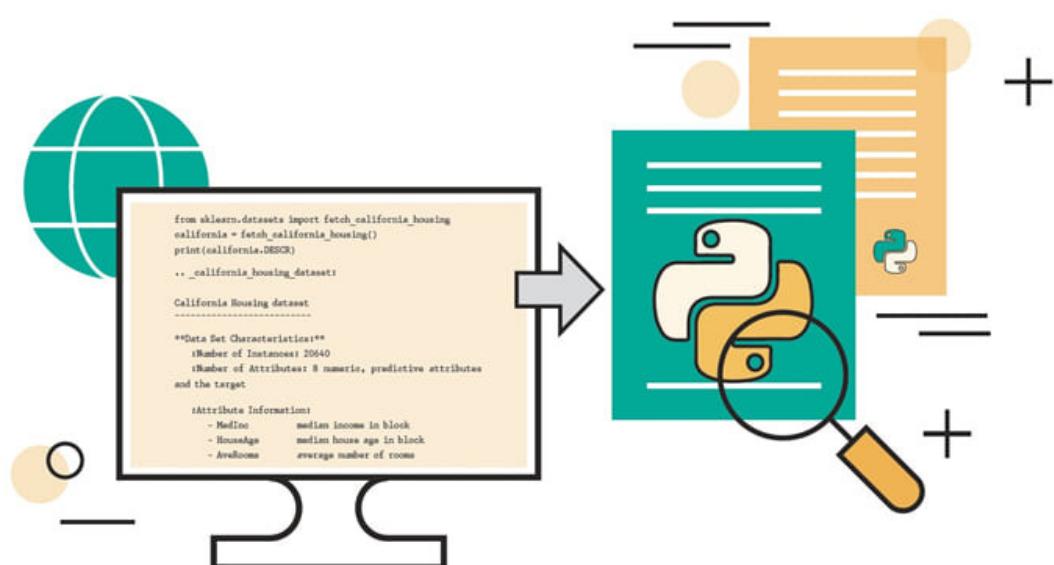


# Python으로 쉽게 배우는 통계 프로그래밍

김재희 지음



자유아카데미

# Python으로 쉽게 배우는 통계 프로그램 (김재희 지음) 으로 통계분석을 학습하고자 합니다.

## ▼ 1 프로그래밍 언어 Python에 대하여

### 1.1 프로그래밍 언어 Python 소개

- Python은 1991년, 프로그래머 귀도 반 로섬이 발표한 고급 프로그래밍 언어로, 플랫폼에 독립적인 인터프리터식, 객체지향적, 동적 타이핑 대화형 언어이다.
- Python은 비영리의 Python 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발 모델을 가지고 있으며, 무료로 사용할 수 있는 오픈소스다.
- Python은 동적 타이핑 범용 프로그래밍 언어로 다양한 플랫폼에서 사용할 수 있으며 라이브러리가 풍부하다.
- Python은 통계분석이나 머신러닝 등의 프로그램을 매우 짧은 코드로 작성할 수 있는데, 문법 자체가 통계에 특화된 프로그래밍 언어 R에 비해 범용적인 언어이며 라이브러리가 풍부하기에 그 활용성이 증가하고 있다.

## ▼ 1.5 Python 코딩의 기본 문법

- 정적 타입 언어 : 자료형을 컴파일 타임에 결정하는 언어, 컴파일 시 자료형에 맞지 않은 값이 들어있으면 컴파일 에러가 발생한다. (C언어)
- 동적 타입 언어 : 자료형을 실행 시점에 결정하는 언어 (JavaScript, Ruby, Python)
- 약 타입 언어 : 자료형이 일치하지 않을 경우 암묵적으로 타입을 변환하는 언어
- 강 타입 언어 : 자료형이 일치하지 않을 경우 에러가 발생하며 암묵적 변환을 지원하지 않음

👉 Python 은 동적 타입이면서, 강 타입 프로그래밍 언어이다.

## ▼ 1.5.1 들여쓰기

가독성을 위해 들여쓰기를 하고 같은 블록 내에서는 들여쓰기 칸의 수가 같아야 하고 공백과 탭을 혼용하여 사용하면 안된다.

```
a = 10

if a == 10:
    print('a is 10')

a is 10
```

## ▼ 1.5.2 주석

'#' 기호로 시작하는 구문은 주석으로서 프로그램 수행에 영향을 주기 않으며, 사람만이 알아볼 수 있도록 코드의 여러 정보를 기록하는 주석을 이용하면 가독성을 높일 수 있다.

```
# 이것이 바로 주석이다. 이말이다!
```

## ▼ 1.5.4 문장 구분 세미콜론 활용

한 줄에 한 개의 코딩문장을 쓰는 것이 기본이나 한 줄에 여러 개의 코딩문장을 쓰고자 한다면 세미콜론 ';'을 사용 한다.

```
print('Hello');print('world!')
```

```
Hello  
world!
```

```
print('Hello')  
print('World!')
```

```
Hello  
World!
```

## ▼ 1.5.5 기본 라이브러리

😊 Python 라이브러리란, 전 세계의 Python 사용자들이 만든 유용한 프로그램을 모아 놓은 것이며, Python을 설치할 때 자동으로 컴퓨터에 설치된다.

