**4. NumPy Basics**

**1) 개요**

★NumPy(Numerical Python): 과학 계산, 데이터 분석에 사용되는 배열 패키지

- 빠르면서도 효율적으로 공간을 사용해 고성능 수치 계산을 추구

- 배열에 대한 기본적인 수학 계산 함수 제공

- 배열 파일 읽기/쓰기

- 선형대수, 난수 발생, 푸리에 변환

- C, C++, 포트란(Fortran) 언어 코드 통합: C API를 쉽게 사용 가능

- Pandas 패키지를 효율적으로 사용하는 데 기초가 됨

★ R 언어와 유사한 형태로 사용 가능

**2) NumPy ndarray: 다차원 배열 객체**

벡터 연산을 제공하는 NumPy의 기본 객체로, 한 가지 데이터 타입만 저장 가능함

(1) 객체 생성/객체 속성

① array([Seq.]) 함수를 사용해 [Seq.] 의 내용을 배열로 변환

→ asarray([Seq.]): [Seq.]가 이미 배열 객체이면 새로 객체를 만들지 않음

→ [Seq.]: 리스트, 튜플, 문자열 등 순서가 있는 객체

② zeros([차원]): 0을 원소로 하는 배열 생성 / ones([차원]): 1을 원소로 하는 배열 생성

③ empty([차원]): 0~1 사이의 garbage 값을 가지는 배열 생성

④ arrange([숫자]): 0부터 [숫자]-1 의 값을 가지는 1차원 배열(벡터) 생성

→ Python 기본 함수 range()와 같음

⑤ eye[identity]([숫자]): [숫자]\*[숫자] 단위행렬 배열 생성

⑥ ndim: 객체의 차원 반환, shape: 객체의 모양을 튜플로 반환

⑦ dtype: 객체의 타입 반환

⑧ reshape([차원]): [차원]의 형태로 객체의 모양 변경

→ 인수로 튜플이 들어감(z축, y축, x축 순)

(2) ndarray 데이터 타입

① 배열을 만들 때, dtype 인수로 타입을 지정할 수 있고, dtype() 함수(인스턴스)로 속성을 가져올 수 있음

→ 뒤에 붙은 숫자는 bit 수를 의미

→ Default: 정수는 32bit, 실수는 64bit, 문자열은 최대 문자열의 길이

→ 이름 대신 Type Code로도 타입 지정 가능

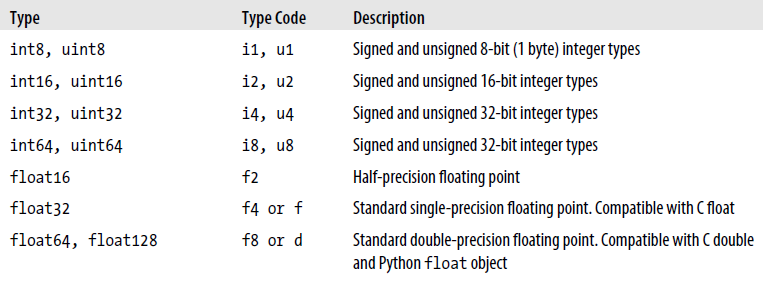
② Type Casting(형 변환): astype([타입]) 인스턴스 활용

