직장인을 위한 파이썬 데이터 분석

Numpy Cheat Sheet

Numpy

NumPy

Numpy 는 다차원 배열을 **쉽고 효율적으로 계산**할 수 있도록 해주는 **Python 의 대표 라이브러리** 입니다.

배열(Arrays)

같은 타입을 가지는 데이터가 **나열된 집합**

라이브러리 로딩

```
import numpy as np
```

배열(Arrays) 생성하기

리스트(list)로 만들기

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

Numpy 자체 생성 함수

```
np.zeros((3,3))          0 배열 생성
np.ones((3,3))           1 배열 생성
np.arange(10,25,5)       균등 간격 배열 생성 (10~25 사이 5 간격)
np.linspace(10,25,5)     균등 간격 배열 생성 (10~25 사이 5개)
np.full((3,3),5)         지정 된 수로 배열 생성
np.eye(3)                3x3 단위 행렬 생성
np.random.random((3,3))  3x3 랜덤 배열 생성
np.empty((3,3))          3x3 빈 배열 생성
```

배열(Arrays) 파일 로딩 / 저장

```
np.save("myfile.npy", arr)                  # numpy 파일 저장
np.load("myfile.npy")                       # numpy 파일 로딩
np.loadtxt("myfile.txt")                    # txt 파일 로딩
np.genfromtxt("myfile.csv", delimiter=',')  # csv 파일 로딩
```

배열(Arrays) 정보 확인

```
array.shape              배열 차원 확인
len(array)               배열 길이 확인
array.ndim               차원 개수 확인
array.size               총 값 개수 확인
array.dtype              데이터 타입 확인
array.dtype.name         데이터 타입 이름 확인
array.astype(dtype)      데이터 타입 변경
```

배열(Arrays) 조작

복사 (Copying)

```
c = array.copy()         복사된 배열 생성
v = array.view()         뷰(view) 배열 생성
```

정렬 (Sorting)

```
array.sort()             배열 정렬 (오름차순)
array.sort(axis=0)       축을 기준으로 배열 정렬 (0 : 행, 1 : 열)
```

값 추가 / 삭제

```
np.append(array,[1])     값 추가
np.insert(array, 1, 5)   지정된 위치에 값 삽입 (1 인덱스에 5 삽입)
np.delete(array,[1])     값 삭제
```

배열 결합

```
np.concatenate((a,b), axis=0)   배열 결합 (0 : 행, 1 : 열)
np.vstack((a,b))                수직 방향으로 배열 결합
np.hstack((a,b))                수평 방향으로 배열 결합
```

배열 분할

```
np.vsplit(a,3)           수직 방향으로 배열 분할 (3개)
np.hsplit(a,3)           수평 방향으로 배열 분할 (3개)
```

차원 변환

```
array.ravel()            1차원 배열로 변환
array.reshape(2,2)       원하는 차원으로 변환 (2x2)
```

전치 연산

```
array.T                  배열 차원 전치 연산
```

Numpy 데이터 타입

```
np.int64                 64 비트 정수형
np.float32               32 비트 실수형
np.bool                  참(True), 거짓(False) 형
np.object                오브젝트 형 (가변 길이 문자형 포함)
```

Numpy 슬라이싱 / 인덱싱 / 산술 연산

슬라이싱 (Slicing)

```
a[2]                     두번째 인덱스 값 선택
> 3
b[1,2]          # b[1][2]
> 6.0                    1행 2열 인덱스 값 선택

a[0:2]                   0이상 2 미만 인덱스 슬라이싱
> array([1, 2])
b[0:2,1]                 0,1행 1열 인덱스 값 선택
> array([ 2., 5.])

b[:1]           # b[0:1, :]   0행 인덱스 슬라이싱
> array([[1., 2., 3.]])
c[1,...]        # c[1,:,:]    1행 인덱스 슬라이싱
> array([[[ 3., 2., 1.], [ 4., 5., 6.]]])
a[ : :-1]                    배열 역전 (reverse)
> array([3, 2, 1])
```

Boolean 인덱싱

```
a[a<2]                   2보다 작은 값 선택
> array([1])
```

Fancy 인덱싱

```
b[[1, 0, 1, 0],[0, 1, 2, 0]]   (1,0),(0,1),(1,2), (0,0) 인덱스
> array([ 4. , 2. , 6. , 1.])  값 선택
b[[1, 0, 1, 0]][:,[0, 1, 2, 0]]
> array([[ 4. ,5. , 6. , 4. ],  인덱스 지정을 통한 행렬의
         [ 1., 2. , 3. , 1.],   하위 집합 선택
         [ 4. , 5. , 6. , 4. ],
         [ 1., 2. , 3. , 1.]])
```

산술 / 집계 연산

```
array.sum()              합계 (0: 행, 1: 열)
array.min()              최소값 (0: 행, 1: 열)
array.max(axis=0)        최대값 (0: 행, 1: 열)
array.mean()             평균
array.median()           중앙값
np.std(array)            표준 편차

a - b        # np.subtract(a,b)   뺄셈
b + a        # np.add(b,a)        덧셈
a / b        # np.divide(a,b)     나눗셈 (element-wise)
a * b        # np.multiply(a,b)   곱셈 (element-wise)
np.dot(arr1, arr2)                내적연산
```

비교 연산

```
a == b                   배열 요소 간 값 비교 (등호)
> array([[True, True, True],
         [False, False, False]])
a < 2                    배열 요소 간 값 비교 (부등호)
> array([True, False, False])
np.array_equal(a, b)     배열 단위 비교 (등호)
```