⚡ '라이브러리(library)'란, '도서관'이라는 뜻 그대로 원하는 정보를 찾아보는 곳이다.

→ 자주 사용하는 필요한 라이브러리를 알아 두고 임포트하여 활용하면 된다.

👉 Python에 내장된 함수만으로는 작업의 한계가 또렷하다. 따라서 좀 더 복잡한 프로그램을 만들기 위해서는 Python의 모듈과 패키지를 이용해야한다.

- 모듈이란, 각종 변수, 함수, 클래스를 담고 있는 파일이다.
- 패키지란, 여러 모듈을 묶은 것이다.

```
import 모듈 as 이름
```

```
from 모듈 import 변수  
from 모듈 import 함수  
from 모듈 import 클래스
```

```
from 모듈 import *
```

```
from 모듈 import 변수 as 이름
```

```
from 모듈 import 함수 as 이름
from 모듈 import 클래스 as 이름

from 모듈 import 변수 as 이름1, 함수 as 이름2, 클래스 as 이름3
```

**import**로 가져온 모듈(변수, 함수, 클래스) 삭제 가능

```
del 모듈
```

모듈에 포함된 함수 목록 확인 가능

```
dir(모듈)
```

math : 수학 연산 기본 라이브러리

math 패키지를 임포트한 후 math.sqrt()와 같이 사용하면 된다.

```
import math
math.sqrt(4)
```

2.0

```
import math as m
m.sqrt(4)
```

2.0

```
from math import sqrt
sqrt(4.0)
```

2.0

```
from math import sqrt, pi
print(pi)
sqrt(4.0)
```

3.141592653589793  
2.0

```
from math import *
print(pi)
sqrt(4.0)
```

3.141592653589793  
2.0

```
from math import sqrt as s  
s(4.0)
```

2.0

```
from math import pi as p, sqrt as s  
print(p)  
s(4.0)
```

3.141592653589793  
2.0

```
import math  
del math
```

```
from math import factorial  
factorial(4)
```

24

```
from math import factorial as fac  
fac(4)
```

24

```
import math  
dir(math)
```

```
[ '__doc__',  
  '__loader__',  
  '__name__',  
  '__package__',  
  '__spec__',  
  'acos',  
  'acosh',  
  'asin',  
  'asinh',  
  'atan',  
  'atan2',  
  'atanh',  
  'ceil',  
  'comb',  
  'copysign',  
  'cos',  
  'cosh',  
  'degrees',  
  'dist',  
  'e',  
  'erf',  
  'erfc',  
  'exp',  
  'expm1',  
  'fabs',  
  'gcd',  
  'inf',  
  'isfinite',  
  'isinf',  
  'isnan',  
  'pi',  
  'prod',  
  'tau',  
  'trunc']
```

```
'factorial',
'floor',
'fmod',
'frexp',
'fsum',
'gamma',
'gcd',
'hypot',
'inf',
'isclose',
'isfinite',
'isinf',
'isnan',
'isqrt',
'ldexp',
'lgamma',
'log',
'log10',
'log1p',
'log2',
'modf',
'nan',
'perm',
'pi',
'pow',
'prod',
'radians',
'remainder',
'sin',
'sinh',
'sqrt',
'tan',
'tanh'
```

#### ▼ 1.5.6 통계 분석에 유용한 라이브러리

😊 Python의 scikit-learn은 딥러닝을 제외한 머신러닝 라이브러리이며 머신러닝에서 가장 많이 활용되는 분류, 회귀, 랭킹, 예측 등의 다양한 알고리즘을 내장하고 있다.

참고 : Python 머신러닝 애플리케이션에서 수행하는 실제 연산작업 대부분은 일반적으로 C, C+, 자바 등으로 작성한다.

😊 Python은 라이브러리를 통해 이를 정리하고 상호작용하는 것이다.

