K-Digital Training 신재생에너지 활용 IoT 과정

codingOn

NumPy

codingOn

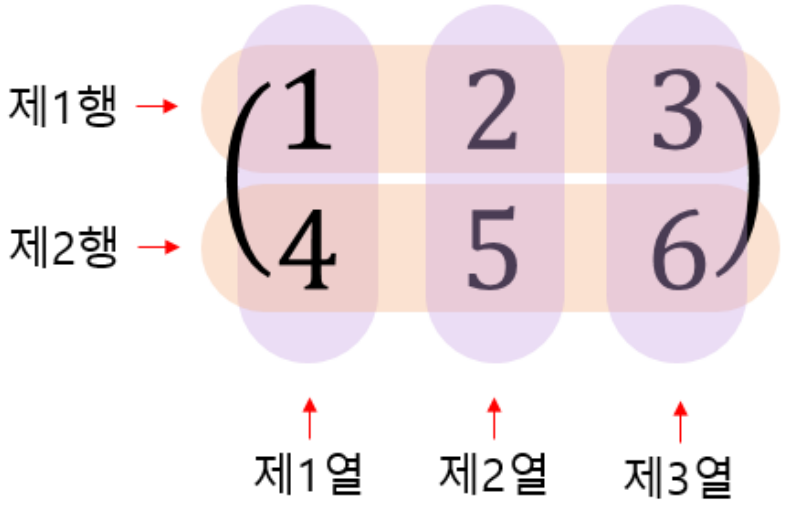
NumPy?

- 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

codingOn

NumPy

- **Ndarray 타입**의 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
```

codingOn

NumPy

- 배열생성

```python
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

실습 1

- array_1d, array_2d, array_3d 각각에 대해 다음 속성을 출력하시요.
  - shape, ndim, dtype, itemsize, size

NumPy

- 배열접근

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

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

1-1. 원소 읽기 - [행번호, 열번호]

```
print(arr[0, 1])
print(arr[2, 2])
```

```
2
9
```

1-2. 원소에 값 쓰기

```
arr[1, 1] = 0
print(arr)
```

```
[[1 2 3]
 [4 0 6]
 [7 8 9]]
```

NumPy

- 배열접근

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

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

2-1. 행 읽기 - [행번호] 또는 [행번호, :]

```
print(arr[2])
```

```
[7 8 9]
```

```
print(arr[2, :])
```

```
[7 8 9]
```

2-2. 행에 값을 쓰기

```
arr[2] = 0
print(arr)
```

```
[[1 2 3]
 [4 5 6]
 [0 0 0]]
```

3-1. 열 읽기 - [: , 열번호]

```
print(arr[:, 2])
```

```
[3 6 9]
```

3-2. 열에 값을 쓰기

```
arr[:, 2] = 0
print(arr)
```

```
[[1 2 0]
 [4 5 0]
 [0 0 0]]
```

실습 2

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

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

- 주어진 2차원 배열에서,
  - 두번째 행을 모두 100으로 값을 변경하시요.
  - 두번째 열을 모두 -100으로 값을 변경하시요.
  - 변경된 2차원 배열을 출력하시요.

```
[[   1 -100    3]
 [ 100 -100  100]
 [   7 -100    9]]
```

NumPy

- 배열접근

```
arr = np.array([10, 20, 30, 40, 50])
print(arr)
```

```
[10 20 30 40 50]
```

4. 조건식으로 값 읽기

```
print(arr[arr > 30])
```

```
[40 50]
```

```
print(arr[arr % 20 == 0])
```

```
[20 40]
```

5. 원소 선택하기

```
select = [0, 2, 4]
print(arr[select])
```

```
[10 30 50]
```

실습 3

- 다음 1차원 배열에서 양수 값만 출력하시요. `[20 10 50]`

```
arr = np.array([-10, 20, 0, -30, 10, 50])
print(arr)
```

```
[-10  20   0 -30  10  50]
```

