파이썬 라이브러리를 활용한 데이터 분석

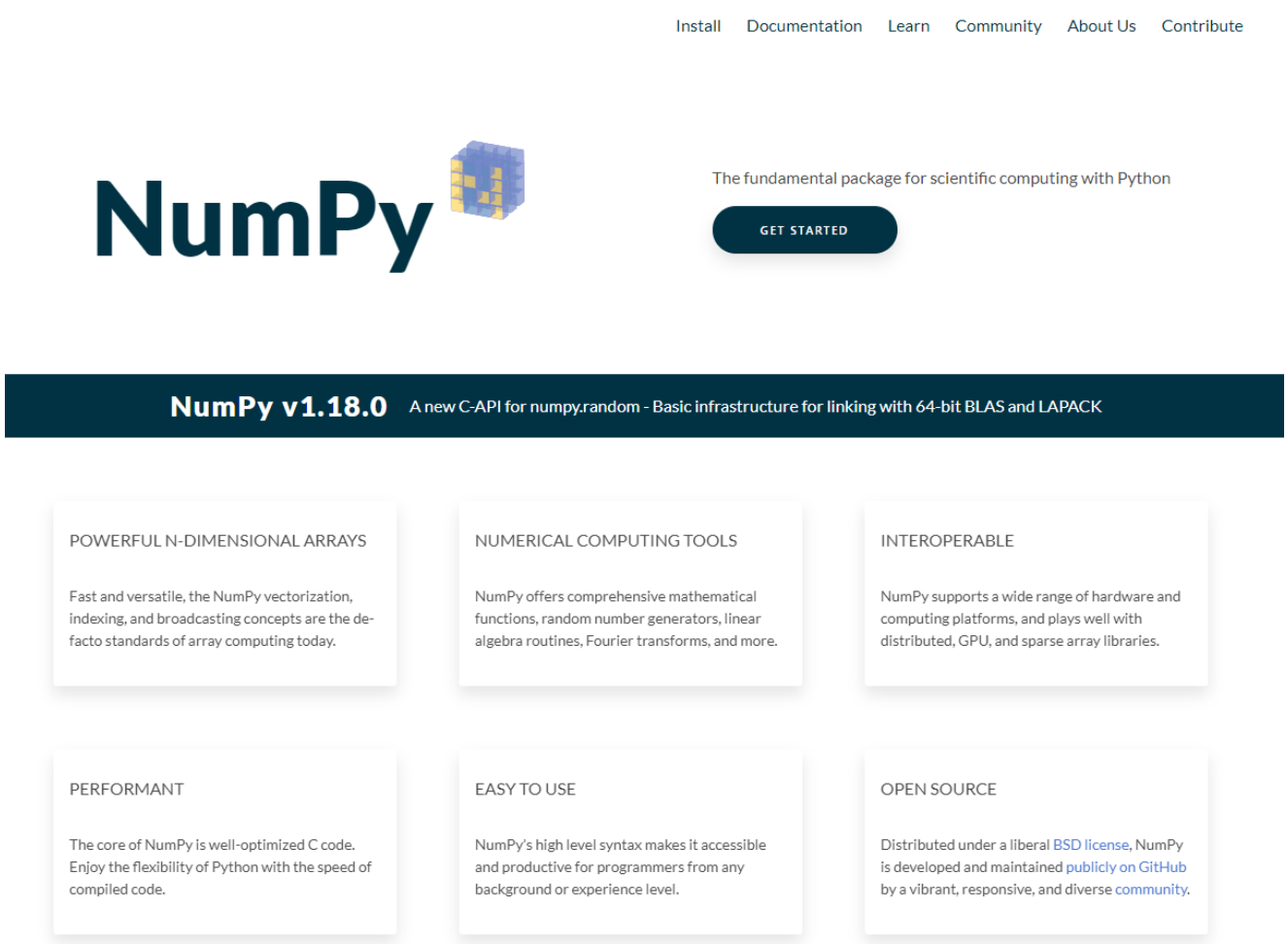
4장 numpy 기본: 배열과 벡터 연산

2020.06.19(금) 2h

Numpy 기본: 배열과 벡터 연산

- **Numerical python**
  - 과학 기술을 위한 산술 계산 라이브러리
    - **대규모 다차원 배열과 행렬 연산에 필요한 다양한 함수를 제공**
  - 제공 기술
    - **다차원 배열 ndarray**
      - **정교한 브로드캐스팅(Broadcast) 기능**
    - **전체 데이터 배열을 빠르게 계산하는 표준 수학 함수**
    - **선형대수, 난수 생성기**
  - 장점
    - **대용량 배열 데이터를 효율적으로 다뤄 빠르게 처리**
      - **중요 알고리즘 구현은 C로 작성**
      - **반복문을 사용하지 않고 빠르게 계산**

numpy.org


Install
Documentation
Learn
Community
About Us
Contribute
NumPy
The fundamental package for scientific computing with Python
GET STARTED
NumPy v1.18.0
A new C-API for numpy.random - Basic infrastructure for linking with 64-bit BLAS and LAPACK
POWERFUL N-DIMENSIONAL ARRAYS
Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.
NUMERICAL COMPUTING TOOLS
NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.
INTEROPERABLE
NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.
PERFORMANT
The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.
EASY TO USE
NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.
OPEN SOURCE
Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