표 1.3 주요 라이브러리

주요 라이브러리	주요 기능	함수 예
pandas	다양한 형식의 데이터 로드	pd.read_csv() pd.read_excel()
	다양한 형식의 데이터 생성	pd.DataFrame() pd.Series()
	데이터 병합, 결합, 연결	pd.concat(), pd.merge()
	데이터 테이블 모양 변경 및 피벗	dataname.melt() dataname.pivot()
	누락된 데이터 작업	pd.isna() dataname.fillna()
	통계적 함수: 공분산, 상관계수, 순위, 사분위수 등	dataname.cov() dataname.corr() dataname.rank() dataname.quantile()
	데이터를 그룹으로 분할하여 집계, 변환, 여과	dataname.groupby.sum() dataname.groupby.last() dataname.groupby.first()
	간단한 시계열 분석 금융데이터 처리	pd.date_range() pd.Series().resample().mean()
	행렬, 배열(array) 생성	np.arange() np.zeros()
	기본 연산	A+B, A-B, A@B
numpy	인덱싱, 슬라이싱 및 반복	A[], A[:,], A[:, :] np.fromfunction()
	배열의 구조 변경	arrayname.resize() arrayname.reshape() arrayname.ravel()
	그래프 구현 및 편집	plt.plot() plt.title() plt.xlabel()
matplotlib	matplotlib 기반으로 복잡한 그래프 시각화	sns.replot() sns.catplot()
seaborn		

표 1.4 통계 관련 라이브러리

주요 라이브러리	기능	함수 예
scipy	numpy와 상호작용 - 인덱스 트릭 - 다항식 - 벡터화 함수 - 유형 처리	np.some_function() np.vectorize()
	특수기능(scipy.special) - 물리수학의 함수 정의	# 실수차수의 bessel 함수 special.jn_zeros()
	통계분석(scipy.stats): 확률분포의 확률밀도함수, 누적분포함수, 신뢰구간 등	stat.norm.pdf() stat.norm.cdf() stat.expon.rvs() stat.uniform.ppf()
statsmodels	회귀 및 선형모형 - 선형회귀분석 - 일반화 선형모형 - 일반화 추정방정식 - 이산 종속 변수를 사용한 회귀 - 분산분석	sm.ols() sm.GLM() smf.gee() sm.RLM() sm.stats.anova_lm() sm.Logit()
	시계열 분석 - 시계열 분석 tsa - 상태 공간 방법에 의한 시계열 분석 statespace - 벡터 자동회귀 tsa.vector_ar	sm.tsa.UnobservedComponents() sm.tsa.VARMAX() sm.tsa.base.datetools()
	- 생존분석 방법 - 비모수 방법	sm.SurvfuncRight() sm.nonparametric.kernel_regression.KernelReg()
scikit-learn (sklearn)	머신러닝 라이브러리	

```

import numpy as np      # 기초 수학 연산 및 행렬 계산
import pandas as pd     # 데이터프레임 사용
import scipy as sp       # numpy와 상호작용, 통계분석
from scipy import stats
from sklearn import datasets    # iris와 같은 내장 데이터 사용
from sklearn.model_selection import train_test_split    # train, test 데이터 분할
from sklearn.linear_model import LinearRegression      # 선형회귀분석

```

```

from sklearn.linear_model import LogisticRegression      # 로지스틱 회귀분석
from sklearn.naive_bayes import GaussianNB            # 나이브 베이즈
from sklearn import svm                                # 서포트 벡터 머신
from sklearn import tree                             # 의사결정나무
from sklearn.ensemble import RandomForestClassifier   # 랜덤포레스트
import matplotlib.pyplot as plt                      # plot 그릴 때 사용
import seaborn as sns

import statsmodels.formula.api as smf    # 통계모형
import statsmodels.api as sm
sns.set()
%precision 3    # 소수점 3자리까지 표현

' %r '

```

## 2 Python 사용 기초

### 2.1 데이터란 무엇인가?

데이터는 지식을 형성하는 데 매우 중요하다.

데이터를 가공하고 분석하여 정보를 만들어내고, 정보에서 지식을 창출하기 때문이다.

빅데이터란, 데이터의 생성 양, 주기, 형식 등이 기존 데이터보다 너무 크기 때문에 기존의 방법으로는 수집, 저장, 검색, 분석이 어려운 방대한 데이터를 말한다.

