NumPy

- 행렬 연산을 위한 핵심 라이브러리
- NumPy는 "**Numerical Python**"의 약자로 대규모 다차원 배열과 행렬 연산에 필요한 다양한 함수를 제공
- 특히 메모리 버퍼에 배열 데이터를 저장하고 처리하는 효율적인 인터페이스를 제공
- 파이썬 list 객체를 개선한 NumPy의 ndarray 객체를 사용하면 더 많은 데이터를 더 빠르게 처리할 수 있음

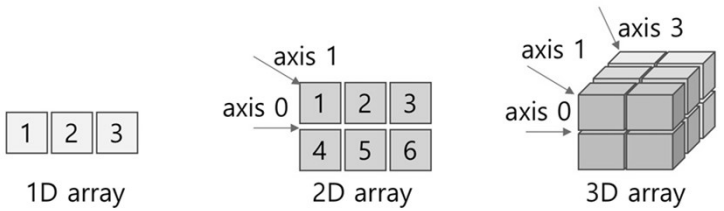
NumPy 특징

- 강력한 N 차원 배열 객체
- 정교한 브로드케스팅(Broadcast) 기능
- C/C ++ 및 포트란 코드 통합 도구
- 유용한 선형 대수학, 푸리에 변환 및 난수 기능
- 범용적 데이터 처리에 사용 가능한 다차원 컨테이너

NumPy 배열

- 과학 연산을 위한 파이썬 핵심 라이브러리
- NumPy를 사용할 때, numpy 모듈을 "np"로 임포트하여 사용
- 다차원 배열을 지원함
- NumPy 배열의 구조는 "**Shape**"으로 표현
- Shape는 배열의 구조를 파이썬 튜플 자료형을 이용하여 정의
- NumPy 배열은 numpy.ndarray 객체

NumPy 배열



NumPy 배열 생성

- 파이썬 배열을 인자로 NumPy 배열을 생성할 수 있음
- 파라미터로 list 객체와 데이터 타입(dtype)을 입력하여 생성
- dtype을 생략할 경우, 입력된 list 객체의 요소 타입이 설정됨

NumPy 배열 생성 및 초기화

- NumPy는 원하는 shape으로 배열을 설정하고,
- 각 요소를 특정 값으로 초기화하는 zeros, ones, full, eye 함수 제공
- 또한, 파라미터로 입력한 배열과 같은 shape의 배열을 만드는 zeros_like, ones_like, full_like 함수도 제공

NumPy 데이터 생성 함수

- NumPy는 주어진 조건으로 데이터를 생성한 후,
- 배열을 만드는 데이터 생성 함수를 제공
- numpy.linspace
- numpy.arange
- numpy.logspace

NumPy 난수 기반 배열 생성

- NumPy는 난수 발생 및 배열 생성을 생성하는 numpy.random 모듈을 제공

- np.random.normal
- np.random.rand
- np.random.randn
- np.random.randint
- np.random.random

NumPy 난수 발생 재연하기

- 무작위 수를 만드는 난수는 특정 시작 숫자로부터 난수처럼 보이는 수열을 만드는 알고리즘의 결과물임
- 따라서, 시작점을 설정함으로써 난수 발생을 재연할 수 있음
- 난수의 시작점을 설정하는 함수는 np.random.seed

NumPy 데이터 타입

- 배열을 생성할 때 dtype속성으로 다음의 데이터 타입을 지정할 수 있음

- np.int64 : 64 비트 정수 타입
- np.float32 : 32 비트 부동 소수 타입
- np.complex : 복소수 (128 float)
- np.bool : 불린 타입 (Trur, False)
- np.object : 파이썬 객체 타입
- np.string_ : 고정자리 스트링 타입
- np.unicode_ : 고정자리 유니코드 타입

NumPy 배열 상태 검사

배열 속성 검사 항목	배열 속성 확인 방법	예시	결과
배열 shape	np.ndarray.shape 속성	arr.shape	(5, 2, 3)
배열 길이	일차원의 배열 길이 확인	len(arr)	5
배열 차원	np.ndarray.ndim 속성	arr.ndim	3
배열 요소 수	np.ndarray.size 속성	arr.size	30
배열 타입	np.ndarray.dtype 속성	arr.dtype	dtype('float64')
배열 타입 명	np.ndarray.dtype.name 속성	arr.dtype.name	float64
배열 타입 변환	np.ndarray.astype 함수	arr.astype(np.int)	배열 타입 변환