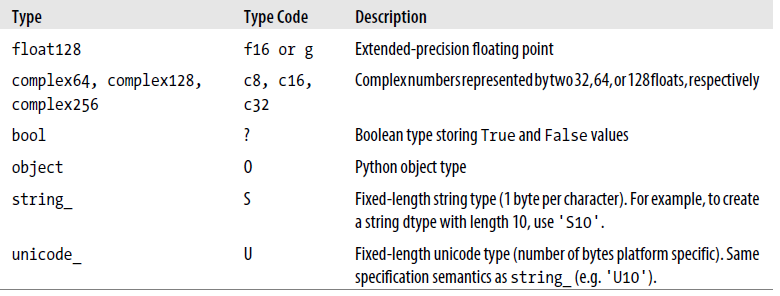
→ 실수를 정수로 변환할 경우 소수점 이하는 버림

→ 다른 배열의 형을 dtype 메소드로 가져와 인수로 활용할 수도 있음

→ 변환이 불가능한 타입을 지정하면 에러를 띄움

→ astype는 변환 전과 변환 후 타입이 같더라도 객체를 복사함





(3) 배열과 스칼라(숫자)의 연산

① for 문 없이도 일괄적으로 연산 가능

② 기본 Bulit-in 연산과 동일(벡터 합과 차, 스칼라 곱)

(4) Indexing, Slicing

① Python의 Indexing과 Slicing 규칙을 따름

② NumPy 배열 객체의 Indexing이나 Slicing은 원본 배열을 참조하는 형태(‘뷰’ 개념)

→ 데이터를 바꾸면 원본이 바뀜

③ 복사(원본 유지)를 원할 경우 배열의 copy() 인스턴스를 사용해 새로운 변수에 저장해야 함

(5) Boolean Indexing

① Bool 리스트가 True가 되는 위치의 값을 Indexing

② 다양한 조건식 사용 가능

(6) Fancy Indexing

① 원하는 순서로 결과를 확인하고 싶을 때 사용

② Indexing Number를 리스트로 넣어 데이터를 뽑을 수 있음

→ ex) arr[[4, 3, 0, 6]]: 배열의 4, 3, 0, 6 번째 행을 순서대로 추출

③ 리스트 내의 숫자 순서대로 정렬해 추출함

(7) 배열 Transpose(전치), 축(x, y, z) 변환

① 간단한 Transpose는 (객체).T 인스턴스로 가능함(swapaxes()의 특수 케이스)

② transpose([축 번호]), swapaxes([축 번호]) 인스턴스: 인수로 주어진 축끼리 자리이동

→ transpose는 모든 축 번호를, swapaxes는 바꿀 축 번호만 입력

→ swapaxes 인스턴스가 더 직관적

③ 축 번호가 낮을수록 고차원 축: 일반적인 논리 순서가 아닌, 인덱싱 순서

→ ex) 3차원 0, 1, 2 = z, y, x / 2차원 0, 1 = y, x

④ 축 번호 이상의 숫자를 넣으면 에러

**3) Universal 함수**

① sqrt([객체]): 제곱근, exp([객체]):

② maximum[minimum](x, y): 인수로 들어간 배열 객체의 같은 자리 원소를 비교해 큰 값[작은 값]을 리스트로 반환

③ modf([객체]): 정수 부분과 소수 부분을 분리해 실수형 객체로 각각 반환

④ 이 외 수많은 함수들(p. 96)



**4) 배열을 활용한 데이터 처리**

반복 작업이 필요한 데이터를 손쉽게 처리할 수 있음

(1) 조건문을 이용한 처리

① where([조건], [참], [거짓]): [조건]이 참이면 [참], 거짓이면 [거짓] 값을 반환

→ ex) np.where(arr>0, 1, -1)

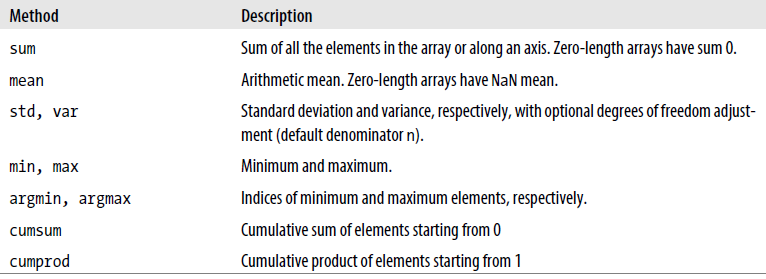
② if 구문을 간단하게 표현 가능

(2) 수학, 통계적 도구

① mean([축]), sum([축]): [축]을 기준으로 계산

→ 디폴트는 축을 고려하지 않은 전체 계산

② cumsum([축]), cumprod([축]): 누적 합, 누적 곱 계산



(3) Boolean 배열 도구

① any([축]), all([축]): 축을 기준으로 논리 연산 수행

→ any = or, all = and 의미를 가짐

② Boolean(True/False)이 아닌 일반 값도 연산 가능(Python의 T/F 판정 방법에 따름)