## ▼ 2.2 Python 프로그래밍 기본

### ▼ 2.2.1 사칙연산

Python에서 연산연산자(+,-,/,\* )을 이용하여 간단한 수한 계산 기능을 이용할 수 있다.

```

print(2+3)
print(3-1)
print(3*3)
print(4/2)
print(4**2)

```

```

5
2
9
2.0
16

```

### 비교연산자

```
<      lesser than
>      greater than
<=     lesser than or equal to
>=     greater than or equal to
==     equal
!=     different (not equal)
```

```
print(1>0.8)
print(2<3)
print(2==2)
print(2!=2)
print(3<2)
```

```
True
True
True
False
False
```

## 논리연산자

```
!x      logical Not
x&y    logical And
x|y    logical Or
xor(x,y)  exclusive Or
```

표 2.8 Python에서의 연산자

Operator	의미	예
=	equal	
+	add : 더하기	
-	subtract: 빼기	
*	multiply: 곱하기	
/	divide: 나누기	
%	moduls: 나머지	$9/2 = 1$
//	floor: 내림	$9//2 = 4 \quad -9//2 = -5$
**	exponent	
&	and	
	or	
^	exponent	$2^3 = 2*2*2$
<< binary	binary left shift	$a=0011\ 1100 \quad a<<2 :1111\ 0000$
>> binary	binary right shift	$a=0011\ 1100 \quad a>>2 :0000\ 1111$

표 2.9 Python에서 연산자를 이용한 할당

Operator	Example	Equivalent to
=	$x = 5$	$x = 5$
+=	$x += 5$	$x = x + 5$
-=	$x -= 5$	$x = x - 5$
*=	$x *= 5$	$x = x * 5$
/=	$x /= 5$	$x = x / 5$
%=	$x %= 5$	$x = x \% 5$
//=	$x //= 5$	$x = x // 5$
**=	$x **= 5$	$x = x ** 5$
&=	$x &= 5$	$x = x \& 5$
=	$x  = 5$	$x = x   5$
^=	$x ^= 5$	$x = x ^ 5$
>>=	$x >>= 5$	$x = x >> 5$
<<=	$x <<= 5$	$x = x << 5$

### ▼ 2.2.3 데이터 유형

## 문자형, 숫자형, 부울형

숫자형은 정수형과 실수형이 있다.

```
type() # 데이터 유형 확인 가능
```

다른 유형의 데이터끼리 연산하면 에러가 발생한다.

```
print("A")
print(type("A"))
print(type(1))
print(type("1"))
print(type(2.5))
print(type(True))
```

```
A
<class 'str'>
<class 'int'>
<class 'str'>
<class 'float'>
<class 'bool'>
```

```
print("A"+2)
```

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-26-e0a5e54e7920> in <module>
----> 1 print("A"+2)

TypeError: can only concatenate str (not "int") to str
```

SEARCH STACK OVERFLOW

## ▼ 2.2.4 변수

데이터를 할당하는 객체로 변수를 이용한다.

```
x = 2
x+1
```

3

## ▼ 2.2.5 함수를 이용한 계산

함수란 계산 로직을 저장해놓은 것이다.

계산 로직을 '처리', 또는 '메서드'로도 부른다.

같은 계산을 여러 번 해야 할 경우 처리를 함수로 저장해두는 것이 편리하다.

```

def 함수명(인수):
    처리

x = 3
(x+2)*4

20

def ex_function(data):
    return ((data+2)*4)

print(ex_function(3))

20

```

## ▼ 2.2.6 클래스와 인스턴스

클래스와 인스턴스란,

데이터 구조와 계산 로직을 한 군데서 사용할 수 있도록 함께 묶어준 목록이라고 이해할 수 있다.

- 클래스 : 데이터와 기능을 함께 묶는 방법 제공
- 인스턴스 : 클래스에 의해 만들어진 객체이며 각자의 값을 가짐
- 메서드 : 함수와 비슷한 개념으로 클래스에 묶여서 클래스 안에 있는 인스턴스와 관계되는 함수

 Python 클래스는 객체지향형 프로그래밍의 모든 표준 기능들을 제공한다.

즉 클래스 상속 메커니즘은 다중 베이스 클래스를 허락하고, 자식 클래스는 베이스 클래스나 클래스들의 어떤 메서드도 재정의할 수 있으며, 메서드는 같은 이름의 베이스 클래스의 메서드를 호출할 수 있다.

객체들은 임의의 종류의 데이터를 양적 제한 없이 가질 수 있다. 모듈과 마찬가지로, 클래스는 Python의 동적인 본성을 함께 나누며 실행 시간에 만들어지고, 만들어진 후에도 더 수정될 수 있다.

