



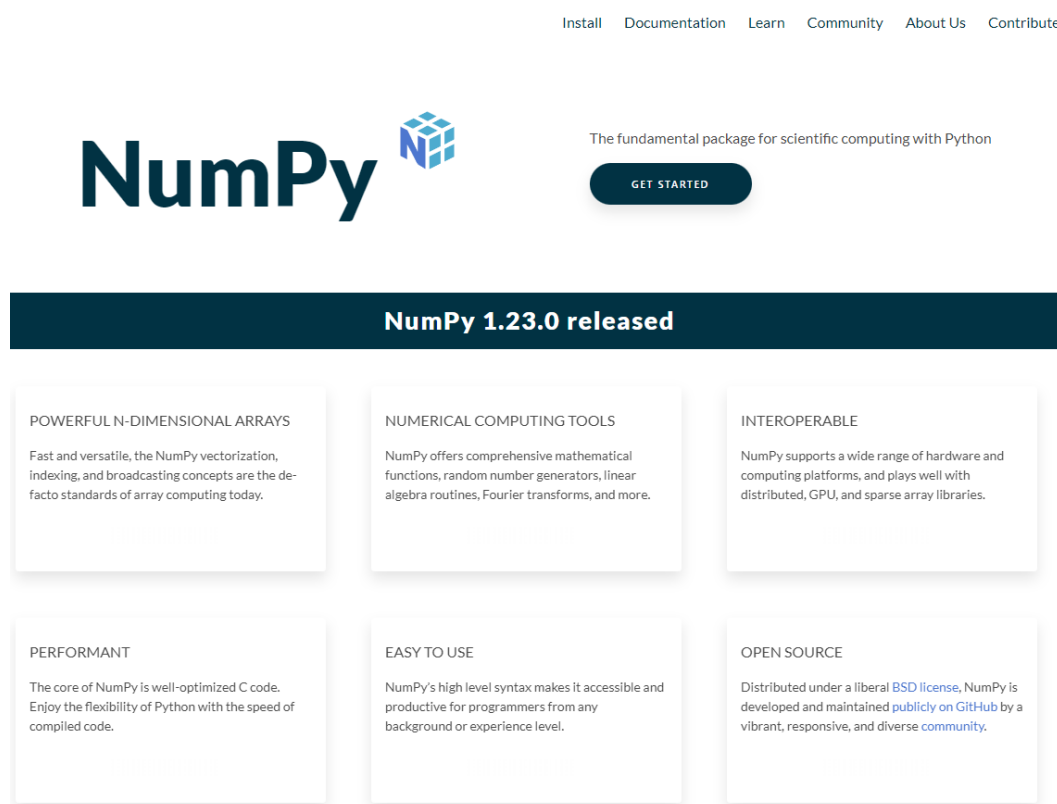
의료인공지능

파이썬 - Numpy, pandas, matplotlib

고려대학교 의료빅데이터연구소
채민수(minsuchae@korea.ac.kr)

1. numpy

- 넘파이 : 기존 리스트를 개선하여 2차원 이상 배열을 사용하기 위한 오픈소스 라이브러리
 - 2차원 이상 배열 지원
 - 행렬끼리 연산 지원
 - 행렬과 스칼라 연산 지원
 - 다차원 슬라이싱



The screenshot shows the NumPy website. At the top, there's a navigation bar with links: Install, Documentation, Learn, Community, About Us, and Contribute. The main header features the NumPy logo (a blue cube) and the text "The fundamental package for scientific computing with Python". Below this is a dark blue button labeled "GET STARTED". A dark blue banner below the header says "NumPy 1.23.0 released". The main content area is divided into six white boxes with dark blue borders, each describing a feature of NumPy: 1. POWERFUL N-DIMENSIONAL ARRAYS: Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today. 2. NUMERICAL COMPUTING TOOLS: NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more. 3. INTEROPERABLE: NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries. 4. PERFORMANT: The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code. 5. EASY TO USE: NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level. 6. OPEN SOURCE: Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

<https://numpy.org>

1. numpy

- 넘파이 설치
 - Anaconda
 - conda install numpy
 - Python
 - pip install numpy
- Jupyter Notebook에서의 설치

```
!conda install numpy
```

```
!pip install numpy
```

```
1 !pip install numpy
```

```
Collecting numpy
  Using cached numpy-1.23.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
Installing collected packages: numpy
  WARNING: The scripts f2py, f2py3 and f2py3.8 are installed in '/home/minsuchae/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed numpy-1.23.3
```

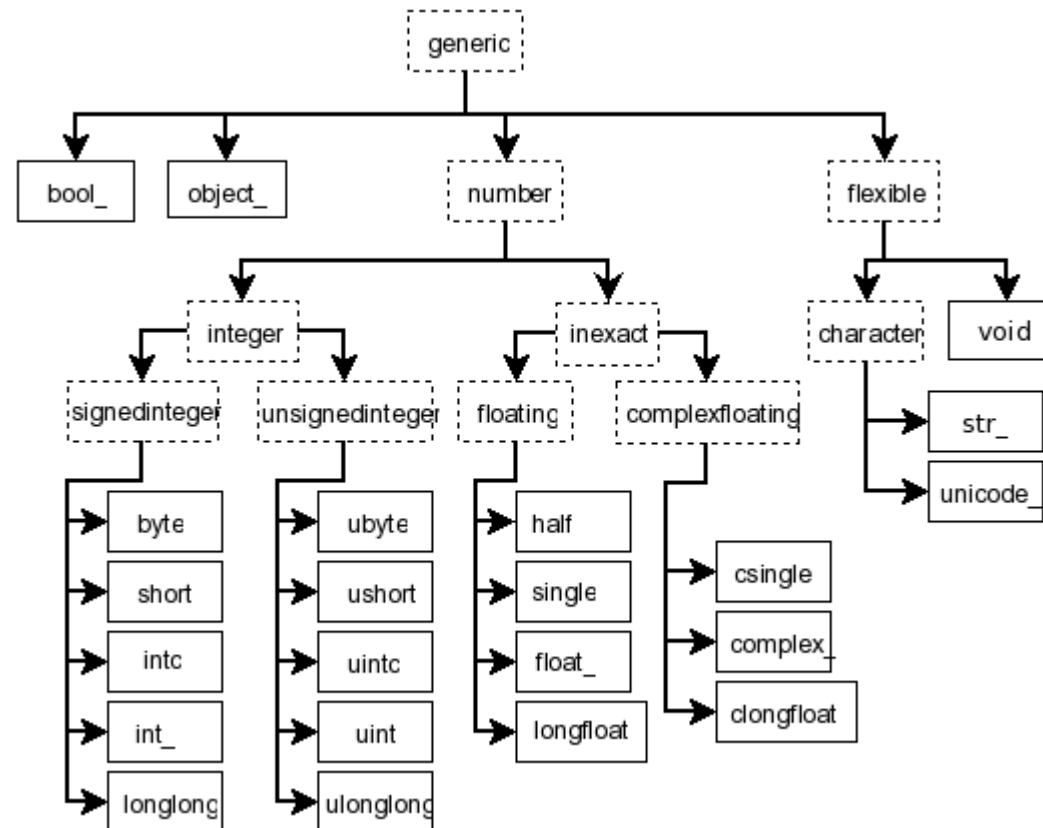
1. numpy

- 넘파이 배열 생성

- numpy.array 를 통해 리스트나 튜플 등의 값을 통해 생성
- pandas의 데이터프레임에서의 변환
- 그 외 함수를 통해 생성
 - numpy.empty
 - numpy.zeros
 - numpy.ones
 - numpy.full
 - numpy.eyes
 - copy
 - fromfunction

1. numpy

- 넘파이 데이터 타입



<https://numpy.org/doc/stable/reference/arrays.scalars.html>

1. numpy

- 넘파이 배열 생성
 - numpy.array 를 통한 생성

numpy.array

```
numpy.array(object, dtype=None, *, copy=True, order='K', subok=False, ndmin=0,  
            like=None)
```

Create an array.

Parameters: object : *array_like*

An array, any object exposing the array interface, an object whose `__array__` method returns an array, or any (nested) sequence. If object is a scalar, a 0-dimensional array containing object is returned.

dtype : *data-type, optional*

The desired data-type for the array. If not given, then the type will be determined as the minimum type required to hold the objects in the sequence.

<https://numpy.org/doc/stable/reference/generated/numpy.array.html>

1. numpy

- 넘파이 배열 생성

- numpy.array 를 통한 생성

```
import numpy as np
```

```
my_list = [1,2,3,4,5]  
my_array = np.array(my_list)  
print(my_array)
```

```
[실행결과]  
[1 2 3 4 5]
```

```
my_list2 = [[1,2],[3,4]]  
my_array2 = np.array(my_list2)  
print(my_array2)
```

```
[실행결과]  
[[1 2]  
 [3 4]]
```

1. numpy

- 넘파이 배열 생성
 - numpy.empty 를 통한 생성

```
numpy.empty(shape, dtype=float, order='C', *, like=None) #
```

Return a new array of given shape and type, without initializing entries.

Parameters: shape : *int or tuple of int*

Shape of the empty array, e.g., (2, 3) or 2.

dtype : *data-type, optional*

Desired output data-type for the array, e.g, `numpy.int8`. Default is `numpy.float64`.

<https://numpy.org/doc/stable/reference/generated/numpy.empty.html>

1. numpy

- 넘파이 배열 생성

- numpy.zeros 를 통한 생성

numpy.zeros

`numpy.zeros(shape, dtype=float, order='C', *, like=None)`

Return a new array of given shape and type, filled with zeros.

Parameters: *shape* : *int or tuple of ints*

Shape of the new array, e.g., (2, 3) or 2.

dtype : *data-type, optional*

The desired data-type for the array, e.g., `numpy.int8`. Default is `numpy.float64`.

<https://numpy.org/doc/stable/reference/generated/numpy.zeros.html>

1. numpy

- 넘파이 배열 생성

- numpy.empty 를 통한 생성

```
my_array3 = np.empty((2,3))  
print(my_array3)
```

[실행결과]

```
[[9.437e-322 9.486e-322 1.892e-321]  
 [1.897e-321 2.841e-321 2.846e-321]]
```

- numpy.zeros 를 통한 생성

```
my_array4 = np.zeros((3,3))  
print(my_array4)
```

[실행결과]

```
[[0. 0. 0.]  
 [0. 0. 0.]  
 [0. 0. 0.]]
```

1. numpy

- 넘파이 배열 생성

- numpy.xxxx_like 를 통한 생성

- 기존 함수와 동일한 차원을 갖는 넘파이 배열 생성 시 사용

- 지원하는 함수

- empty_like

- ones_like

- zeros_like

- full_like

```
my_array5 = np.zeros_like(my_array3)
print(my_array5)
```

[실행결과]

[[0. 0. 0.]

[0. 0. 0.]]

1. numpy

◦ 넘파이 배열 생성

- numpy.arange 를 통한 순회(반복문)
numpy.arange

`numpy.arange([start,]stop, [step,]dtype=None, *, Like=None)`

Return evenly spaced values within a given interval.

arange can be called with a varying number of positional arguments:

- **arange(stop)**: Values are generated within the half-open interval `[0, stop)` (in other words, the interval including `start` but excluding `stop`).
- **arange(start, stop)**: Values are generated within the half-open interval `[start, stop)`.
- **arange(start, stop, step)** Values are generated within the half-open interval `[start, stop)`, with spacing between values given by `step`.