NumPy 도움말

- NumPy의 모든 API는 np.info 함수를 이용하여 도움말을 확인할 수 있음

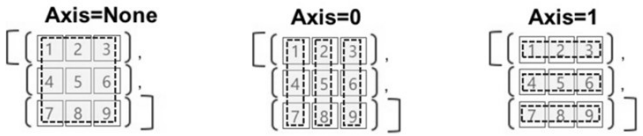
NumPy 배열 연산

- NumPy는 기본 연산자를 연산자 재정의하여 배열(행렬) 연산에 대한 직관적으로 표현을 강화하였음
- 모든 산술 연산 함수는 np 모듈에 포함되어 있음

- 산술 연산(Arithmetic Operations)
- 비교 연산(Comparison)

NumPy 집계 함수

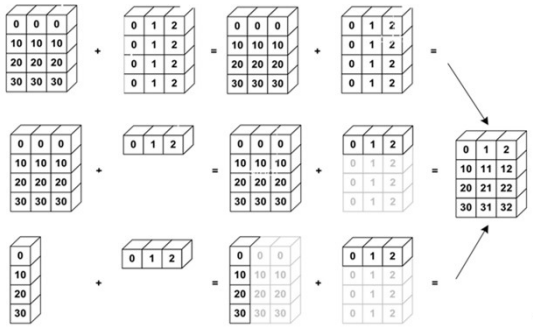
- NumPy의 모든 집계 함수는 집계 함수는 axis를 기준으로 계산
- 집계함수에 axis를 지정하지 않으면 axis=None임



NumPy 브로드캐스팅

- Shape가 같은 두 배열에 대한 이항 연산은 배열의 요소별로 수행
- 두 배열 간의 Shape이 다를 경우, 두 배열 간의 형상을 맞추는 Broadcasting 과정을 거침

NumPy 브로드캐스팅



NumPy 백터연산

- ▶ NumPy는 벡터 연산을 지원함
- ▶ NumPy의 집합 연산에는 Vectorization이 적용되어 있음
- ▶ 즉, 배열 처리에 대한 벡터 연산을 적용할 경우, 처리 속도가 100배 이상 향상됨
- ▶ 머신러닝에서 선형대수 연산을 처리할 때 매우 높은 효과가 있음

NumPy 배열 정렬

- ▶ ndarray 객체는 axis를 기준으로 요소 정렬하는 sort 함수를 제공함

NumPy 요소 선택

- ▶ 배열의 각 요소는 axis 인덱스 배열로 참조할 수 있음
- ▶ 1차원 배열은 1개 인덱스, 2차원 배열은 2개 인덱스, 3차원 인덱스는 3개 인덱스로 요소를 참조할 수 있음
- ▶ 인덱스로 참조한 요소는 값 참조, 값 수정이 모두 가능함

NumPy 슬라이싱

- 여러 개의 배열 요소를 참조할 때 슬라이싱을 사용함
- 슬라이싱은 axis 별로 범위를 지정하여 실행함
- 범위는 fromindex:toindex 형태로 지정함
- from_index는 범위의 시작 인덱스
- to_index는 범위의 종료 인덱스
- 범위를 지정할 때 to_index는 결과에 포함되지 않음

NumPy 슬라이싱

- fromindex:toindex의 범위 지정에서 from_index는 생략 가능
- 생략할 경우 0을 지정한 것으로 간주
- to_index 역시 생략 가능 => 마지막 인덱스로 설정
- from_index와 to_index에 음수를 지정하면 반대 방향을 의미
- 슬라이싱은 원본 배열의 뷰임
- 따라서, 슬라이싱 결과의 요소를 업데이트하면 원본에 반영됨

NumPy 슬라이싱

- fromindex:toindex의 범위 지정에서 from_index는 생략 가능
- 생략할 경우 0을 지정한 것으로 간주
- to_index 역시 생략 가능 => 마지막 인덱스로 설정
- from_index와 to_index에 음수를 지정하면 반대 방향을 의미
- 슬라이싱은 원본 배열의 뷰임
- 따라서, 슬라이싱 결과의 요소를 업데이트하면 원본에 반영됨