```

class 클래스명 :
    def 함수명1(인수):
        처리1
    def 함수명2(인수):
        처리2

class Ex_class:
    def __init__(self,data1,data2):
        self.r = data1
        self.l = data2

x = Ex_class(3.0,4.0)
x.r + x.l

7.0

```

```
class Rectangle:
    count = 0
    def __init__(self, width, height):
        self.width = width
        self.height = height
        Rectangle.count += 1

    def calcArea(self):
        area = self.width * self.height
        return area

r = Rectangle(2,3)

area = r.calcArea()
print("area = ", area)

r.width = 10
print("width = ", r.width)

print(Rectangle.count)
print(r.count)

area = 6
width = 10
1
1

class Animal:
    def __init__(self, name):
        self.name = name
    def move(self):
        print('move')
    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        print('bark')

class Duck(Animal):
    def speak(self):
        print('quack')

dog = Dog('doggie')
n = dog.name
dog.move()
dog.speak()

animals = [Dog('doggie'), Duck('duck')]
for a in animals:
    a.speak()
```

```
dog.speak
```

```
dd = Duck('duck')
dd.speak()
```

```
move
bark
bark
quack
quack
```

## ▼ 2.2.7 내장 함수

Python에서는 다양한 수학적 또는 통계적 함수를 사용할 수 있다.

표 2.10 Python에 내장된 np. 주요 수학 함수

Python 내장 수학 함수	내용
<code>sqrt(x)</code>	$x$ 의 제곱근
<code>sin(x)</code>	$\sin x$
<code>cos(x)</code>	$\cos x$
<code>tan(x)</code>	$\tan x$
<code>abs(x)</code>	$x$ 의 절댓값(absolute value)
<code>log(x)</code>	$\log_e x$
<code>log(x, a)</code>	$\log_a x$
<code>log10(x)</code>	$\log_{10} x$
<code>math.factorial(n)</code>	$n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$

```
import numpy as np

print(np.sqrt(2)) # 제곱근
print(np.sin(np.pi)) # 사인
1.4142135623730951
1.2246467991473532e-16

import math
print(math.log10(10)) # 로그
print(math.log(10,2))
print(math.log(10))

1.0
3.3219280948873626
2.302585092994046

np.exp(1) # 지수
```

```
2.718281828459045
```

```
print(abs(-3)) # 절댓값  
print(math.factorial(5)) # 계승함수  
  
3  
120
```

## 2.2.8 경고와 오류

NameError : 함수 이름에 대한 오류 메세지로 정확한 함수 이름을 사용하였는지 체크 필요  
 SyntaxError : 문법 에러 또는 파싱 에러로 오류의 위치에 대한 표시됨  
 nan : 계산할 수 없는 경우 (not a number 의 약자)  
 ZeroDivisionError : 계산이 정의되지 않는 경우  
 TypeError: 데이터 유형이 달라 계산이 정의되지 않는 경우  
 [] : 값이 없는 경우  
 SyntaxError : unexpected EOF while parsing : 명령문이 끝나지 않은 상태에서 [Enter] 키를 입력한 것

## ▼ 2.3 할당

나중에 다시 사용할 값에 이름을 붙이는 것

- 객체 이름을 지을 때 알파벳, 숫자, '\_'을 함께 사용할 수 있음
- 첫 문자로 숫자는 사용할 수 없음
- 할당연산자('=') 사용 가능

```
xconst_1 = 3 # 스칼라 할당  
print(xconst_1)
```

```
3
```

```
sample_list = [1,2,3,4,5] # 벡터 할당  
sample_list
```

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

```
sample_array = np.array([1,2,3,4]) # 벡터 할당  
print(sample_array)
```

```
[1 2 3 4]
```

```
print(sample_array+3) # 벡터 할당  
  
[4 5 6 7]
```

```
sample_array2 = np.array([[1,2,3,4,5],[6,7,8,9,10]]) # 2*5 행렬
print(sample_array2)

[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
```

```
sample_array2.shape
```

```
(2, 5)
```

## ▼ 데이터베이스 만들기

### ▼ 2.4.1 데이터 벡터

- numpy : 배열 데이터를 다루는 클래스를 주로 사용하며 행렬연산에 강한 패키지
- pandas : 데이터프레임 관리에 강력한 클래스를 주로 사용

```
import numpy as np
import pandas as pd
```

```
k_score = np.array([96,80,76,96,88,75,78,89,92,70]) # 데이터 벡터에 [], np.array([])을
k_score

array([96, 80, 76, 96, 88, 75, 78, 89, 92, 70])
```

```
k_score2 = np.array([67,80,79,80,36,45,88,79,93,90]) # 데이터 벡터에 [], np.array([])을
k_score2

array([67, 80, 79, 80, 36, 45, 88, 79, 93, 90])
```

```
score = np.append(k_score,k_score2)
score
```

```
array([96, 80, 76, 96, 88, 75, 78, 89, 92, 70, 67, 80, 79, 80, 36, 45, 88,
       79, 93, 90])
```