For integer arguments the function is roughly equivalent to the Python built-in **range**, but returns an ndarray rather than a **range** instance.

When using a non-integer step, such as 0.1, it is often better to use **numpy.linspace**.

<https://numpy.org/doc/stable/reference/generated/numpy.arange.html>

1. numpy

- 넘파이 배열 생성
 - numpy.arange 를 통한 순회(반복문)

```
print(np.arange(0,5))
```

[실행결과]
[0 1 2 3 4]

```
print(np.arange(0,5, 0.5))
```

[실행결과]
[0. 0.5 1. 1.5 2. 2.5 3. 3.5 4. 4.5]

1. numpy

◦ 넘파이 배열 생성

- numpy.linspace 를 통한 선형적인 데이터 생성

numpy.linspace

`numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)`

Return evenly spaced numbers over a specified interval.

[\[source\]](#)

Returns *num* evenly spaced samples, calculated over the interval *[start, stop]*.

The endpoint of the interval can optionally be excluded.

Changed in version 1.16.0: Non-scalar *start* and *stop* are now supported.

Changed in version 1.20.0: Values are rounded towards *-inf* instead of *0* when an integer *dtype* is specified. The old behavior can still be obtained with `np.linspace(start, stop, num).astype(int)`

Parameters: *start* : *array_like*

The starting value of the sequence.

stop : *array_like*

The end value of the sequence, unless *endpoint* is set to False. In that case, the sequence consists of all but the last of *num* + 1 evenly spaced samples, so that *stop* is excluded. Note that the step size changes when *endpoint* is False.

num : *int, optional*

Number of samples to generate. Default is 50. Must be non-negative.

<https://numpy.org/doc/stable/reference/generated/numpy.linspace.html>

1. numpy

- 넘파이 배열 생성

- numpy.linspace 를 통한 선형적인 데이터 생성

```
print(np.linspace(1,100,10))
```

[실행결과]

```
[ 1. 12. 23. 34. 45. 56. 67. 78. 89. 100.]
```

```
print(np.linspace(10,11,20))
```

[실행결과]

```
[10.      10.05263158 10.10526316 10.15789474 10.21052632 10.26315789
 10.31578947 10.36842105 10.42105263 10.47368421 10.52631579 10.57894737
 10.63157895 10.68421053 10.73684211 10.78947368 10.84210526 10.89473684
 10.94736842 11.      ]
```

1. numpy

- 넘파이 배열의 차원

- ndarray로 처리로 배열의 차원을 내부적으로 저장
- 1-dimension
 - e.g.; (10,)
- 2-dimension : 테이블 형태를 갖는 데이터
 - e.g.; (2,5)
- 3-dimension : 시계열 데이터, 이미지
 - e.g.; (1,3,5)

1. numpy

- 넘파이 배열의 차원 변경 및 확인
 - shape : 배열의 차원 확인
 - reshape : 배열의 차원을 변경

```
arr = np.linspace(1,100,10)
```

```
print(arr.shape)
```

```
print(arr.reshape(2,5))
```

[실행결과]

```
(10,)
```

```
[[ 1. 12. 23. 34. 45.]
```

```
 [56. 67. 78. 89. 100.]]
```

```
print(arr.reshape(-1,2))
```

[실행결과]

```
[[ 1. 12.]
```

```
 [23. 34.]
```

```
 [45. 56.]
```

```
 [67. 78.]
```

```
 [89. 100.]]
```

```
print(arr.reshape(3,4))
```

[실행결과]

ValueError

Traceback (most recent call last)

<ipython-input-64-28cf0bbabd3a> in <module>

----> 1 print(arr.reshape(3,4))

ValueError: cannot reshape array of size 10 into shape (3,4)

1. numpy

- 넘파이 배열 연산

- 지원하는 연산자

- +

- -

- *

- **

- /

- //

- array 연산자 array

- array 연산자 scalar

1. numpy

◦ 넘파이 배열 연산 실습

```
print(my_array5 + 5)
```

[실행결과]

```
[[5. 5. 5.]
```

```
 [5. 5. 5.]]
```

```
my_array6=np.array([[1,2,3],[4,5,6],[7,8,9]])  
print(my_array6 + my_array6)
```

[실행결과]

```
[[ 2  4  6]
```

```
 [ 8 10 12]
```

```
 [14 16 18]]
```

```
print(my_array6 * my_array6)
```

[실행결과]

```
[[ 1  4  9]
```

```
 [16 25 36]
```

```
 [49 64 81]]
```

```
print(my_array6 - my_array6)
```

[실행결과]

```
[[0 0 0]
```

```
 [0 0 0]
```

```
 [0 0 0]]
```

```
print(my_array2 * my_array6)
```

[실행결과]



1. numpy

- 넘파이 배열이 지원하는 함수

- ndarray.min
- ndarray.max
- ndarray.mean
- ndarray.median
- ndarray.size
- ndarray.sum

```
my_array2.sum()
```

[실행결과]

10

1. numpy

◦ 넘파이가 지원하는 고급 수학 연산

- numpy.cos - numpy.radians - numpy.log2 - numpy.arcsin - numpy.tanh
- numpy.sin - numpy.abs - numpy.log10 - numpy.arctan - numpy.pi
- numpy.tan - numpy.abs - numpy.sqrt - numpy.cosh - numpy.e
- numpy.exp - numpy.log - numpy.arccos - numpy.sinh

<https://numpy.org/doc/stable/reference/routines.math.html>

```
print(np.linspace(0,np.pi,10))
```

[실행결과]

```
[0.         0.34906585 0.6981317  1.04719755 1.3962634  1.74532925
 2.0943951  2.44346095 2.7925268  3.14159265]
```

```
print(np.cos(np.linspace(0,np.pi,10)))
```

[실행결과]

```
[ 1.         0.93969262 0.76604444 0.5         0.17364818 -0.17364818
 -0.5        -0.76604444 -0.93969262 -1.         ]
```

1. numpy

- 넘파이 슬라이싱
 - 기존 파이썬과 달리 다중 슬라이싱 지원

```
print(my_array6)
```

```
[실행결과]  
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

```
print(my_array6[:,1])
```

```
[실행결과]  
[2 5 8]
```

```
print(my_array6[:,1])
```

```
[실행결과]  
[4 5 6]
```

```
print(my_array6[:])
```

```
[실행결과]  
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

1. numpy

- 넘파이 슬라이싱
 - 기존 파이썬과 달리 다중 슬라이싱 지원

```
print(my_array6[:,(0,2)])
```

```
[실행결과]  
[[1 3]  
 [4 6]  
 [7 9]]
```

```
print(my_array6[:2,(0,2)])
```

```
[실행결과]  
[[1 3]  
 [4 6]]
```

```
print(my_array6[(0,2)])
```

```
[실행결과]  
3
```

```
print(my_array6[(0,2),(0,2)])
```

```
[실행결과]  
[1 9]
```

1. numpy

- 넘파이 마스킹
 - 조건이 만족하는 값만 출력

```
print(my_array6)
```

```
[실행결과]  
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

```
print(my_array6 > 3)
```

```
[실행결과]  
[[False False False]  
 [ True  True  True]  
 [ True  True  True]]
```

```
print(my_array6[my_array6 > 3])
```

```
[실행결과]  
[4 5 6 7 8 9]
```

```
print(my_array6[(my_array6 > 3) & (my_array6 < 6)])
```

```
[실행결과]  
[4 5]
```


1. numpy

- 넘파이 마스킹

- 고급 마스킹

➤ `tril_indices` : 넘파이 배열 기준으로 왼쪽 하단의 삼각형 영역의 인덱스를 반환

```
print(my_array6)
```

[실행결과]

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
print(my_array6[np.tril_indices(my_array6.shape[0])])
```

[실행결과]

```
[1 4 5 7 8 9]
```

```
print(np.tril_indices(my_array6.shape[0]))
```

[실행결과]

```
(array([0, 1, 1, 2, 2, 2]), array([0, 0, 1, 0, 1, 2]))
```

```
my_array6[np.tril_indices(my_array6.shape[0])] = 0
print(my_array6)
```

[실행결과]

```
[[0 2 3]
 [0 0 6]
 [0 0 0]]
```

1. numpy

- 넘파이 마스킹

- 고급 마스킹

➤ triu_indices : 넘파이 배열 기준으로 오른쪽 상단의 삼각형 영역의 인덱스를 반환

```
my_array6=np.array([[1,2,3],[4,5,6],[7,8,9]])
```

[실행결과]

```
print(np.triu_indices(my_array6.shape[0]))
```

[실행결과]

```
(array([0, 1, 1, 2, 2, 2]), array([0, 0, 1, 0, 1, 2]))
```

```
print(my_array6[np.triu_indices(my_array6.shape[0])])
```

[실행결과]

```
[1 2 3 5 6 9]
```

```
my_array6[np.triu_indices(my_array6.shape[0])] = 0
```

```
print(my_array6)
```

[실행결과]

```
[[0 0 0]
```

```
 [4 0 0]
```

```
 [7 8 0]]
```

1. numpy

- 넘파이 마스킹

- 고급 마스킹

- tril_indices_from
 - triu_indices_from
 - shape를 입력받지 않고 넘파이 배열을 입력받도록 개선

```
my_array6=np.array([[1,2,3],[4,5,6],[7,8,9]])  
my_array6[np.triu_indices_from(my_array6)] = 0  
print(my_array6)
```

[실행결과]

```
[[0 0 0]  
 [4 0 0]  
 [7 8 0]]
```

2. pandas

◦ 판다스 설치

- Anaconda

➤ conda install pandas

- Python

➤ pip install pandas

◦ Jupyter Notebook에서의 설치

```
!conda install pandas
```

```
!pip install pandas
```

```
1 !pip install pandas
```

```
Collecting pandas
  Downloading pandas-1.5.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.2 MB)
    |████████████████████| 12.2 MB 242 kB/s eta 0:00:01
Collecting numpy>=1.20.3: python_version < "3.10"
  Downloading numpy-1.23.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
    |████████████████████| 17.1 MB 151 kB/s eta 0:00:01
Collecting pytz>=2020.1
  Using cached pytz-2022.2.1-py2.py3-none-any.whl (500 kB)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/minsuchae/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from python-dateutil>=2.8.1->pandas) (1.14.0)
Installing collected packages: numpy, pytz, pandas
Successfully installed numpy-1.23.3 pandas-1.5.0 pytz-2022.2.1
```