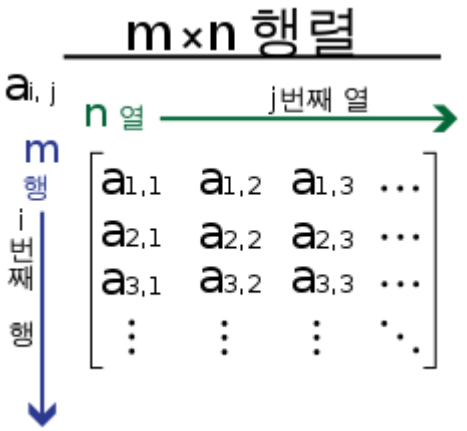
참고 사이트

- http://taewan.kim/post/numpy_cheat_sheet/
- https://rfriend.tistory.com/290
- https://cs231n.github.io/python-numpy-tutorial/
- https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Numpy_Python_Cheat_Sheet.pdf
- https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/Python4_DataAnalysis.html
- http://jalammar.github.io/visual-numpy/
- https://www.plus2net.com/python/numpy-ndarray-result.php
- https://deepage.net/features/numpy-axis.html

스칼라, 벡터, 행렬, 텐서

- **넓은 의미로 자료의 모임이 텐서(tensor)**
  - 작은 의미로 특히 3차원 이상 배열을 텐서(tensor)라고도 부름
    - **[[[1, 2], [3, 4]], [[5, 6], [7, 8]]]**
- **스칼라**
  - 스칼라는 하나의 숫자만으로 이루어진 데이터를 의미
    - **3**
- **벡터**
  - 여러 숫자가 순서대로 모여 있는 것으로, 일반적인 일차원 배열이 벡터
    - **[1, 2, 3, 4]**
- **행렬**
  - 복수의 차원을 가지는 데이터가 다시 여러 개 있는 경우의 데이터를 합쳐서 표기한 것
  - 일반적으로 2차원 배열이 행렬
    - **[[1, 2], [3, 4]]**

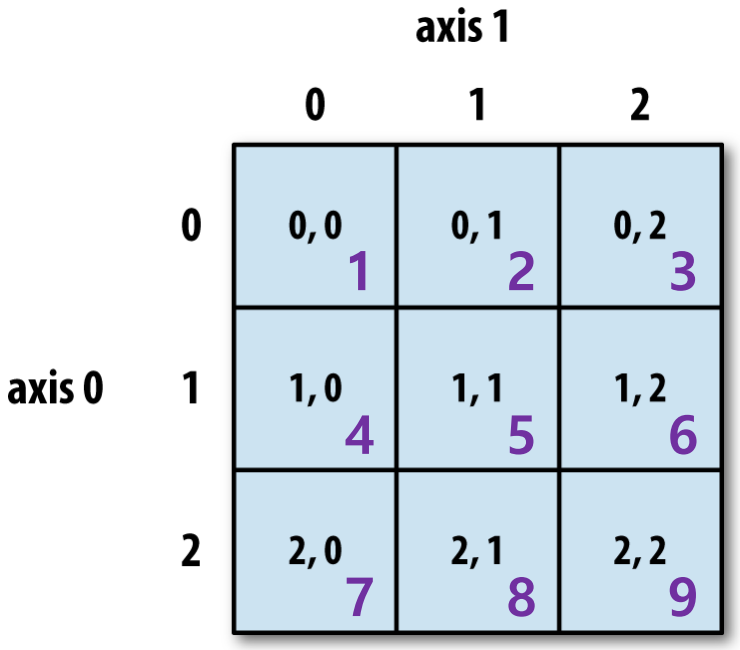
$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$



