPy

codingOn

- 배열생성

```
import numpy as np

# 1차원 배열 생성
array_1d = np.array([1, 2, 3, 4, 5])
print("1차원 배열:", array_1d)
# 2차원 배열 생성
array_2d = np.array([[1, 2, 3], [4, 5, 6]])
print("2차원 배열:\n", array_2d)
# 3차원 배열 생성
array_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print("3차원 배열:\n", array_3d)
```

배열 속성

- shape : 배열의 크기(모양) 반환. 튜플형태
- ndim : 배열의 차원을 반환
- dtype : 배열의 각 원소의 자료형을 반환
- itemsize : 배열의 각 원소 하나의 크기(바이트 단위)
- size : 배열의 전체 원소 개수를 반환

NumPy

codingOn

- 배열접근

```
arr = np.array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])
print(arr[0, 1])  # 2
print(arr[2, 2])  # 9
rows = [0, 2]
cols = [1, 2]
print(arr[rows, cols]) # [2 9]

arr = np.array([10, 20, 30, 40, 50])
print(arr[arr > 30])  # [40 50]
print(arr[arr % 20 == 0])  # [20 40]
arr[arr > 30] = 0
print(arr)  # [10 20 30  0  0]
lists = [0, 2, 4]
print(arr[lists])  # [10 30 50]
```

NumPy 메서드

codingOn

- 배열생성
- zeros(): 모든 값이 0인 배열 생성
- ones(): 모든 값이 1인 배열 생성
- arange(start, stop, step, type): 연속된 숫자로 배열 생성
- linspace(): 구간을 일정하게 나눈 값으로 배열 생성(매개변수 아래표참고)

start	시작 값	없음
stop	끝 값	없음
num	생성할 값의 개수.	50
endpoint	True이면 끝 값을 포함, False이면 미포함	TRUE
retstep	True로 설정하면 값 사이의 간격(스텝)도 반환	FALSE
dtype	생성된 배열의 데이터 타입. 입력되지 않으면 자동 결정	None

NumPy 메서드

codingOn

- 예제코드

```
import numpy as np

zeros_array = np.zeros((2, 3))
ones_array = np.ones((3, 2))
range_array = np.arange(1, 10, 2)  # 1부터 10까지, 2 간격
linspace_array = np.linspace(0, 1, 5)  # 0에서 1까지 5개 값
```

- arange vs linspace 비교

속성	np.arange()	np.linspace()
생성 기준	증가 간격(step)을 직접 지정	값 개수(num)를 기준으로 생성.
끝 값 포함 여부	끝 값 미포함	기본적으로 끝 값 포함(endpoint=True)
용도	증가 간격이 명확한 정수/실수 배열 생성	지정된 구간을 일정한 간격으로 나눌 때 사용

NumPy 메서드

- 배열형태 변경
- reshape()
  - 배열의 형태를 변경하지만, 기존 배열의 데이터 크기를 유지
  - 새롭게 지정한 shape의 총 원소 개수는 기존 배열의 원소 개수와 같아야 함
  - 원소 개수가 틀리면 에러
- resize()
  - 배열의 형태를 변경하면서, 새로운 크기에 맞게 배열을 조정
  - 새 shape의 총 원소 개수가 기존 배열과 달라도 가능
  - 새 shape에 원소가 부족하면 데이터를 반복하여 채움

NumPy 메서드

- reshape() 과 resize()

```python
import numpy as np

# 1차원 배열을 2x3 배열로 변환
array = np.array([1, 2, 3, 4, 5, 6])
reshaped = np.reshape(array, (2, 3))  # (행, 열)

# 새 shape에 맞게 조정
resized = np.resize(array, (3, 5))
```