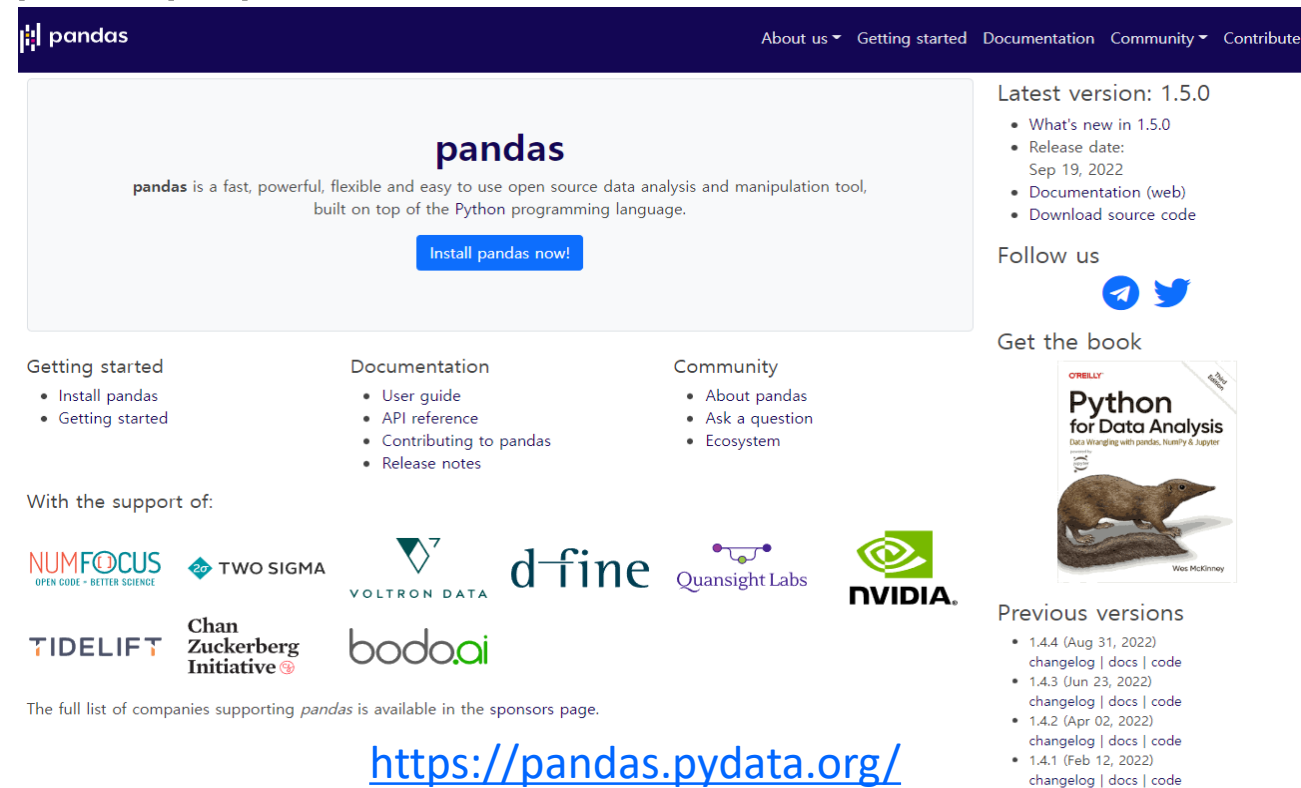
2. pandas

- 판다스 : 데이터 구조를 사용하여 고성능 데이터 조작 및 데이터 분석에 사용되는 오픈소스 라이브러리

- 데이터셋 로드

- 결측치 처리

- 데이터 분석



The screenshot shows the pandas website homepage. The header includes the pandas logo and navigation links: About us, Getting started, Documentation, Community, and Contribute. The main content area features the pandas logo, a description of pandas as a fast, powerful, flexible, and easy-to-use open source data analysis and manipulation tool built on top of the Python programming language, and a prominent blue button labeled "Install pandas now!". To the right, it lists the latest version as 1.5.0, with details on what's new, release date (Sep 19, 2022), documentation, and source code download. Below this, there are social media links for GitHub and Twitter. The "Get the book" section promotes the O'Reilly book "Python for Data Analysis" by Wes McKinney. The "Getting started" section lists links for installing pandas and getting started. The "Documentation" section lists links for the user guide, API reference, contributing to pandas, and release notes. The "Community" section lists links for about pandas, asking a question, and the ecosystem. A "With the support of:" section displays logos of various sponsors including NUMFOCUS, TWO SIGMA, VOLTRON DATA, d-fine, Quansight Labs, NVIDIA, TIDELIFT, and Chan Zuckerberg Initiative. At the bottom, it states that the full list of companies supporting pandas is available in the sponsors page, followed by the URL <https://pandas.pydata.org/>. On the far right, a "Previous versions" section lists links for versions 1.4.4, 1.4.3, 1.4.2, and 1.4.1, each with links to changelog, docs, and code.

2. pandas

◦ 시리즈 객체

pandas.Series

```
class pandas.Series(data=None, index=None, dtype=None, name=None, copy=False,  
fastpath=False) [source]
```

One-dimensional ndarray with axis labels (including time series).

<https://pandas.pydata.org/docs/reference/api/pandas.Series.html>

- 일반적으로 하나의 특징(feature)를 포함하는 형태
- 인덱스 : 시리즈 간의 구분을 위한 값
 - 숫자가 아닌 문자열이나 다른 자료형으로도 가능
 - 인덱스 값의 중복을 허용
- 데이터 : 값을 저장하는 요소
- 데이터 타입 : 데이터를 저장하는 타입
 - 데이터의 타입이 한 가지일 경우 데이터 타입을 추론

인덱스(index) 값(values)

A	→	5
B	→	6
C	→	12
D	→	-5
E	→	6

2. pandas

- 데이터프레임 객체

pandas.DataFrame

```
class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=None)
```

[source]

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html>

- 테이블 형태를 갖는 객체
- ndarray나 dictionary 나 파일로부터 변환이 가능

		열(column)		
		age	Blood types	Gender
		↓	↓	↓
인덱스(index)	A	10	a	Man
	B	11	b	Female
	C	15	o	Female
	D	39	ab	Man
	E	45	a	Man

2. pandas

- 데이터프레임 생성 실습
 - Dictionary를 통한 생성

```
import pandas as pd
raw_data = {'first_name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
            'last_name': ['Miller', 'Jacobson', 'Ali', 'Milner', 'Cooze'],
            'age': [42, 52, 36, 24, 73],
            'city': ['San Francisco', 'Baltimore', 'Miami', 'Douglas', 'Boston']}
```

```
df = pd.DataFrame(raw_data, columns = ['first_name', 'last_name', 'age', 'city'])
df.head()
```

[실행결과]

	first_name	last_name	age	city
0	Jason	Miller	42	San Francisco
1	Molly	Jacobson	52	Baltimore
2	Tina	Ali	36	Miami
3	Jake	Milner	24	Douglas
4	Amy	Cooze	73	Boston

2. pandas

- 데이터프레임 생성 실습
 - read_csv 함수를 통한 데이터 로드

```
data_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data'  
df = pd.read_csv(data_url, sep='\s+')  
df.head()
```

[실행결과]

	0.00632	18.00	2.310	0	0.5380	6.5750	65.20	4.0900	1	296.0	15.30	396.90	4.98	24.00
0	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
1	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
2	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
3	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
4	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222.0	18.7	394.12	5.21	28.7

https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

2. pandas

◦ 데이터프레임 생성 실습

- read_csv 함수를 통한 데이터 로드 - header = None

```
data_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data'  
df = pd.read_csv(data_url, sep='\s+', header=None)  
df.head()
```

[실행결과]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

2. pandas

◦ 데이터프레임 생성 실습

- read_csv 함수를 통한 데이터 로드 - header 설정

7. Attribute Information:

1. CRIM	per capita crime rate by town
2. ZN	proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS	proportion of non-retail business acres per town
4. CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX	nitric oxides concentration (parts per 10 million)
6. RM	average number of rooms per dwelling
7. AGE	proportion of owner-occupied units built prior to 1940
8. DIS	weighted distances to five Boston employment centres
9. RAD	index of accessibility to radial highways
10. TAX	full-value property-tax rate per \$10,000
11. PTRATIO	pupil-teacher ratio by town
12. B	$1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town
13. LSTAT	% lower status of the population
14. MEDV	Median value of owner-occupied homes in \$1000's

<https://archive.ics.uci.edu/ml/machine-learning-databases/housing/>

2. pandas

◦ 데이터프레임 생성 실습

- read_csv 함수를 통한 데이터 로드 - header 설정

```
data_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data'  
df=pd.read_csv(data_url, sep='\s+', names=['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'STAT', 'MEDV'])  
df.head()
```

[실행결과]

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	STAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

<https://archive.ics.uci.edu/ml/machine-learning-databases/housing/>

2. pandas

- 데이터프레임 생성 실습

- read_xxx 함수를 통한 데이터 로드

- read_csv : csv 파일
 - read_pickle : 판다스 객체를 pickle을 통해 직렬화 시킨 파일
 - read_table : 탭문자로 구분되어 있는 파일
 - read_excel : 엑셀 파일
 - read_json : JSON 파일
 - read_xml : xml 파일
 - read_sas : SAS 파일
 - read_spss : SPSS 파일
 - read_sql : SQL 통해서 얻은 결과

2. pandas

- 데이터프레임 생성 실습

- read_excel 함수 - xlsx

- openpyxl 라이브러리 설치 필요

```
!conda install --y openpyxl #아나콘다를 사용하는 경우 설치
```

```
!pip install --y openpyxl #아나콘다를 사용하지 않는 경우 설치
```

2. pandas

- 데이터프레임 함수 실습

- head : 데이터프레임에서 첫 n개 행 추출. default : 5

```
df.head()
```

[실행결과]

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	STAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

2. pandas

- 데이터프레임 함수 실습

- head : 데이터프레임에서 첫 n개 행 추출. default : 5

```
df.head(3)
```

[실행결과]

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	STAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7

2. pandas

- 데이터프레임 함수 실습
 - tail : 데이터프레임에서 끝 n개 행 추출. default : 5

df.tail()

[실행결과]

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	STAT	MEDV
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

2. pandas

- 데이터프레임 함수 실습
 - tail : 데이터프레임에서 끝 n개 행 추출. default : 5

```
df.tail(3)
```

[실행결과]

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	STAT	MEDV
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

2. pandas

- 데이터프레임 함수 실습
 - drop : 특정 행 혹은 특정 열 삭제

```
df.drop(0)  
df.head()
```

[실행결과]

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	STAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2



2. pandas

◦ 데이터프레임 함수 실습

- drop : 특정 행 혹은 특정 열 삭제

➤ drop 결과를 다른 변수에 저장 혹은 inplace 인자를 True로 설정

```
df_drop_0_index = df.drop(0)
df_drop_0_index.head()
```

[실행결과]

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	STAT	MEDV
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222.0	18.7	394.12	5.21	28.7

2. pandas

◦ 데이터프레임 함수 실습

- drop : 특정 행 혹은 특정 열 삭제

➤ axis 기본 값은 0이며 0은 행, 1은 열을 삭제함

```
df_drop_CRIM_column_axis_1 = df.drop('CRIM',axis=1)  
df_drop_CRIM_column_axis_1.head()
```

[실행결과]

	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	STAT	MEDV
0	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

2. pandas

- 데이터프레임 함수 실습
 - reset_index : 인덱스 리셋

```
df_drop_0_index.reset_index(inplace=True)  
df_drop_0_index.head()
```

[실행결과]

	index	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	STAT	MEDV
0	1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
1	2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
2	3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
3	4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
4	5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222.0	18.7	394.12	5.21	28.7

2. pandas

- 데이터프레임 함수 실습
 - groupby : 그룹핑하여 통계 수행

```
df = pd.read_csv('Pima_Indians_Diabetes_Database.csv')  
df.head()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

2. pandas

- 데이터프레임 함수 실습
 - groupby : 그룹핑하여 통계 수행

```
print(df.groupby("Age")['BMI'].mean())
```

[실행결과]

Age

21 27.817460

22 29.509722

23 31.502632

24 32.569565

...

2. pandas

- 데이터프레임 함수 실습
 - groupby : 그룹핑하여 통계 수행
 - 다중 키를 통해 그룹핑 가능