2차원 행렬의 축과 첨자

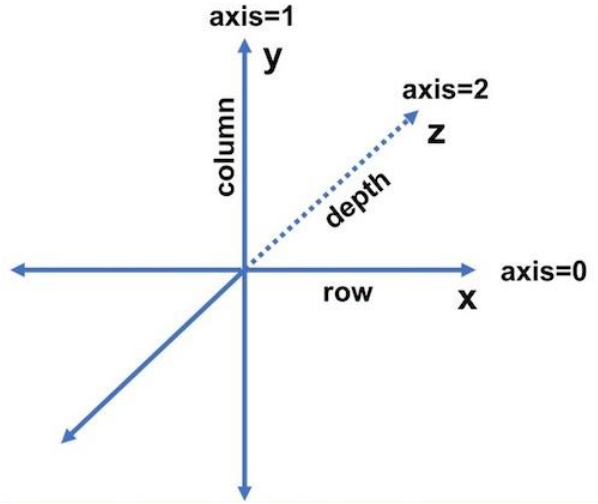
```
In [62]: arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

In [63]: arr2d[2]
Out[63]: array([7, 8, 9])
```

```
In [64]: arr2d[0][2]
Out[64]: 3

In [65]: arr2d[0, 2]
Out[65]: 3
```



다차원 배열

- **자료형 ndarray 제공**
  - 다차원 배열의 데이터 방향을 axis로 표현
  - 각각 axis=0, axis=1 그리고 axis=2로 지정
    - **행방향(높이), 열방향(폭), 깊이(채널 방향)이라는 표현**



3차원 행렬, 텐서

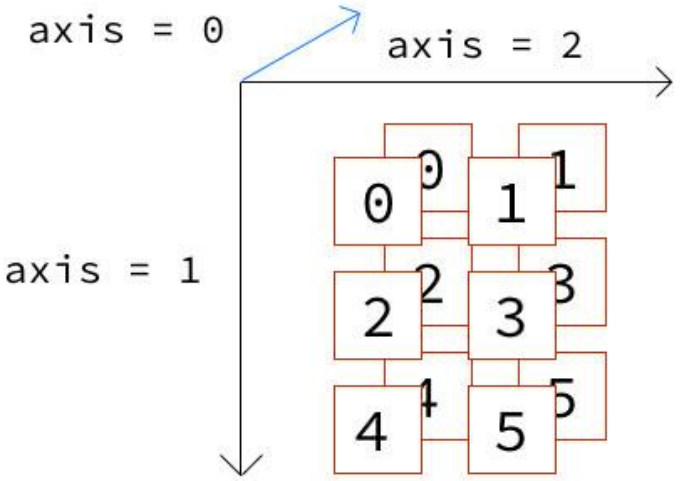
```
In [26]: a = np.arange(6).reshape(3, 2)
         a

Out[26]: array([[0, 1],
                [2, 3],
                [4, 5]])
```

```
In [27]: m = np.array([a, a])
         m

Out[27]: array([[[0, 1],
                 [2, 3],
                 [4, 5]],

                [[0, 1],
                 [2, 3],
                 [4, 5]]])
```

2 축

1 축

0 축




```
In [28]: m.shape

Out[28]: (2, 3, 2)
```

실습

- **교재 파일**
  - ch04.ipynb
  - Ch04-study.ipynb로 복사해서 연습
- **난수와 실수의 정확도**
  - import numpy as np
  - np.random.seed(12345)
    - **난수를 발생시키기 위한 초기 값 지정**
      - **이후 난수가 동일하게 발생**
  - np.set_printoptions(precision=4, suppress=True)
    - **precision=4: 소수점 이하 반올림해 4개 표시**
    - **np.array(3.123456)**
      - **array(3.1235)**
    - **suppress=True: e-04와 같은 scientific notation을 억제하고 싶으면**

- **Alt + Enter**
  - 현재 셀 실행 후, 다음 셀 삽입
- **Ctrl + shift + enter**
  - 셀 분리

4.1 ndarray: 다차원 배열 객체

- **대용량 데이터를 다루는 유연한 자료 구조**
  - 같은 종류의 데이터를 저장하는 포괄적 자료구조
    - **모든 원소는 같은 자료형**
  - 표준 파이썬의 리스트와 다름
- **주요 속성**
  - ndim: 차원 또는 차수
  - shape: 구조
    - **(3,)**
    - **(3, 2)**
    - **(4, 2, 3)**
  - dtype: 원소의 자료형
- **ndarray 생성**
  - np.array(다른 배열이나 순차적인 데이터)
  - np.zeros(), np.ones()
    - **np.zeros(10), np.zeros((2, 3))**
  - np.arange(): range()의 인자로 1차원 배열을 생성
    - **np.arange(10)**

다양한 배열 생성 함수

- 자료형을 명시하지 않으면 float64(부동소수)