실습 4 (선택문제)

- 다음 1차원 배열의 음수를 모두 0으로 변경하시오. `[ 0 20  0  0 10 50]`

```
arr = np.array([-10, 20, 0, -30, 10, 50])
print(arr)
```

```
[-10  20   0 -30  10  50]
```

NumPy 메서드

- 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

- 예제코드

```python
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
```

```
[[0. 0. 0.]     [[1. 1.]
 [0. 0. 0.]]     [1. 1.]
                 [1. 1.]]

[1 3 5 7 9]

[0.   0.25 0.5  0.75 1.  ]
```

- 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

실습 5.

- 모든 원소 값이 0인, 1000 x 1000 2차원 배열을 생성하시요.
- 생성한 배열의 총 원소 개수를 출력하시요.

실습 6. (선택문제)

- n x m의 2차원 배열을 생성하고,1 부터 n*m까지 숫자로 값을 채우시요.
  - ex) 2 x 3 배열

```
[[1 2 3]
 [4 5 6]]
```

- 이러한 배열을 반환하는 함수를 작성하시요.
  - make_array(int n, int m)
    - n x m 배열을 만들고 값을 채워서 반환

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))
```

```
[[1 2 3]
 [4 5 6]]
```

```
[[1 2 3 4 5]
 [6 1 2 3 4]
 [5 6 1 2 3]]
```

실습 7

- arange로 다음 1차원 배열을 생성하시요.

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
```

- reshape()을 사용하여 4 x 4 배열로 변경하시요.

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
```

실습 8 (선택문제)

- 실습 6의 make_array(int n, int m) 함수를
  - arange()와 reshape() 메소드를 이용하여 작성하시요.

NumPy 연산

- 연산

```python
# 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)
```

상수 연산

```python
print(a + 10)
```

```
[11 12 13]
```

```python
print(a * 10)
```

```
[10 20 30]
```

평균 구하기

```python
print((a + b) / 2)
```

```
[2.5 3.5 4.5]
```

실습 9

- 다음은 학생들의 중간고사, 기말고사 점수이다.

```
mid_score = np.array([90, 100, 85, 60, 75])
final_score = np.array([80, 90, 75, 50, 55])
```

- 학생별 평균 점수를 구하시요.

```
[85. 95. 80. 55. 65.]
```

실습 10

- 다음 1차원 배열 원소의 부호를 반대로 변경하여 출력하시요.

```
arr = np.array([-10, 20, 0, -30, 10, 50])
print(arr)
```

```
[-10  20   0 -30  10  50]
```

```
[ 10 -20   0  30 -10 -50]
```

실습 11 (선택문제)

- 다음 1차원 배열에서 음수인 원소를 양수로 변경하시요.

```
arr = np.array([-10, 20, 0, -30, 10, 50])
print(arr)
```

```
[-10  20   0 -30  10  50]
```

```
[10 20  0 30 10 50]
```

배열합치기

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

```
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]]
```

실습 12

- 다음은 학생들의 중간고사, 기말고사 점수이다.

```
mid_score = np.array([90, 100, 85, 60, 75])
final_score = np.array([80, 90, 75, 50, 55])
```

- 중간, 기말 점수를 아래와 같이 하나로 만드시요.

```
[[ 90 100  85  60  75]
 [ 80  90  75  50  55]]
```

-> 중간고사

-> 기말고사

실습 13 (선택 문제)

- 다음은 학생들의 중간고사, 기말고사 점수이다.

```
mid_score = np.array([90, 100, 85, 60, 75])
final_score = np.array([80, 90, 75, 50, 55])
```

- 중간, 기말, 평균 점수를 아래와 같이 하나로 만드시요.