NumPy 불린 인덱싱(Boolean Indexing)

- NumPy의 불린 인덱싱은 배열 각 요소의 선택 여부를 True, False 지정하는 방식
- 해당 인덱스의 True만을 조회함

NumPy 배열 변환 – 전치

- 배열을 변환하는 방법으로는 전치, 배열 shape 변환, 배열 요소 추가, 배열 결합, 분리 등이 있음
- Tranpose는 행렬의 인덱스가 바뀌는 변환임
- NumPy에서 행렬을 전치하기 위하여 [numpy.ndarray 객체].T 속성을 사용함

$$[1\ \ 2]^T = \begin{bmatrix}1\\2\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^T = \begin{bmatrix}1 & 3\\2 & 4\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}^T = \begin{bmatrix}1 & 3 & 5\\2 & 4 & 6\end{bmatrix}$$

NumPy 배열 형태 변경

- numpy.ndarray 객체는 배열의 Shape을 변경하는 ravel 메서드와 reshape 메서드를 제공함
- ravel은 배열의 shape을 1차원 배열로 만드는 메서드
- reshape은 데이터 변경없이 지정된 shape으로 변환하는 메서드

numpy.ndarray객체.ravel()

- 배열을 1차원 배열로 반환하는 메서드
- numpy.ndarray 배열 객체의 View를 반환함
- ravel 메서드가 반환하는 배열은 원본 배열이 뷰임
- 따라서 ravel 메서드가 반환하는 배열의 요소를 수정하면 원본 배열 요소에도 반영됨

numpy.ndarray객체.reshape()

- [numpy.ndarray 객체]의 shape을 변경함
- 실제 데이터를 변경하는 것은 아님
- 배열 객체의 shape 정보만을 수정함

NumPy 배열 요소 추가 삭제

- 배열의 요소를 변경, 추가, 삽입 및 삭제하는
- resize, append, insert, delete 함수를 제공함

resize()

- 배열의 shape과 크기를 변경함
- np.resize와 np.reshape 함수는 배열의 shape을 변경한다는 부분에서 유사함
- 차이점은 reshape 함수는 배열 요소 수를 변경하지 않음
- reshape 전후 배열의 요소 수는 같은 반면에
  resize는 shape을 변경하는 과정에서 배열 요소 수를 줄이거나 늘림

append()

- 배열의 끝에 값을 추가
- np.append 함수는 arr의 끝에 values(배열)을 추가
- axis로 배열이 추가되는 방향을 지정할 수 있음

insert()

- axis를 지정하지 않으며 1차원 배열로 변환
- 추가할 방향을 axis로 지정

delete()

- np.delete(arr, obj, axis=None)
- axis를 지정하지 않으며 1차원 배열로 변환
- 삭제할 방향을 axis로 지정
- delete 함수는 원본 배열을 변경하지 않으며 새로운 배열을 반환

delete()

- np.delete(arr, obj, axis=None)
- axis를 지정하지 않으며 1차원 배열로 변환
- 삭제할 방향을 axis로 지정
- delete 함수는 원본 배열을 변경하지 않으며 새로운 배열을 반환

NumPy 배열 결합

- **np.concatenate**
- concatenate((a1, a2, ...), axis=0)
- a1, a2....: 배열

NumPy 배열 결합

- **np.vstack**
- 튜플로 설정된 여러 배열을 수직 방향으로 연결 (axis=0 방향)
- np.concatenate(tup, axis=0)와 동일

NumPy 배열 결합

- **np.hstack**
- 튜플로 설정된 여러 배열을 수평 방향으로 연결 (axis=1 방향)
- np.concatenate(tup, axis=1)와 동일

NumPy 배열 분리

- NumPy는 배열을 수직, 수평으로 분할하는 함수를 제공함

- np.vsplit(ary, indices_or_sections)
  지정한 배열을 수직(행) 방향으로 분할
- np.hsplit(ary, indices_or_sections)
  지정한 배열을 수평(열) 방향으로 분할