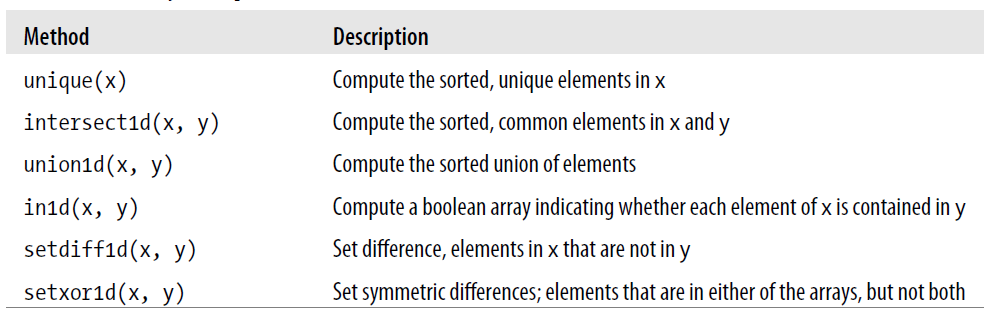
(4) 정렬/기타

① sort([축]): [축]을 기준으로 정렬

② unique([객체]): 중복을 제거한 배열 객체 반환

③ in1d([객체], [조건 리스트]): 객체와 조건 리스트를 비교해 T/F 반환

④ 이 외 수많은 함수들(p. 103)



**5) 파일 입/출력**

(1) Binary 포맷 입/출력

① save([파일 이름], [객체]): [객체]를 [파일 이름] 파일에 저장

→ 파일 경로 지정 가능

② savez([파일 이름], [객체1], [객체2], …): 객체들을 [파일 이름] 파일에 저장

③ load([파일 이름]): 파일을 불러와 배열 객체 반환

→ 배열 객체가 여러 개 저장된 경우(.npz 확장자) 사전 형태로 반환

(2) Text(CSV) 파일 입/출력

① savetxt([파일 이름], [객체], delimiter=[분리 문자열]): [분리 문자열]을 [객체]의 원소 사이에 넣어 [파일 이름]에 저장

