numpy

- 모든 데이터 분석과 인공지능 학습에 있어 가장 필수적으로 사용되고 이용
- Numerical Python
- 파이선의 고성능 과학 계산용 패키지
- **Matrix와 Vector 와 같은 Array 연산의 사실상의 표준**
- 일반 리스트에 비해 빠르고 효율적
- 반복문 없이 데이터 배열에 대한 처리 지원
- 선형대수와 관련된 다양한 기능을 제공
- C, C++ 포트란 등의 언어와 통합 가능
- numpy install

  `conda install numpy`
- import

  `import numpy as np`
- ndarray
  - np.array([1, 4, 5, 8], float)
  - ndarray 객체
  - 하나의 데이터 타입만 배열에 넣을 수 있음
  - dynamic typing not supported
  - scalar, vector, matrix, n-tensor
- reshape
- -1: size를 기반으로 알아서 개수 선정
- np.flatten으로도 가능: 다차원 -> 1차원 array
- indexing, slicing
  - array [0] [2], array[0, 2]
  - array[:, 2:]
- arange
  - np.arange(start, end, step)
  - python과 달리 float도 가능
- eye
  - 대각선이 1인 행렬
  - 시작 index 설정 가능 (k=0일 때 np.identity와 같음)
- diag
  - 대각성분 추출
  - 시작 index 설정 가능 (k=2)
- random sampling
  - np.random.uniform()
  - np.random.normal()

    - np.random.exponential()
- operation functions
    - .sum(), .mean(), .std()
    - axis: operation functions 실행 시 기준이 되는 dimention 축
    - (axis=0, axis=1, axis=2, ...)
    - concatenate((a, b), axis=0)
    - vstack, hstack
    - newaxis (arr = arr[np.newaxis, :])
- operations b/w arrays
    - element-wise (same shape)
    - dot product
    - broadcasting (different shape)
    - all & any

```
a = np.arange(10)
np.all(a < 4)
# False
np.any(a < 4)
# True
# return: boolean array
# numpay array끼리 비교 시, 역시 return boolearn array
```

- np.where
    - np.where(condition, TRUE, FALSE)
    - np.where(condintion): index 값 반환

```
np.where(a > 0, 3, 2)
# array([3, 3, 2]) 처럼, True일 경우 3, False일 경우 2 return
a = np.arange(10)
np.where(a > 5)
# array([6, 7, 8, 9],) 조건에 해당하는 인덱스 반환
```

- argmax, argmin
    - array 내 최대(최소)값의 index 반환

```
a = np.array([1, 2, 3, 5, 7, 30])
np.argmax(a), np.argmin(a)
# (5, 0)
a = np.array([[1,2,4,7],[9,88,6,45],[9,76,3,4]])
np.argmax(a, axis=1), np.argmin(a, axis=0)
# (array([3, 1, 1]), array[0, 0, 2, 2])
```

- fancy index

```
a = np.array([2, 4, 6, 8], float)
condition = np.array([0, 3, 1], int) # index 값이므로 반드시 int!
a[condition]
# array([2., 8., 4.])
a.take(condition) # take 함수를 이용해 bracket index와 같은 결과 반환 가능
# array([2., 8., 4.])
```