```
sc = np.hstack([k_score,k_score2]) # 수평 결합
print(sc)
```

```
[96 80 76 96 88 75 78 89 92 70 67 80 79 80 36 45 88 79 93 90]
```

### 2.4.2 데이터의 유형

float(), integer(), str(), list()

## ▼ 2.5 데이터베터에 함수 적용하기

### ▼ 2.5.1 반올림

올림: `math.ceil()`

내림: `math.floor()`

버림: `math.trunc()`

반올림: `round()`

```
import math
z = math.ceil(5.76543)
print(z)
z = math.trunc(5.76543)
print(z)
z = round(5.76543, 2) # 소수점 이하 2자리
print(z)

6
5
5.77
```

### ▼ 2.5.2 기초통계량

`scipy`: 통계량의 계산이나 기본적인 데이터 분석에 사용되는 함수가 있는 패키지

```
import scipy as sp
```

데이터 객체에 기초통계량을 구하기 위한 함수

`sum(합계)`, `mean(평균)`, `amax(최댓값)`, `amin(최솟값)`, `var(분산)`, `std(표준편차)`

```
import scipy as sp

%precision 3 # 자리수 지정

'%.3f'

k_score = np.array([96, 80, 76, 96, 88, 75, 78, 89, 92, 70])
sp.sum(k_score)

<ipython-input-30-f9689ea641c8>:2: DeprecationWarning: scipy.sum is deprecated
sp.sum(k_score)
```

840

```
len(k_score)
```

10

```
sum_value = np.sum(k_score)
n = len(k_score)
mu = sum_value/n
print(mu)
```

84.0

```
sp.mean(k_score)
```

```
<ipython-input-33-c2cf2aeelc5a>:1: DeprecationWarning: scipy.mean is deprecated
  sp.mean(k_score)
84.0
```

```
sp.var(k_score, ddof=0) # 최대우도분산추정량
```

```
<ipython-input-34-ff9a7cef57d5>:1: DeprecationWarning: scipy.var is deprecated
  sp.var(k_score, ddof=0)
78.6
```

```
sp.var(k_score, ddof=1) # 불편분산추정량
```

```
<ipython-input-35-04d7d675d6b6>:1: DeprecationWarning: scipy.var is deprecated
  sp.var(k_score, ddof=1) # 불편분산추정량
87.33333333333333
```

```
sp.var(k_score)
```

```
<ipython-input-36-07fca510c3b1>:1: DeprecationWarning: scipy.var is deprecated
  sp.var(k_score)
78.6
```

```
sigma2 = sp.var(k_score)
sp.sqrt(sigma2)
```

```
<ipython-input-37-82e943943751>:1: DeprecationWarning: scipy.var is deprecated
  sigma2 = sp.var(k_score)
<ipython-input-37-82e943943751>:2: DeprecationWarning: scipy.sqrt is deprecated
  sp.sqrt(sigma2)
8.8656641037206
```

```
sp.var(k_score, ddof=0)
```

```
<ipython-input-38-d59fe3870f48>:1: DeprecationWarning: scipy.var is deprecated
  sp.var(k_score, ddof=0)
78.6
```

```

sp.std(k_score,ddof=0)

<ipython-input-40-e98d23ee653c>:1: DeprecationWarning: scipy.std is deprecated
  sp.std(k_score,ddof=0)
8.8656641037206

sp.std(k_score,ddof=1)

<ipython-input-39-d8f1a976b50c>:1: DeprecationWarning: scipy.std is deprecated
  sp.std(k_score,ddof=1)
9.345230512584124

np.max(k_score)

96

np.min(k_score)

70

np.median(k_score)

84.0

k_score/np.std(k_score,ddof=1) # 표준화

array([10.273,  8.561,  8.132, 10.273,  9.417,  8.025,  8.347,  9.524,
       9.845,  7.49 ])

k_score/np.std(k_score,ddof=0) # 표준화 with unbiased var

array([10.828,  9.024,  8.572, 10.828,  9.926,  8.46 ,  8.798, 10.039,
       10.377,  7.896])

from scipy import stats
stats.scoreatpercentile(k_score,25) # 25 percentile

76.5

```

## ▼ 행 열별 함수 적용

`map()` 함수 : 데이터 객체에 대해 원하는 함수를 적용하는 편리한 기능 제공

```

list(map(len, [[1,2,3],[4,5,6]]))