```
print(df.groupby(["Age","Outcome"])['Insulin'].median())
```

[실행결과]

Age	Outcome	
21	0	42.5
	1	0.0
22	0	53.0
	1	0.0
23	0	95.0
		...

2. pandas

◦ 데이터프레임 함수 실습

- isnull : 결측치 확인

```
print(df.isnull())
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	#
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
..	
763	False	False	False	False	False	False	
764	False	False	False	False	False	False	
765	False	False	False	False	False	False	
766	False	False	False	False	False	False	
767	False	False	False	False	False	False	

	DiabetesPedigreeFunction	Age	Outcome
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
..
763	False	False	False
764	False	False	False
765	False	False	False
766	False	False	False
767	False	False	False

[768 rows x 9 columns]

2. pandas

- 데이터프레임 함수 실습

- isnull : 결측치 확인

- NaN만 카운트한다는 단점이 존재함

```
print(df.isnull().sum())
```

[실행결과]

Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0
dtype: int64	

2. pandas

- 데이터프레임 함수 실습

- (df[feature] condition expression).sum() : 해당 조건을 만족하는 데이터 수

```
for key in df.keys():  
    print(key,':',(df[key]==0).sum())
```

[실행결과]

Pregnancies : 111

Glucose : 5

BloodPressure : 35

SkinThickness : 227

Insulin : 374

BMI : 11

DiabetesPedigreeFunction : 0

Age : 0

Outcome : 500

2. pandas

- 데이터프레임 함수 실습
 - describe : 데이터 범주를 확인(이상치 확인)

df.describe()

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

2. pandas

◦ 데이터프레임 함수 실습

- loc : 특정 조건을 만족하는 인덱스의 열을 접근하거나 설정

```
keys = df.keys()
keys = keys.drop('Pregnancies')
keys = keys.drop('Outcome')
for key in keys:
    df.loc[df[key]==0, key] = None
df.describe()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	763.000000	733.000000	541.000000	394.000000	757.000000	768.000000	768.000000	768.000000
mean	3.845052	121.686763	72.405184	29.153420	155.548223	32.457464	0.471876	33.240885	0.348958
std	3.369578	30.535641	12.382158	10.476982	118.775855	6.924988	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	64.000000	22.000000	76.250000	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	29.000000	125.000000	32.300000	0.372500	29.000000	0.000000
75%	6.000000	141.000000	80.000000	36.000000	190.000000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

2. pandas

- 데이터프레임 함수 실습

- loc : 특정 조건을 만족하는 인덱스의 칼럼에 값을 설정

```
print(df.isnull().sum())
```

[실행결과]

Pregnancies	0
Glucose	5
BloodPressure	35
SkinThickness	227
Insulin	374
BMI	11
DiabetesPedigreeFunction	0
Age	0
Outcome	0
dtype:	int64

2. pandas

- 데이터프레임 함수 실습
 - loc 함수를 이용한 이상치 제거

for key in keys:

```
tmp_under = df[key].median() - df[key].std()
```

```
tmp_over = df[key].median() + df[key].std()
```

```
df.loc[df[key] < tmp_under, key] = None
```

```
df.loc[df[key] > tmp_over, key] = None
```

```
df.loc[df['Pregnancies'] > df['Pregnancies'].median() + df['Pregnancies'].std(), 'Pregnancies'] = None
```

```
df.describe()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	599.000000	538.000000	535.000000	358.000000	335.000000	518.000000	626.000000	574.000000	768.000000
mean	2.375626	114.676580	71.603738	28.938547	115.826866	32.154440	0.348422	27.484321	0.348958
std	1.898109	16.496697	6.881010	5.650574	54.936925	3.727623	0.163585	5.494620	0.476951
min	0.000000	87.000000	60.000000	19.000000	14.000000	25.400000	0.078000	21.000000	0.000000
25%	1.000000	101.000000	66.000000	24.000000	71.000000	29.000000	0.222250	23.000000	0.000000
50%	2.000000	113.000000	72.000000	29.000000	110.000000	32.400000	0.305000	26.000000	0.000000
75%	4.000000	127.750000	78.000000	33.000000	158.000000	35.000000	0.466000	31.000000	1.000000
max	6.000000	147.000000	84.000000	39.000000	240.000000	39.200000	0.703000	40.000000	1.000000

2. pandas

- 데이터프레임 함수 실습
 - fillna : 결측치 보간

```
df_fillna_0 = df.fillna(0)  
df_fillna_0.head()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	0.0	72.0	35.0	0.0	33.6	0.627	0.0	1
1	1.0	0.0	66.0	29.0	0.0	26.6	0.351	31.0	0
2	0.0	0.0	64.0	0.0	0.0	0.0	0.672	32.0	1
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	0
4	0.0	137.0	0.0	35.0	168.0	0.0	0.000	33.0	1

2. pandas

- 데이터프레임 함수 실습
 - fillna method 인자를 이용한 보간
 - 시계열 데이터에 용이
 - bfill, backfill : 다음에 측정된 값으로 보정
 - pad, ffill : 마지막에 측정된 값으로 보정

2. pandas

- 데이터프레임 함수 실습
 - fillna를 집계하여 보간

```
import copy
```

```
df_copy = copy.deepcopy(df)
```

```
for key in df.keys():
```

```
    df_copy.fillna(df_copy.groupby(["Age", "Outcome"])[key].median(), inplace=True)
```

```
df_copy.head()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	NaN	72.0	35.0	NaN	33.6	0.627	NaN	1
1	1.0	NaN	66.0	29.0	NaN	26.6	0.351	31.0	0
2	NaN	NaN	64.0	NaN	NaN	NaN	0.672	32.0	1
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	0
4	0.0	137.0	NaN	35.0	168.0	NaN	NaN	33.0	1

2. pandas

- 데이터프레임 함수 실습
 - fillna를 이용한 간단한 보간

```
import copy

df_copy = copy.deepcopy(df)
for key in df.keys():
    df_copy.fillna(df_copy[key].mean(), inplace=True)
df_copy.head()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.000000	2.375626	72.000000	35.000000	2.375626	33.600000	0.627000	2.375626	1
1	1.000000	2.375626	66.000000	29.000000	2.375626	26.600000	0.351000	31.000000	0
2	2.375626	2.375626	64.000000	2.375626	2.375626	2.375626	0.672000	32.000000	1
3	1.000000	89.000000	66.000000	23.000000	94.000000	28.100000	0.167000	21.000000	0
4	0.000000	137.000000	2.375626	35.000000	168.000000	2.375626	2.375626	33.000000	1

2. pandas

- 데이터프레임 함수 실습
 - fillna를 집계하여 보간

```
print(df_copy.isnull().sum())
```

[실행결과]

Pregnancies	169
Glucose	230
BloodPressure	233
SkinThickness	410
Insulin	433
BMI	250
DiabetesPedigreeFunction	142
Age	194
Outcome	0
dtype:	int64

2. pandas

◦ 데이터프레임 함수 실습

- dropna : 결측치 항목이 있는 행 혹은 열 삭제
 - axis : 행을 삭제할 지(0) 열을 삭제할지(1) 을 설정함(기본 값 : 0)
 - how : axis 값에 따라 행 혹은 열의 모든 값이 결측치일 경우 삭제(all) 혹은 하나라도 결측치면 삭제(any)할지 설정함(기본값 any)
 - thresh : 임계치 값보다 데이터가 존재하면 삭제하지 않음

2. pandas

- 데이터프레임 함수 실습
 - dropna : 결측치 항목이 있는 행 혹은 열 삭제

```
df_copy.dropna(inplace=True)
print(df_copy.isnull().sum())
```

[실행결과]

Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0
dtype:	int64

2. pandas

- 데이터프레임 함수 실습

- corr : 상관관계 분석

- method : 상관관계 분석 방법(pearson(기본값), kendall, spearman)

```
df_copy.corr()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.357321	0.129704	-0.156891	0.037620	-0.220542	-0.235890	0.563014	0.015717
Glucose	0.357321	1.000000	0.163481	0.023951	0.602439	-0.054317	-0.014512	0.342557	0.288420
BloodPressure	0.129704	0.163481	1.000000	0.039819	0.017484	-0.005411	-0.150449	0.194348	-0.097988
SkinThickness	-0.156891	0.023951	0.039819	1.000000	0.143791	0.565868	0.088636	-0.143122	0.161583
Insulin	0.037620	0.602439	0.017484	0.143791	1.000000	0.215535	0.173258	0.042256	0.169897
BMI	-0.220542	-0.054317	-0.005411	0.565868	0.215535	1.000000	0.033659	-0.147735	0.061272
DiabetesPedigreeFunction	-0.235890	-0.014512	-0.150449	0.088636	0.173258	0.033659	1.000000	-0.093822	0.219605
Age	0.563014	0.342557	0.194348	-0.143122	0.042256	-0.147735	-0.093822	1.000000	0.134076
Outcome	0.015717	0.288420	-0.097988	0.161583	0.169897	0.061272	0.219605	0.134076	1.000000

2. pandas

- 데이터프레임 함수 실습

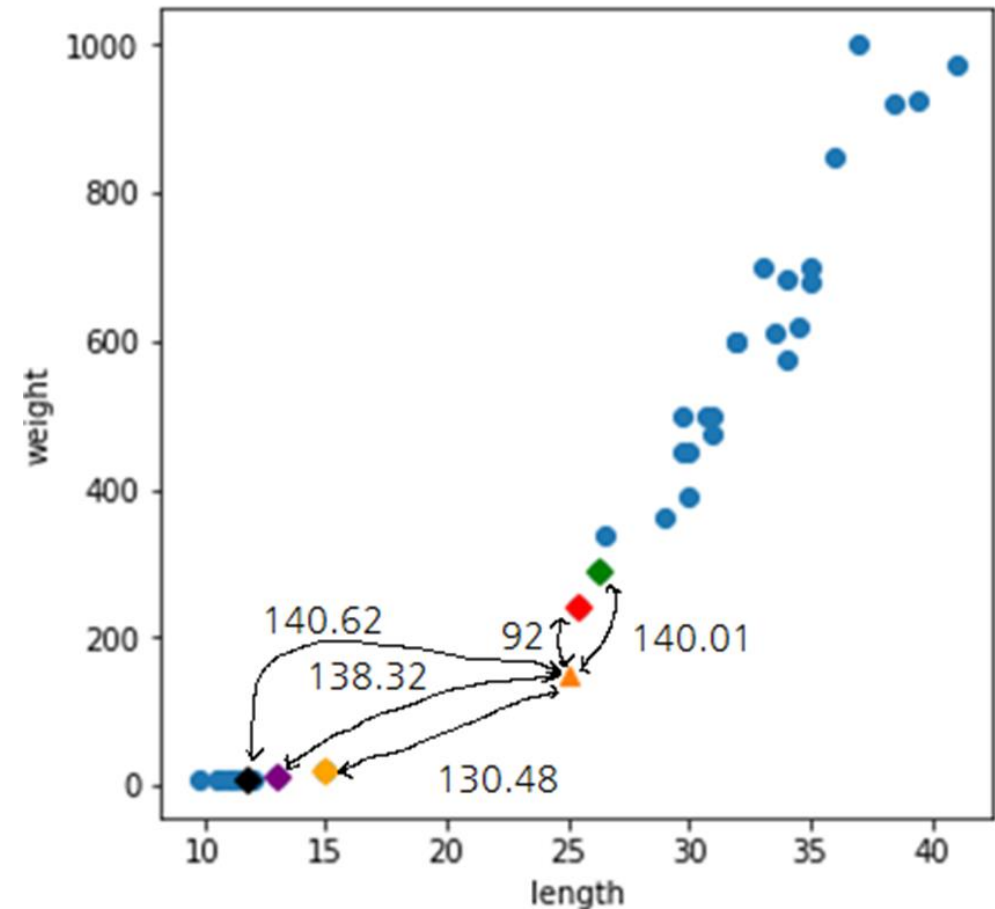
- 데이터프레임에서 칼럼의 값을 접근하여 데이터 스케일링

- MinMaxScaler

$$z_i = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$$

- StandardScaler

$$z_i = \frac{x_i - \text{mean}(x_i)}{\text{standard deviation}(x_i)}$$



2. pandas

- 데이터프레임 함수 실습
 - 데이터프레임에서 칼럼의 값을 접근하여 데이터 스케일링
 - MinMaxScaler 실습

`sklearn.preprocessing.MinMaxScaler`

```
class sklearn.preprocessing.MinMaxScaler(feature_range=(0, 1), *, copy=True, clip=False)
```

[\[source\]](#)

Transform features by scaling each feature to a given range.