```
[[ 90. 100.  85.  60.  75.]
 [ 80.  90.  75.  50.  55.]
 [ 85.  95.  80.  55.  65.]]
```

-> 중간고사

-> 기말고사

-> 평균 점수

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]]
```

Numpy의 N차원 배열

codingOn

❖ 브로드캐스트 – 형상이 다른 배열끼리 연산 수행

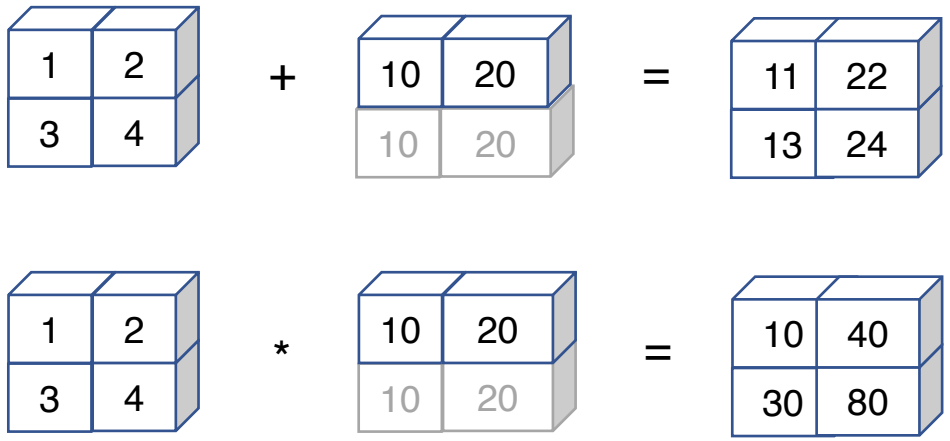
```
a = np.array([[1, 2], [3, 4]])
b = np.array([[10, 20]])
```

```
print(a + b)
```

```
[[11 22]
 [13 24]]
```

```
print(a * b)
```

```
[[10 40]
 [30 80]]
```



실습 13. KBO 데이터

- KBO > 기록,순위 > 팀순위
- 팀 순위 데이터를 크롤링하여 2차원 배열에 저장하시요.

순위	팀명	경기	승	패	무	승률	게임차	최근10경기	연속	홈	방문
1	LG	17	14	3	0	0.824	0	7승0무3패	1패	9-0-1	5-0-2
2	SSG	15	9	6	0	0.600	4	5승0무5패	1패	6-0-1	3-0-5
3	KT	17	9	7	1	0.563	4.5	5승1무4패	2승	8-0-4	1-1-3
4	삼성	18	10	8	0	0.556	4.5	5승0무5패	2패	7-0-4	3-0-4
5	롯데	19	8	10	1	0.444	6.5	5승0무5패	1패	3-1-5	5-0-5
6	NC	16	7	9	0	0.438	6.5	4승0무6패	1승	1-0-4	6-0-5
7	두산	19	8	11	0	0.421	7	5승0무5패	1승	4-0-4	4-0-7
7	한화	19	8	11	0	0.421	7	5승0무5패	1승	4-0-4	4-0-7
9	KIA	17	7	10	0	0.412	7	5승0무5패	1승	4-0-5	3-0-5
10	키움	19	7	12	0	0.368	8	3승0무7패	1패	4-0-5	3-0-7

10 x 12 2차원 배열
데이터는 모두 문자열로 저장

실습 14. KBO 데이터 (선택문제)

- 2025년 팀별 승률 데이터를 수집하여 2차원 배열로 저장하시요.
  - 아래와 같이 행은 팀, 열은 날짜

	2025.03.22	2025.03.23	...	2025.04.13
LG	1.000	1.000	...	0.824
SSG	1.000	1.000	...	0.600
KT	0.000	0.500	...	0.563
삼성	1.000	1.000	...	0.556
롯데	0.000	0.000	...	0.444
...	...	...	...	...

10 x 19 크기의 2d array에 승률을 저장
- 팀수: 10
- 4/13일까지 경기일수: 19

codingOn

복.습.철.저

수고하셨습니다