[3, 3]

```

```
import pandas as pd
df = pd.DataFrame({'grp_col_1': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'],
                   'grp_col_2': [3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
                   'val_1': np.arange(10)})
```

df

	grp_col_1	grp_col_2	val_1
0	a	3	0
1	b	3	1
2	c	3	2
3	d	3	3
4	e	3	4
5	f	3	5
6	g	3	6
7	h	3	7
8	i	3	8
9	j	3	9

df.mean()

```
<ipython-input-52-c61f0c8f89b5>:1: FutureWarning: Dropping of nuisance columns
  df.mean()
  grp_col_2      3.0
  val_1         4.5
  dtype: float64
```

df.mean(axis=0) # 컬럼별 평균

```
<ipython-input-54-584a82ba156d>:1: FutureWarning: Dropping of nuisance columns
  df.mean(axis=0) # 컬럼별 평균
  grp_col_2      3.0
  val_1         4.5
  dtype: float64
```

df.mean(axis=1) # 행별 평균

```
<ipython-input-55-c1f7a3c447c6>:1: FutureWarning: Dropping of nuisance columns
  df.mean(axis=1) # 행별 평균
0    1.5
1    2.0
2    2.5
3    3.0
4    3.5
5    4.0
6    4.5
7    5.0
```

```
8      5.5
9      6.0
dtype: float64
```

## ▼ 2.5.4 순서함수

`sorted()` : 데이터에 대해 순서대로 정렬하기 위해 사용

- 오름차순 정렬 (default값)
- `reverse = True`

```
sorted([5,2,3,1,4])
```

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

```
np.sort([5,2,3,1,4])
```

```
array([1, 2, 3, 4, 5])
```

```
sorted([5,2,3,4,1],reverse=True)
```

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

```
names = ['Harry', 'Suzy', 'Joy', 'Rola']
sorted(names)
```

```
['Harry', 'Joy', 'Rola', 'Suzy']
```

```
df = pd.DataFrame({
    'name' : ['KIM', 'LEE', 'PARK', 'BAE', 'KANG'],
    'age': [24,32,43,24,np.nan],
    'height':[160,170,165,173,158]
})
df
```

	name	age	height
0	KIM	24.0	160
1	LEE	32.0	170
2	PARK	43.0	165
3	BAE	24.0	173
4	KANG	NaN	158

## ▼ rank

```
df['rank_average'] = df['age'].rank(method='min', ascending=False, na_option='bottom')
df
```

	<b>name</b>	<b>age</b>	<b>height</b>	<b>rank_average</b>
0	KIM	24.0	160	3.0
1	LEE	32.0	170	2.0
2	PARK	43.0	165	1.0
3	BAE	24.0	173	3.0
4	KANG	NaN	158	5.0

```
df['rank_average'] = df['age'].rank(method='max', ascending=False, na_option='bottom')
df
```

	<b>name</b>	<b>age</b>	<b>height</b>	<b>rank_average</b>
0	KIM	24.0	160	4.0
1	LEE	32.0	170	2.0
2	PARK	43.0	165	1.0
3	BAE	24.0	173	4.0
4	KANG	NaN	158	5.0

```
df['rank_average'] = df['age'].rank(method='average', ascending=False, na_option='top')
df
```

	<b>name</b>	<b>age</b>	<b>height</b>	<b>rank_average</b>
0	KIM	24.0	160	4.5
1	LEE	32.0	170	3.0
2	PARK	43.0	165	2.0
3	BAE	24.0	173	4.5
4	KANG	NaN	158	1.0

## ▼ sort\_values

```
dfObj = df.sort_values(by='name')
print("Contents of Sorted DataFrame based on a single column 'name' : ")
print(dfObj)
```

```
Contents of Sorted DataFrame based on a single column 'name' :
      name    age   height  rank_average
3     BAE  24.0     173        4.5
4    KANG    NaN     158        1.0
```

0	KIM	24.0	160	4.5
1	LEE	32.0	170	3.0
2	PARK	43.0	165	2.0

```
dfObj = df.sort_values(by='age')
print("Contents of Sorted DataFrame based on a single column 'age' : ")
print(dfObj)
```

Contents of Sorted DataFrame based on a single column 'age' :

	name	age	height	rank_average
0	KIM	24.0	160	4.5
3	BAE	24.0	173	4.5
1	LEE	32.0	170	3.0
2	PARK	43.0	165	2.0
4	KANG	NaN	158	1.0

```
dfObj = df.sort_values(by='height', ascending = False)
print("Contents of Sorted DataFrame based on a single column 'height' : ")
print(dfObj)
```

Contents of Sorted DataFrame based on a single column 'height' :

	name	age	height	rank_average
3	BAE	24.0	173	4.5
1	LEE	32.0	170	3.0
2	PARK	43.0	165	2.0
0	KIM	24.0	160	4.5
4	KANG	NaN	158	1.0