This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

The transformation is given by:

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))  
X_scaled = X_std * (max - min) + min
```

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>

2. pandas

- 데이터프레임 함수 실습
 - 데이터프레임에서 칼럼의 값을 접근하여 데이터 스케일링
 - MinMaxScaler 실습

```
df_minmax = copy.deepcopy(df_copy)
for key in df_minmax.keys():
    feature_min = min(df_minmax[key])
    feature_max = max(df_minmax[key])
    df_minmax[key]=(df_minmax[key]-feature_min)/(feature_max-feature_min)
df_minmax.head()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
3	0.166667	0.033898	0.250000	0.20	0.361905	0.167939	0.129508	0.000000	0.0
19	0.166667	0.474576	0.416667	0.55	0.371429	0.664122	0.722951	0.578947	1.0
71	0.833333	0.881356	0.166667	0.80	0.580952	0.206107	0.529508	0.263158	0.0
85	0.333333	0.389831	0.583333	0.50	0.509524	0.496183	1.000000	0.315789	0.0
87	0.333333	0.220339	0.333333	0.30	0.252381	0.961832	0.386885	0.263158	0.0

2. pandas

- 데이터프레임 함수 실습
 - 데이터프레임에서 칼럼의 값을 접근하여 데이터 스케일링
 - StandardScaler 실습

`sklearn.preprocessing.StandardScaler` ¶

```
class sklearn.preprocessing.StandardScaler(*, copy=True, with_mean=True, with_std=True)
```

[\[source\]](#)

Standardize features by removing the mean and scaling to unit variance.

The standard score of a sample `x` is calculated as:

$$z = (x - u) / s$$

where `u` is the mean of the training samples or zero if `with_mean=False`, and `s` is the standard deviation of the training samples or one if `with_std=False`.

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

2. pandas

- 데이터프레임 함수 실습
 - 데이터프레임에서 칼럼의 값을 접근하여 데이터 스케일링
 - StandardScaler 실습

```
df_std = copy.deepcopy(df_copy)
for key in df_std.keys():
    df_std[key]=(df_std[key]-df_std[key].mean())/(df_std[key].std())
df_std.head()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
3	-0.606164	-1.399579	-0.574061	-0.993849	-0.415946	-1.257643	-1.317182	-1.093942	-0.484701
19	-0.606164	0.117741	0.061307	0.310578	-0.378609	0.529001	0.809979	1.169759	2.026932
71	1.553294	1.518343	-0.891745	1.242311	0.442803	-1.120209	0.116595	-0.064987	-0.484701
85	-0.066299	-0.174052	0.696676	0.124231	0.162776	-0.075709	1.803045	0.140804	-0.484701
87	-0.066299	-0.757636	-0.256377	-0.621156	-0.845321	1.600987	-0.394629	-0.064987	-0.484701

2. pandas

- 데이터프레임 함수 실습
 - values : 데이터프레임을 넘파이로 변환

```
data = df_minmax.values  
print(type(data))  
print(data)
```

[실행결과]

```
<class 'numpy.ndarray'>  
[[0.16666667 0.03389831 0.25          0.2          0.36190476 0.16793893  
 0.1295082  0.          0.          ]  
 [0.16666667 0.47457627 0.41666667 0.55          0.37142857 0.66412214  
 0.72295082 0.57894737 1.          ]  
 [0.83333333 0.88135593 0.16666667 0.8          0.58095238 0.20610687  
 0.5295082  0.26315789 0.          ]  
 [0.33333333 0.38983051 0.58333333 0.5          0.50952381 0.49618321  
 1.          0.31578947 0.          ]  
 [0.33333333 0.22033898 0.33333333 0.3          0.25238095 0.96183206  
 0.38688525 0.26315789 0.          ]  
 [1.          0.96610169 0.5          0.4          1.          0.61068702  
 0.27377049 1.          0.          ]]
```

2. pandas

- 데이터프레임 함수 실습

- 범주형 데이터 처리

- pandas.get_dummies : 숫자가 아닌 데이터들을 범주형으로 처리하도록 변경

```
raw_data = {  
'id': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],  
'blood type': ['A', 'B', 'A', 'O', 'AB', 'A', 'A', 'B', 'O', 'B']}  
df_blood_type = pd.DataFrame(raw_data, columns = ['id', 'blood type'])  
df_blood_type = pd.get_dummies(df_blood_type)  
df_blood_type.head()
```

[실행결과]

	id	blood type_A	blood type_AB	blood type_B	blood type_O
0	1	1	0	0	0
1	2	0	0	1	0
2	3	1	0	0	0
3	4	0	0	0	1
4	5	0	1	0	0

2. pandas

◦ 스스로 해보기

- Medical_Insurance_dataset.csv 데이터셋을 이용하여 이상치를 보정하고, 원 핫 인코딩을 수행하고, MinMaxScaler 혹은 StandardScaler 방식으로 정규화를 수행하여라. 필요에 따라 결측치는 보간하거나 삭제하여라.

	A	B	C	D	E	F	G
1	age	sex	bmi	smoker	region	children	charges
2	21	male	25.745	no	northeast	2	3279.869
3	36.97698	female	25.74416	yes	southeast	3	21454.49
4	18	male	30.03	no	southeast	1	1720.354
5	37	male	30.67689	no	northeast	3	6801.438

3. 시각화

- 시각화 라이브러리 설치

- Anaconda

- conda install matplotlib seaborn

- Python

- pip install matplotlib seaborn

- Jupyter Notebook에서의 설치

!conda install matplotlib seaborn

!pip install matplotlib seaborn

```
In [1]: !pip install matplotlib seaborn

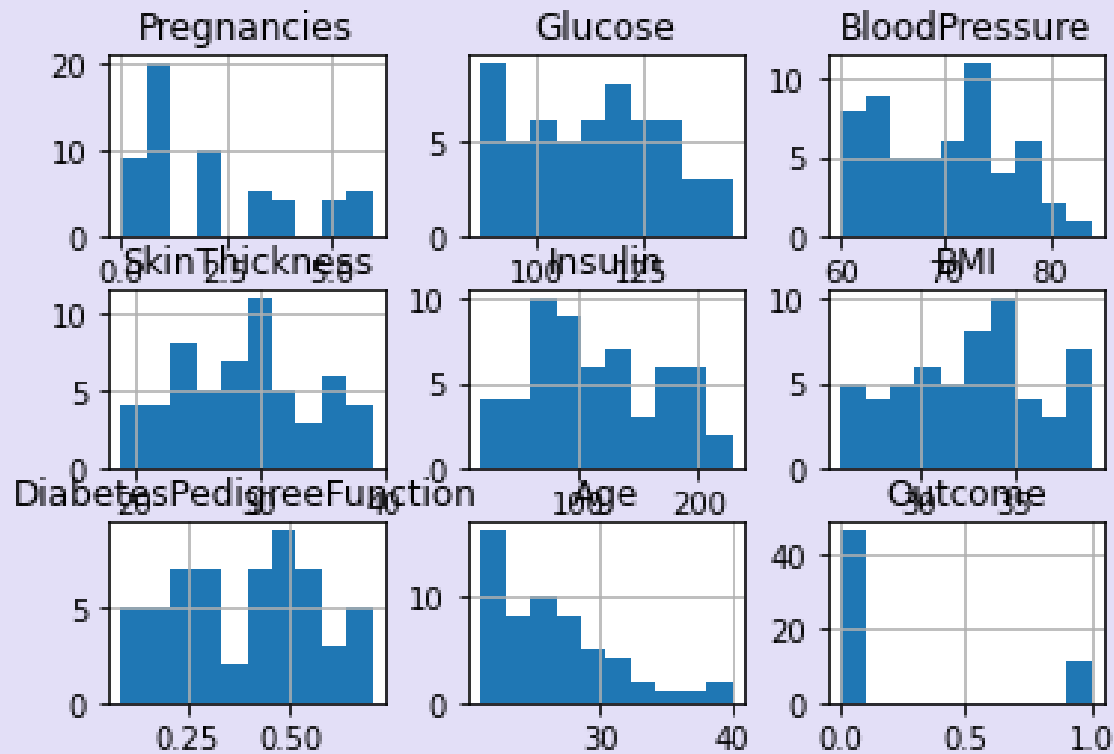
Collecting matplotlib
  Downloading matplotlib-3.6.0-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (9.4 MB)
    |#####| 9.4 MB 27.5 MB/s eta 0:00:01
Collecting seaborn
  Downloading seaborn-0.12.0-py3-none-any.whl (285 kB)
    |#####| 285 kB 18.1 MB/s eta 0:00:01
Collecting contourpy>=1.0.1
  Downloading contourpy-1.0.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (295 kB)
    |#####| 295 kB 10.6 MB/s eta 0:00:01
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.4-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.whl (1.2 MB)
    |#####| 1.2 MB 34.1 MB/s eta 0:00:01
Collecting cycler>=0.10
  Using cached cycler-0.11.0-py3-none-any.whl (6.4 kB)
Requirement already satisfied: python-dateutil>=2.7 in /home/minsuchae/.local/lib/python3.8/site-packages (from matplotlib) (2.8.2)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.37.3-py3-none-any.whl (959 kB)
    |#####| 959 kB 18.5 MB/s eta 0:00:01
Collecting numpy>=1.19
  Using cached numpy-1.23.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
Requirement already satisfied: packaging>=20.0 in /home/minsuchae/.local/lib/python3.8/site-packages (from matplotlib) (21.3)
Collecting pillow>=6.2.0
  Downloading Pillow-9.2.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.1 MB)
    |#####| 3.1 MB 32.5 MB/s eta 0:00:01
Requirement already satisfied: pyparsing>=2.2.1 in /home/minsuchae/.local/lib/python3.8/site-packages (from matplotlib) (3.0.7)
Collecting pandas>=0.25
  Using cached pandas-1.5.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.2 MB)
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from python-dateutil>=2.7->matplotlib) (1.14.0)
Collecting pytz>=2020.1
  Using cached pytz-2022.2.1-py2.py3-none-any.whl (500 kB)
Installing collected packages: numpy, contourpy, kiwisolver, cycler, fonttools, pillow, matplotlib, pytz, pandas, seaborn
WARNING: The scripts f2py, f2py3 and f2py3.8 are installed in '/home/minsuchae/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The scripts fonttools, pyftmerge, pyftsubset and ttx are installed in '/home/minsuchae/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed contourpy-1.0.5 cycler-0.11.0 fonttools-4.37.3 kiwisolver-1.4.4 matplotlib-3.6.0 numpy-1.23.3 pandas-1.5.0 pillow-9.2.0 pytz-2022.2.1 seaborn-0.12.0
```

3. 시각화

- 히스토그램 출력
 - dataframe.hist()