Table 4-1. Array creation functions

Function	Description
`array`	Convert input data (list, tuple, array, or other sequence type) to an ndarray either by inferring a dtype or explicitly specifying a dtype; copies the input data by default
`asarray`	Convert input to ndarray, but do not copy if the input is already an ndarray
`arange`	Like the built-in `range` but returns an ndarray instead of a list
`ones`, `ones_like`	Produce an array of all 1s with the given shape and dtype; `ones_like` takes another array and produces a ones array of the same shape and dtype
`zeros`, `zeros_like`	Like `ones` and `ones_like` but producing arrays of 0s instead
`empty`, `empty_like`	Create new arrays by allocating new memory, but do not populate with any values like `ones` and `zeros`
`full`, `full_like`	Produce an array of the given shape and dtype with all values set to the indicated "fill value" `full_like` takes another array and produces a filled array of the same shape and dtype
`eye, identity`	Create a square N × N identity matrix (1s on the diagonal and 0s elsewhere)

배열 생성 초기화 함수(1)

- **np.zeros 함수**
  - zeros(shape, dtype=float, order='C')
  - 지정된 shape의 배열을 생성하고, 모든 요소를 0으로 초기화
    - **order: order in memory.**
      - **'C': row-major (C-style)**
      - **'F': column-major (Fortran-style)**
- **np.ones 함수**
  - np.ones(shape, dtype=None, order='C')
  - 지정된 shape의 배열을 생성하고, 모든 요소를 1로 초기화
- **np.full 함수**
  - np.full(shape, fill_value, dtype=None, order='C')
  - 지정된 shape의 배열을 생성하고, 모든 요소를 지정한 "fill_value"로 초기화
- **np.eye 함수**
  - np.eye(N, M=None, k=0, dtype=<class 'float'>)
  - (N, N) shape의 단위 행렬(Unit Matrix)을 생성

배열 생성 초기화 함수(2)

- **np.empty 함수**
  - empty(shape, dtype=float, order='C')
  - 지정된 shape의 배열 생성
  - 요소의 초기화 과정에 없고, 기존 메모리값을 그대로 사용
  - 배열 생성 비용이 가장 저렴하고 빠름
  - 배열 사용 시 주의가 필요(초기화를 고려)
- **like 함수**
  - numpy는 지정된 배열과 shape이 같은 행렬을 만드는 like 함수를 제공
  - np.zeros_like
  - np.ones_like
  - np.full_like
  - np.empty_like

dtype

- **dtype 객체**
  - 빠른 메모리 참조를 위해 필요한 정보(메타데이터)를 담는 객체
- **다양한 자료형**
  - 정수, 실수, 논리, 객체, 문자열 등 구분

Table 4-2. NumPy data types

Type	Type code	Description
`int8, uint8`	`i1, u1`	Signed and unsigned 8-bit (1 byte) integer types
`int16, uint16`	`i2, u2`	Signed and unsigned 16-bit integer types
`int32, uint32`	`i4, u4`	Signed and unsigned 32-bit integer types
`int64, uint64`	`i8, u8`	Signed and unsigned 64-bit integer types
`float16`	`f2`	Half-precision floating point
`float32`	`f4 or f`	Standard single-precision floating point; compatible with C float
`float64`	`f8 or d`	Standard double-precision floating point; compatible with C double and Python `float` object
`float128`	`f16 or g`	Extended-precision floating point
`complex64, complex128, complex256`	`c8, c16, c32`	Complex numbers represented by two 32, 64, or 128 floats, respectively
`bool`	`?`	Boolean type storing `True` and `False` values
`object`	`0`	Python object type; a value can be any Python object
`string_`	`S`	Fixed-length ASCII string type (1 byte per character); for example, to create a string dtype with length 10, use `'S10'`
`unicode_`	`U`	Fixed-length Unicode type (number of bytes platform specific); same specification semantics as `string_` (e.g., `'U10'`)

astype() 함수

- **arr.astype(자료형)**
  - 배열 arr에 지정된 자료형으로 변환(casting)하여 새로운 배열을 복사하여 반환
  - 형 변환 실패시
    - **ValueError**

```
In [89]: arr
Out[89]: array([ 3.7, -1.2, -2.6,  0.5, 12.9, 10.1])

In [88]: arr = np.array([3.7, -1.2, -2.6, 0.5, 12.9, 10.1])
         arr
         arr.astype(np.int32)
Out[88]: array([ 3, -1, -2,  0, 12, 10])

In [113]: numeric_strings = np.array(['1.25', '-9.6', '42'], dtype=np.string_)
          numeric_strings.astype(float)
Out[113]: array([ 1.25, -9.6 , 42.  ])

In [114]: numeric_strings = np.array(['1.25', '-9.6', '42f'], dtype=np.string_)
          numeric_strings.astype(float)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-114-efa6b388ed44> in <module>
      1 numeric_strings = np.array(['1.25', '-9.6', '42f'], dtype=np.string_)
----> 2 numeric_strings.astype(float)

ValueError: could not convert string to float: '42f'
```