NumPy 연산

codingOn

- 연산

```
# 기본연산
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print(a + b)  # [5 7 9]
print(a - b)  # [-3 -3 -3]
print(a * b)  # [4 10 18]
print(a / b)  # [0.25 0.4 0.5]

a = np.array([1, 4, 9, 16, 25])
# 제곱근 계산
sqrt_values = np.sqrt(a)
print("제곱근:", sqrt_values)  # [1. 2. 3. 4. 5.]
# 지수 함수 계산
exp_values = np.exp(a)
print("지수 함수:", exp_values)
```

NumPy 연산

codingOn

- 연산

```
a = np.array([1, 2.718, 10, 20])
# 자연 로그 계산
log_values = np.log(a)
print("로그:", log_values)

# 각도 값을 라디안으로 변환
angles = np.array([0, np.pi/6, np.pi/4, np.pi/3, np.pi/2])
# 0°, 30°, 45°, 60°, 90°
# 사인 함수 계산
sin_values = np.sin(angles)
print("사인:", sin_values)
# 코사인 함수 계산
cos_values = np.cos(angles)
print("코사인:", cos_values)
```

배열합치기

- 수평 합치기 (hstack)
- 수직 합치기 (vstack)
- 열 기준 합치기 (column_stack)

```python
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

# 수평 합치기 (hstack)
result = np.hstack((a, b))
print(result)  # [1 2 3 4 5 6]

# 수직 합치기 (vstack)
result = np.vstack((a, b))
print(result)
# [[1 2 3]
#  [4 5 6]]

# 열 기준 합치기 (column_stack)
result = np.column_stack((a, b))
print(result)
# [[1 4]
#  [2 5]
#  [3 6]]
```

배열분할하기

- 수평 분할 (hsplit)
- 수직 분할 (vsplit)

```
a = np.array([[1, 2, 3], [4, 5, 6]])

result = np.hsplit(a, 3)
print(result)
# [array([[1],
#         [4]]),
#  array([[2],
#         [5]]),
#  array([[3],
#         [6]])]
result = np.vsplit(a, 2)
print(result)
# [array([[1, 2, 3]]), array([[4, 5, 6]])]
```

브로드캐스팅

- 서로 다른 크기의 배열 간 연산을 지원하는 기능
- 작은 배열의 크기를 큰 배열의 크기에 맞게 확장
- 필요한 경우 배열의 차원을 추가하여 크기를 일치

```python
import numpy as np

a = np.array([1, 2, 3, 4])
scalar = 10
result = a + scalar
print(result) # [11 12 13 14]

a = np.array([[1, 2, 3],
              [4, 5, 6]])
b = np.array([10, 20, 30])

result = a + b
print(result)
# [[11 22 33]
#  [14 25 36]]

a = np.array([[1, 2, 3],
              [4, 5, 6]])
b = np.array([[10], [20]])

result = a + b
print(result)
# [[11 12 13]
#  [24 25 26]]

a = np.array([[[1, 2, 3],
               [4, 5, 6]]])  # 3차원 배열 (1x2x3)
b = np.array([[10, 20, 30]])  # 2차원 배열 (1x3)

result = a + b
print(result)
# [[[11 22 33]
#   [14 25 36]]]
```

실습. 야구기록 데이터 정렬

codingOn

- KBO > 기록,순위 > 팀순위
- 팀 순위 데이터를 크롤링하여 데이터를 정렬한 후 txt파일에 저장하고 아래와 같이 출력결과를 내보세요

```
import numpy as np

with open("datas/2024kbo.txt", 'r') as f:
    data = f.read()
    print(data)
```

```
========== 2024 한국 프로야구 성적표 ==========
순위    팀      승      패      무      승률
1       KIA     71      48      2       0.597
2       삼성    66      54      2       0.55
3       LG      63      55      2       0.534
4       두산    62      60      2       0.508
5       KT      59      61      2       0.492
6       SSG     58      62      1       0.483
7       한화    56      60      2       0.483
8       롯데    51      61      3       0.455
9       NC      52      63      2       0.452
10      키움    53      67      0       0.442
```

수고하셨습니다