```
df_copy.hist()
```

[실행결과]



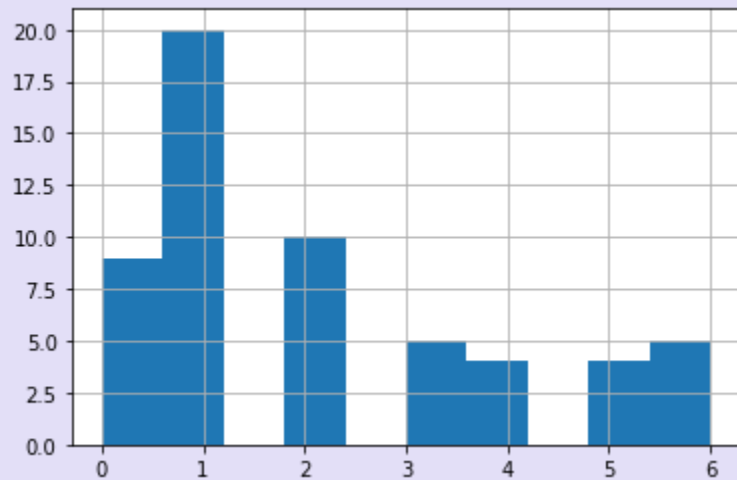
3. 시각화

- 히스토그램 출력

- dataframe.hist()

```
df_copy['Pregnancies'].hist()
```

[실행결과]



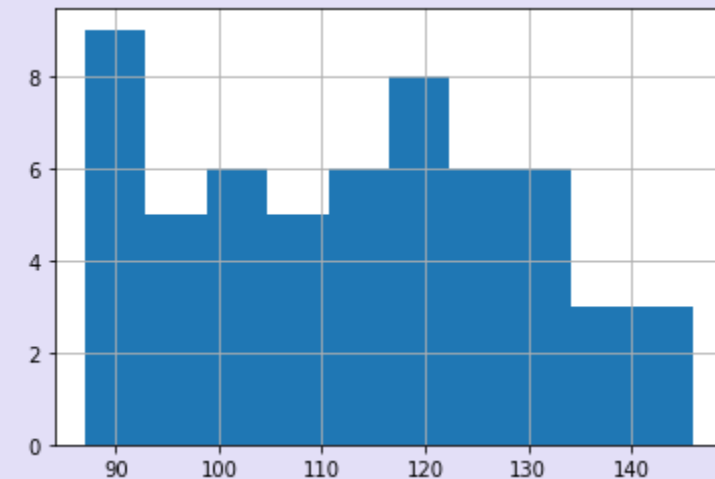
3. 시각화

- 히스토그램 출력

- dataframe.hist()

```
df_copy['Glucose'].hist()
```

[실행결과]



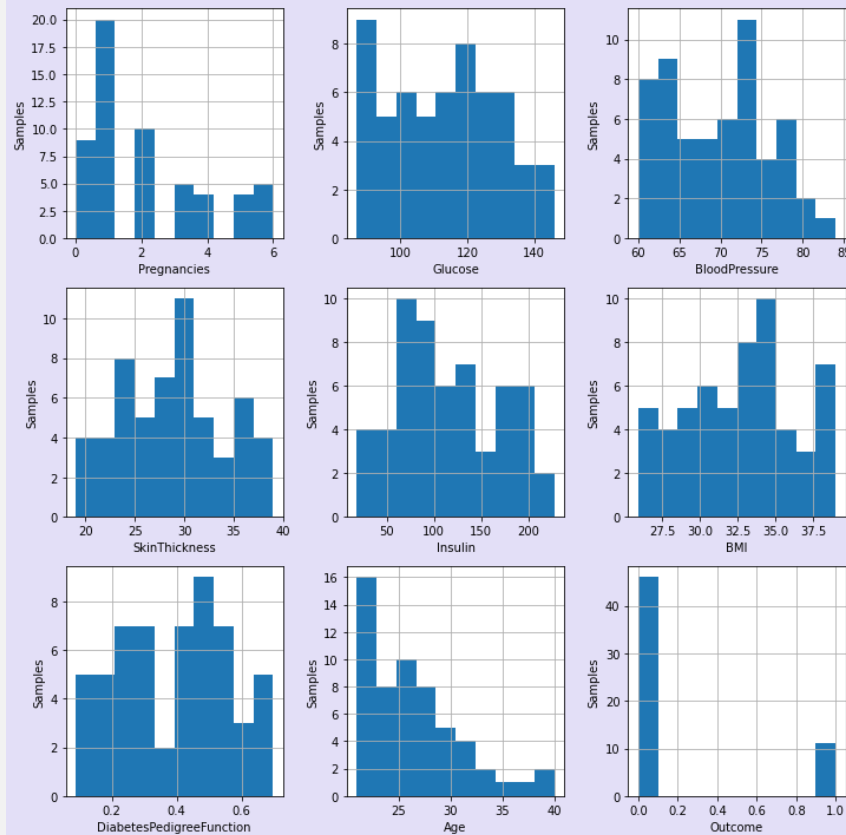
3. 시각화

- 히스토그램 출력 개선

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.figure(figsize=(10,10))
for i, key in enumerate(df_copy.keys()):
    plt.subplot(3, 3, i+1)
    plt.ylabel('Samples')
    plt.xlabel(key)
    df_copy[key].hist()
plt.tight_layout()
plt.show()
```

[실행결과]



3. 시각화

- 히스토그램 출력 개선 설명

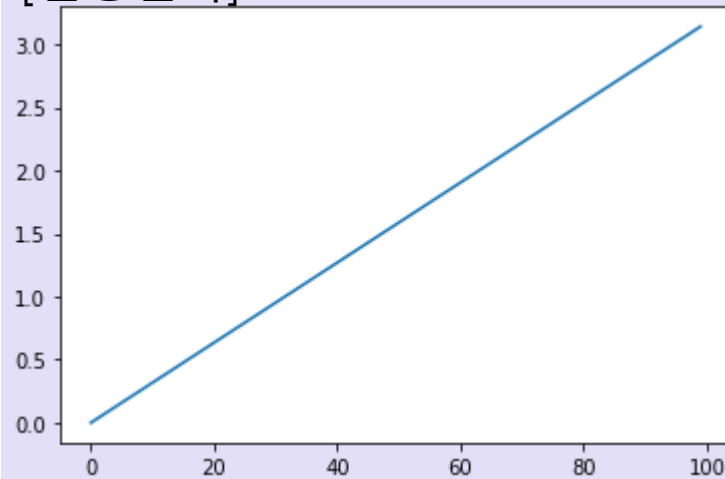
- `%matplotlib inline` : jupyter notebook에 이미지 출력하도록 설정
- `matplotlib.pyplot.figure` : 출력할 이미지 크기
- `matplotlib.pyplot.subplot` : 여러 이미지들을 출력하도록 함(행, 열, 인덱스)
- `matplotlib.pyplot.xlabel` : 그래프 출력 시 x축 레이블
- `matplotlib.pyplot.ylabel` : 그래프 출력 시 y축 레이블

3. 시각화

- 다양한 선 그리기

```
x = np.linspace(0,np.pi,100)  
plt.plot(x)  
plt.show()
```

[실행결과]

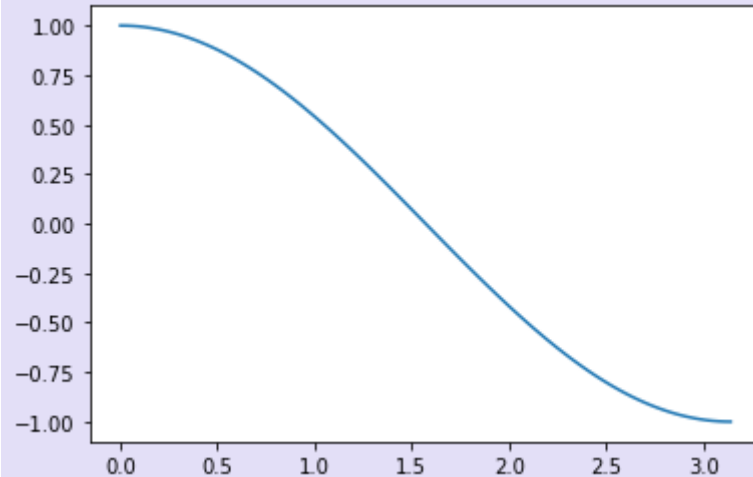


3. 시각화

- 다양한 선 그리기

```
sin_y = np.sin(x)  
plt.plot(x,sin_y)  
plt.show()
```

[실행결과]

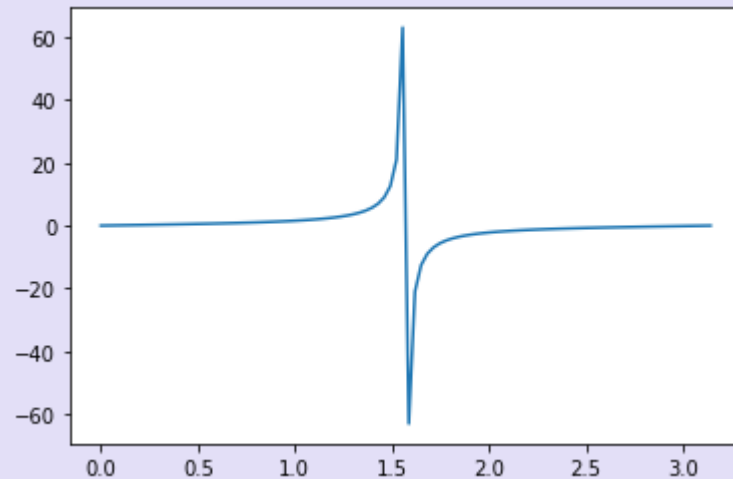


3. 시각화

- 다양한 선 그리기

```
tan_y = np.tan(x)  
plt.plot(x, tan_y)  
plt.show()
```

[실행결과]

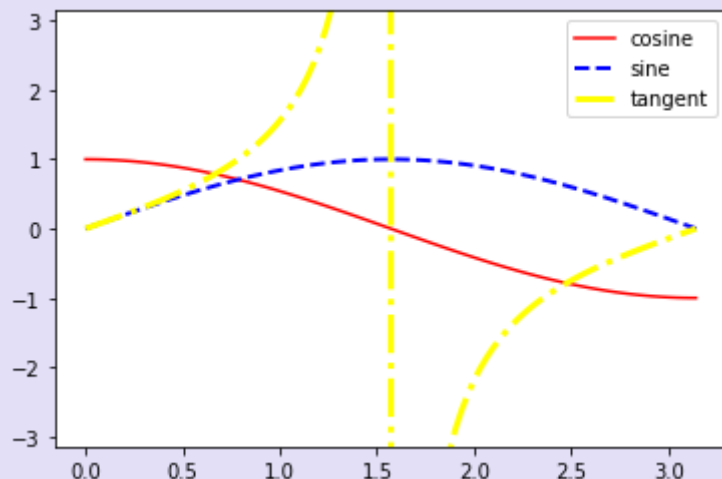


3. 시각화

- 다양한 선 그리기

```
plt.plot(x, cos_y, linestyle='-', color='red', label='cosine')  
plt.plot(x, sin_y, linestyle='--', color='blue', linewidth=2, label='sine')  
plt.plot(x, tan_y, linestyle='-.', color='yellow', linewidth=3, label='tangent')  
plt.ylim(-np.pi, np.pi)  
plt.legend()  
plt.show()
```