② loadtxt([파일 이름], delimiter=[분리 문자열]): [분리 문자열]을 기준으로 원소를 나누어 텍스트를 배열 객체에 불러옴

③ genfromtxt(): loadtxt()와 유사하며, structured array를 불러올 때 사용

→ 결측 값 처리 등 다양한 기능 제공

**6) 선형 대수**

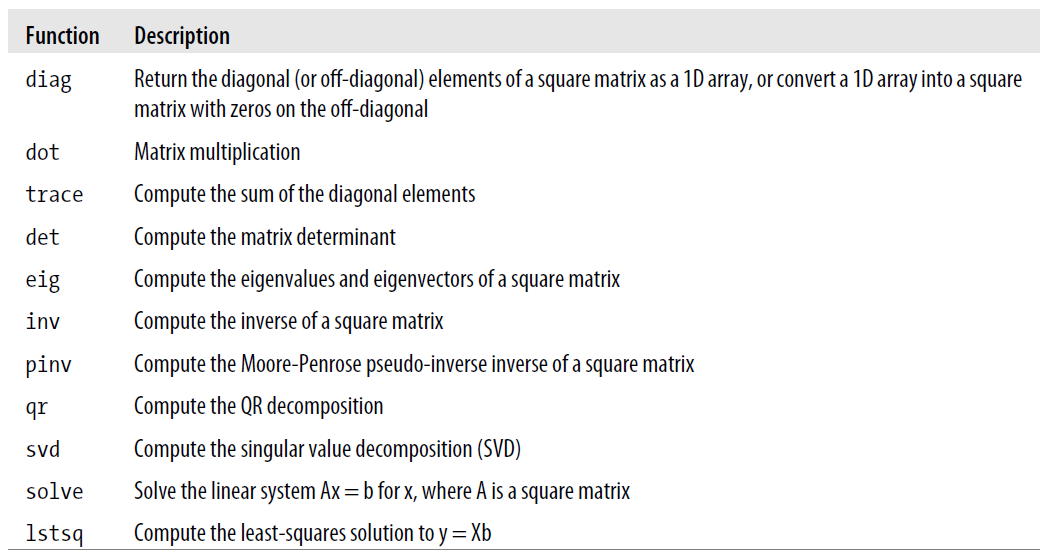
① NumPy 내 linalg 모듈을 import해 사용

② inv([객체]): 역행렬, qr([객체]): Q-R Decomposition(튜플 형태로 반환)

③ [객체1].dot([객체2]): [객체1] 과 [객체2]의 행렬 곱(내적) 수행

→ dot([객체1], [객체2])와 동일

④ 이 외 수많은 함수들(p. 106)

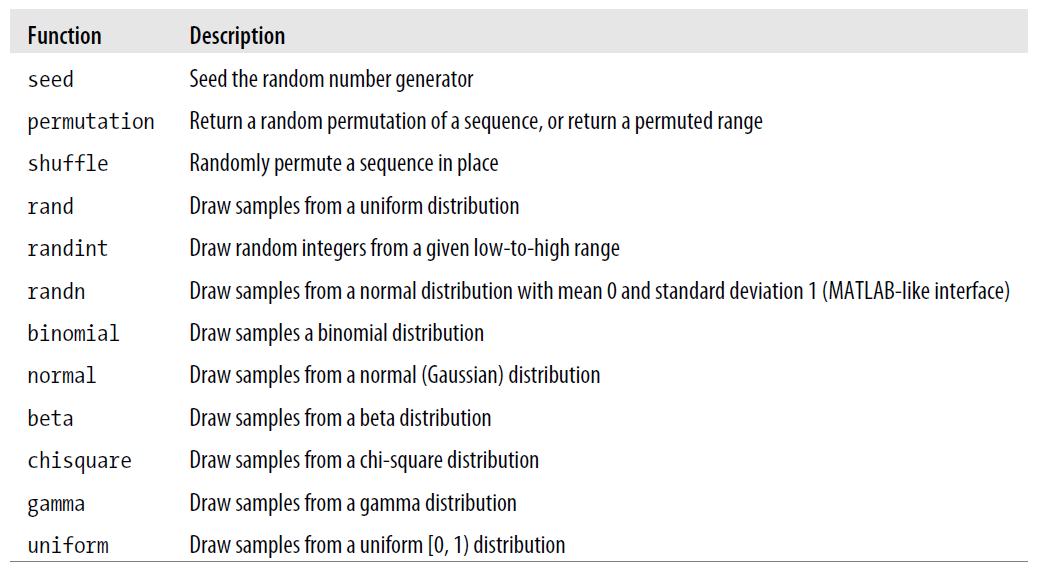


**7) 난수 생성**

① NumPy 내 random 모듈을 import해 사용

② 기본 제공 함수보다 속도가 빠름

③ 다양한 분포에서의 난수 생성 함수를 제공함



④ ex) Random Walk 구현

→ 시계열 모형에서 활용되는 개념

→ ARIMA 모델에서, MA(Moving Average)를 구성하는 요소

참고자료

Python for Data Analysis CH.4 Numpy Basics

https://s3.amazonaws.com/assets.datacamp.com/blog\_assets/Numpy\_Python\_Cheat\_Sheet.pdf