[실행결과]



3. 시각화

- 다양한 선 그리기 설명

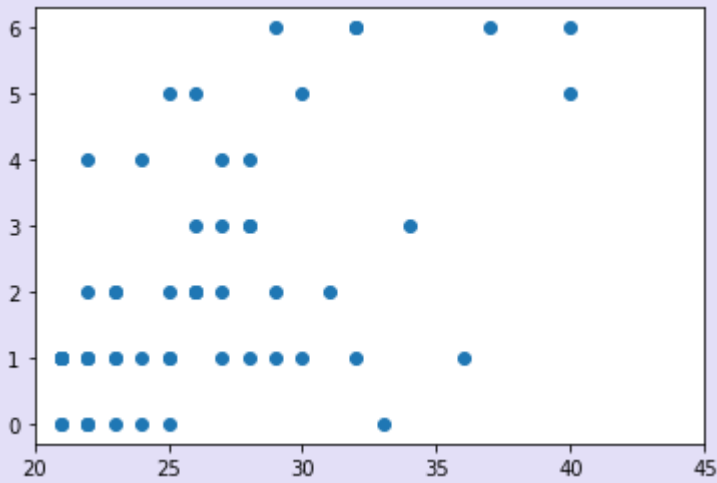
- matplotlib.pyplot.plot : x좌표와 y좌표를 바탕으로 선을 그림
 - 좌표를 하나만 입력하면 x축은 데이터순번
 - linestyle : 선 스타일
 - color : 선 색상
 - linewidth : 선의 굵기
 - label : 선의 구분자
 - marker : 마커
 - markersize : 마커 크기
- matplotlib.pyplot.ylim : 출력할 그래프의 y축 값의 상한과 하한을 지정. 생략 시 데이터의 값 범위를 바탕으로 추론
- matplotlib.pyplot.legend : 범례 표시. loc 인자를 통해 범례를 출력할 좌표 혹은 지점 설정. 생략 시 좌상단, 좌하단, 우상단, 우하단 중 최적으로 선택.

3. 시각화

◦ 산점도 그리기

```
age = df_copy['Age'].values  
pregnancies = df_copy['Pregnancies'].values  
plt.xlim(20,45)  
plt.scatter(age,pregnancies)  
plt.show()
```

[실행결과]

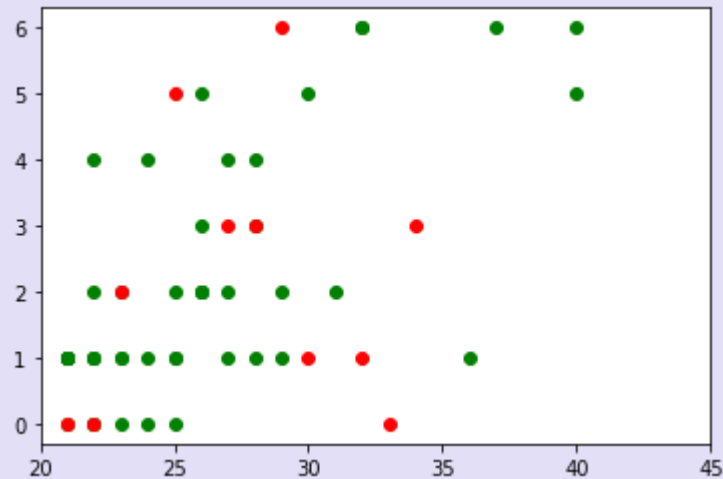


3. 시각화

- 산점도 그리기 - 당뇨병 여부에 따른 분류

```
outcome = df_copy['Outcome'].values  
plt.xlim(20,45)  
plt.scatter(age[outcome==0],pregnancies[outcome==0],c='green')  
plt.scatter(age[outcome==1],pregnancies[outcome==1],c='red')  
plt.show()
```

[실행결과]



3. 시각화

- 산점도 그리기 - 당뇨병 여부에 따른 분류

```
count_patients = {}
for i in range(len(age)):
    if (age[i],pregnancies[i],outcome[i]) not in count_patients:
        count_patients[age[i],pregnancies[i],outcome[i]] = 1
    else:
        count_patients[age[i],pregnancies[i],outcome[i]] += 1
count_patients = dict(sorted(count_patients.items(), key=lambda item: (item[0][0],item[0][1],item[0][2])))
print(count_patients)
```

[실행결과]

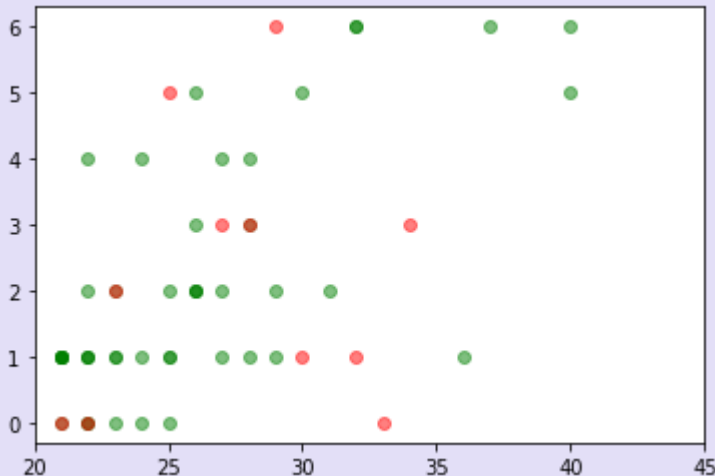
```
{(21.0, 0.0, 0): 1, (21.0, 0.0, 1): 1, (21.0, 1.0, 0): 6, (22.0, 0.0, 0): 2, (22.0, 0.0, 1): 1, (22.0, 1.0, 0): 3, (22.0, 2.0, 0): 1, (22.0, 4.0, 0): 1, (23.0, 0.0, 0): 1, (23.0, 1.0, 0): 2, (23.0, 2.0, 0): 1, (23.0, 2.0, 1): 1, (24.0, 0.0, 0): 1, (24.0, 1.0, 0): 1, (24.0, 4.0, 0): 1, (25.0, 0.0, 0): 1, (25.0, 1.0, 0): 2, (25.0, 2.0, 0): 1, (25.0, 5.0, 1): 1, (26.0, 2.0, 0): 3, (26.0, 3.0, 0): 1, (26.0, 5.0, 0): 1, (27.0, 1.0, 0): 1, (27.0, 2.0, 0): 1, (27.0, 3.0, 1): 1, (27.0, 4.0, 0): 1, (28.0, 1.0, 0): 1, (28.0, 3.0, 0): 1, (28.0, 3.0, 1): 1, (28.0, 4.0, 0): 1, (29.0, 1.0, 0): 1, (29.0, 2.0, 0): 1, (29.0, 6.0, 1): 1, (30.0, 1.0, 1): 1, (30.0, 5.0, 0): 1, (31.0, 2.0, 0): 1, (32.0, 1.0, 1): 1, (32.0, 6.0, 0): 2, (33.0, 0.0, 1): 1, (34.0, 3.0, 1): 1, (36.0, 1.0, 0): 1, (37.0, 6.0, 0): 1, (40.0, 5.0, 0): 1, (40.0, 6.0, 0): 1}
```

3. 시각화

- 산점도 그리기 - 당뇨병 여부에 따른 분류

```
plt.xlim(20,45)  
plt.scatter(age[outcome==0],pregnancies[outcome==0],c='green',alpha=0.5)  
plt.scatter(age[outcome==1],pregnancies[outcome==1],c='red',alpha=0.5)  
plt.show()
```

[실행결과]



3. 시각화

- 산점도 그리기 설명

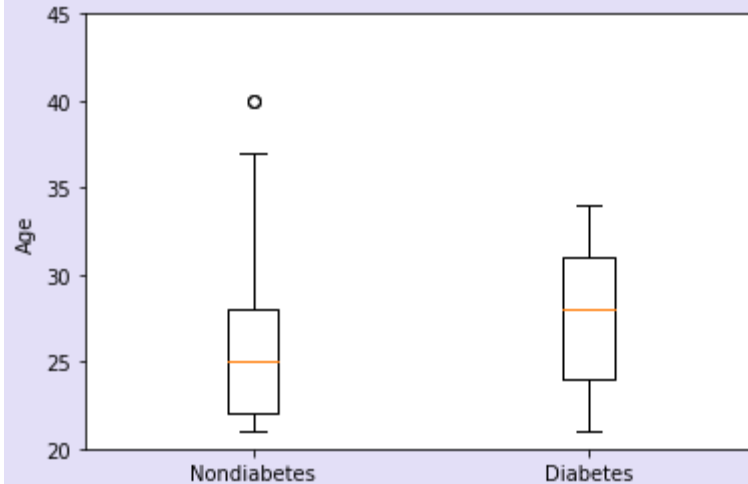
- matplotlib.pyplot.scatter : 해당하는 좌표에 찍음
 - c : 색상
 - alpha : 투명도
- matplotlib.pyplot.xlim : 출력할 그래프의 x축 값의 상한과 하한을 지정. 생략 시 데이터의 값의 범위를 바탕으로 추론

3. 시각화

- 수치 데이터 표현

```
plt.boxplot([age[outcome==0], age[outcome==1]],labels=['Nondiabetes','Diabetes'])  
plt.ylim(20,45)  
plt.ylabel('Age')  
plt.show()
```

[실행결과]

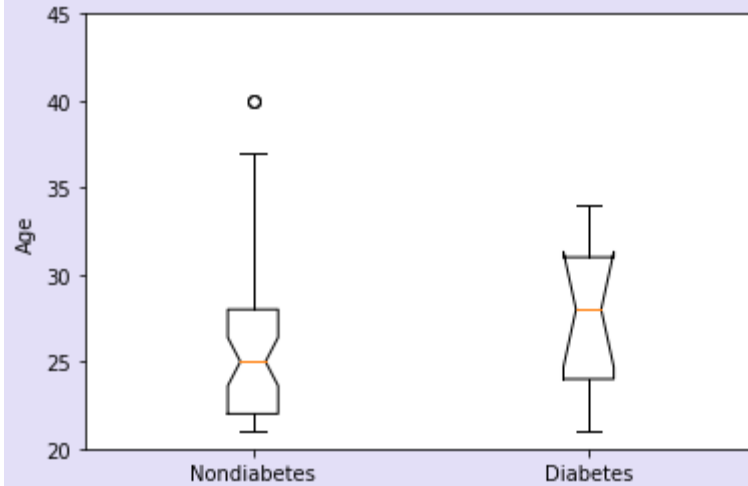


3. 시각화

- 수치 데이터 표현

```
plt.boxplot([age[outcome==0], age[outcome==1]], labels=['Nondiabetes', 'Diabetes'], notch=True)  
plt.ylim(20,45)  
plt.ylabel('Age')  
plt.show()
```

[실행결과]



3. 시각화

- 수치 데이터 표현 설명

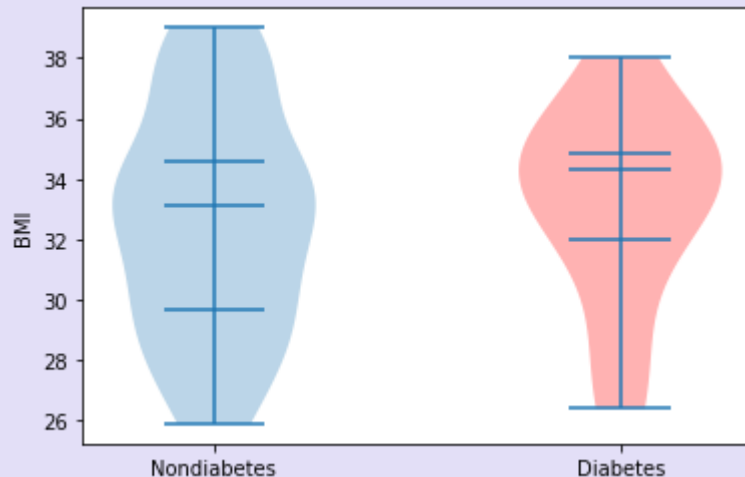
- matplotlib.pyplot.boxplot : 박스플롯을 그림으로써 데이터 범주 확인
 - notch : 중앙값의 신뢰 구간 95%값에 노치를 그림
 - labels : x축 항목별 레이블

3. 시각화

- 수치 데이터 표현

```
bmi = df_copy['BMI'].values  
violin=plt.violinplot([bmi[outcome==0],bmi[outcome==1]],quantiles=[[0.25,0.5,0.75],[0.25,0.5,0.75]])  
violin['bodies'][1].set_facecolor('red')  
plt.xticks(ticks=[1,2], labels=['Nondiabetes','Diabetes'])  
plt.ylabel('BMI')  
plt.show()
```

[실행결과]



3. 시각화

- 수치 데이터 표현 설명

- matplotlib.pyplot.violinplot : 바이올린 플롯을 그림으로써 데이터 범주 확인
 - quantiles : 데이터셋 범주 확인할 구분선 설정(0 ~ 1)
- matplotlib.pyplot.violinplot[name] : 바이올린 플롯 영역의 키에 따른 값 접근
 - bodies : 채워지는 영역
 - ✓ set_facecolor를 통해 색상 지정
 - cmeans : 평균값
 - cmins : 최소값
 - mmaxes : 최대값
 - cbars : 중심
 - cmedians : 중간값
 - cquantiles : 분위값 확인
 - ✓ set_edgecolor를 통해 직선 색상 지정

3. 시각화

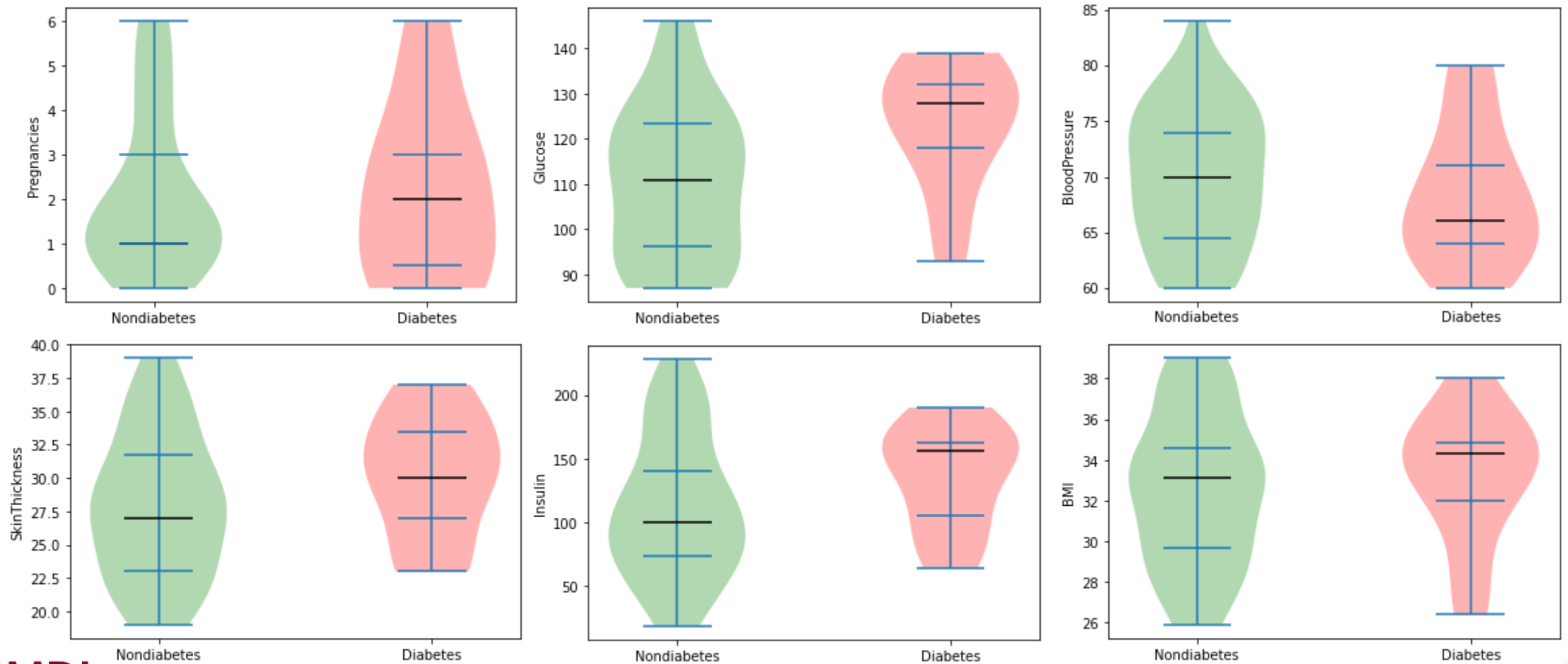
- 모든 특징 수치 데이터 표현

```
keys = df_copy.keys()
keys = keys.drop('Outcome')

outcome = df_copy['Outcome'].values
for key in keys :
    values = df_copy[key].values
    violin=plt.violinplot([values[outcome==0],values[outcome==1]],quantiles=[[0.25,0.75],[0.25,0.75]], showmedians=True)
    violin['bodies'][0].set_facecolor('green')
    violin['bodies'][1].set_facecolor('red')
    violin['cmedians'].set_edgecolor('black')
    plt.xticks(ticks=[1,2], labels=['Nondiabetes','Diabetes'])
    plt.ylabel(key)
    plt.show()
```

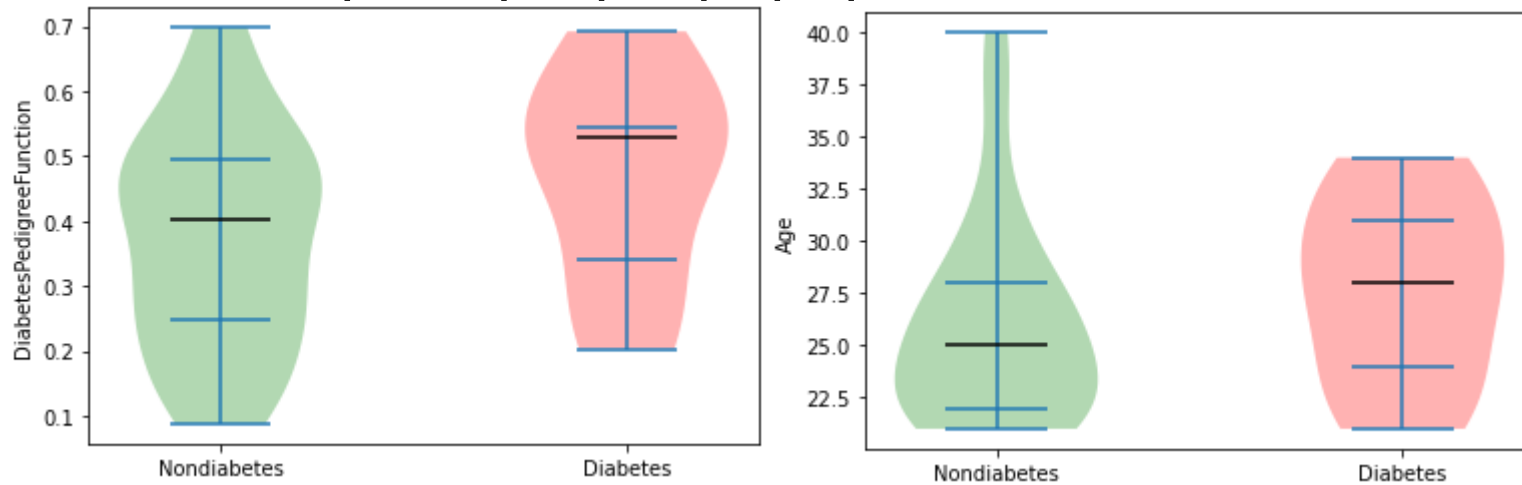
3. 시각화

◦ 모든 특징 수치 데이터 표현



3. 시각화

◦ 모든 특징 수치 데이터 표현



3. 시각화

- 상관관계 분석 시각화

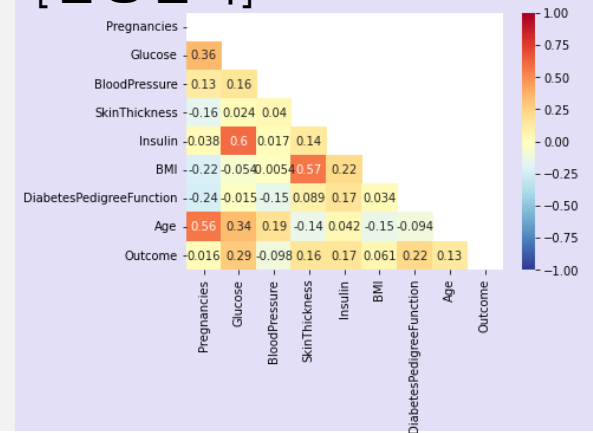
- seaborn.heatmap : 값의 범위에 따라 시각화

```
import seaborn as sns
```

```
corr = df_copy.corr()  
mask = np.zeros_like(corr, dtype=bool)  
mask[np.triu_indices_from(mask)] = True
```

```
sns.heatmap(corr, cmap='RdYlBu_r', annot = True, mask=mask, vmin=-1,vmax=1)  
plt.show()
```

[실행결과]



3. 시각화

- 상관관계 분석 시각화

- 부가 설명

- <https://wikidocs.net/book/5011>
 - https://matplotlib.org/stable/api/pyplot_summary.html
 - <https://seaborn.pydata.org/generated/seaborn.boxplot.html>
 - <https://seaborn.pydata.org/generated/seaborn.boxenplot.html>
 - <https://seaborn.pydata.org/generated/seaborn.violinplot.html>
 - <https://seaborn.pydata.org/generated/seaborn.boxenplot.html>
 - <https://seaborn.pydata.org/generated/seaborn.pointplot.html>
 - <https://seaborn.pydata.org/generated/seaborn.barplot.html>
 - <https://seaborn.pydata.org/generated/seaborn.heatmap.html>

3. 시각화

- 스스로 해보기

- Medical_Insurance_dataset.csv 데이터셋을 이용하여 바이올린 플롯을 그려라. 상관관계 분석에 대한 시각화를 수행하여라

	A	B	C	D	E	F	G
1	age	sex	bmi	smoker	region	children	charges
2	21	male	25.745	no	northeast	2	3279.869
3	36.97698	female	25.74416	yes	southeast	3	21454.49
4	18	male	30.03	no	southeast	1	1720.354
5	37	male	30.67689	no	northeast	3	6801.438