```
dfObj = df.sort_values(by=['age','height'])
print("Contents of Sorted DataFrame by 'age' then by 'height' : ")
print(dfObj)
```

Contents of Sorted DataFrame by 'age' then by 'height' :

	name	age	height	rank_average
0	KIM	24.0	160	4.5
3	BAE	24.0	173	4.5
1	LEE	32.0	170	3.0
2	PARK	43.0	165	2.0
4	KANG	NaN	158	1.0

## ▼ 2.6 구조적인 데이터 만들기

`arange(), tile()` : 구조 또는 패턴이 있는 데이터를 생성할 때 사용하는 함수

```
np.arange(start=1, stop=6, step =1)

array([1, 2, 3, 4, 5])
```

```
np.arange(start=0.1, stop=1.0, step=0.2)

array([0.1, 0.3, 0.5, 0.7, 0.9])
```

```

np.tile('A', 3)

array(['A', 'A', 'A'], dtype='<U1')

np.tile(0, 5)

array([0, 0, 0, 0, 0])

np.zeros(4)

array([0., 0., 0., 0.])

np.zeros([2,3]) # 2차원 배열

array([[0., 0., 0.],
       [0., 0., 0.]]) 

np.ones(3)

array([1., 1., 1.])

```

## ▼ 2.7 데이터베이스 다루기

```

x = np.array([100,120,130,124,150,167,170,163,160,155,145,157])
print(x[1])
print(x[1:3])
print(x[5:])
print(x[-1])
print(x[1:-3])
print(x[:3])

120
[120 130]
[167 170 163 160 155 145 157]
157
[120 130 124 150 167 170 163 160]
[100 120 130]

a = [1,2,3,4,5]
a

[1, 2, 3, 4, 5]

sample_df = pd.DataFrame({
    'col1' : [1,2,3,4,5],
    'col2' : [6,7,8,9,10],
    'col3' : ['a','b','c','d','e']
})

```

```
sample_df
```

	col1	col2	col3
0	1	6	a
1	2	7	b
2	3	8	c
3	4	9	d
4	5	10	e

## ▼ 2.8 벡터와 행렬 표현과 연산

행렬 : 숫자 또는 변수를 직사각형 또는 정사각형 모양으로 정렬한 배열

성분 : 배열된 수

- 행렬은 열벡터 또는 행벡터로 구성된다.

### ▼ 2.8.1 벡터 기본 연산

벡터의 합은 +를 이용하고 곱은 matmul()을 이용한다.

numpy에서 제공하는 두 함수 dot과 matmul은 2차원 행렬의 곱셈에서는 서로 같은 기능을 수행하나 고차원 배열 또는 텐서의 곱셈에서는 그 용법이 전혀 다르다.

```
x = np.array([1,2,3,4,5])
y = np.array([-1,-2,-3,-4,-5])

print(x+y)
```

[ 0 0 0 0 0 ]

```
x = np.array([1,2,3,4,5])
y = np.array([-1,-2,-3,-4,-5])
np.matmul(x.transpose(),y)
```

-55

```
np.matmul(y.transpose(),x)
```

-55

```
np.outer(x,x.T) # 외적을 계산하는 함수, 벡터 또는 행렬이 그 결과로 나옴
```

```
array([[ 1,   2,   3,   4,   5],
       [ 2,   4,   6,   8,  10],
```

```
[ 3,  6,  9, 12, 15],
[ 4,  8, 12, 16, 20],
[ 5, 10, 15, 20, 25]])
```

```
print(x*y)
print(x*x)
np.multiply(x,x)

[ -1 -4 -9 -16 -25]
[ 1  4  9 16 25]
array([ 1,  4,  9, 16, 25])
```

# x,y를 한 개의 열벡터로 만들기

```
print(np.r_[x,y])
print(np.hstack([x,y]))
np.concatenate((x,y),axis=0)

[ 1  2  3  4  5 -1 -2 -3 -4 -5]
[ 1  2  3  4  5 -1 -2 -3 -4 -5]
array([ 1,  2,  3,  4,  5, -1, -2, -3, -4, -5])
```

# 두개의 열벡터를 가진 행렬로 만들기

```
temp = np.column_stack([x,y])
print(temp)
np.c_[x,y]

[[ 1 -1]
 [ 2 -2]
 [ 3 -3]
 [ 4 -4]
 [ 5 -5]]
array([[ 1, -1],
       [ 2, -2],
       [ 3, -3],
       [ 4, -4],
       [ 5, -5]])
```

# 두개의 행벡터를 가진 행렬로 만들기

```
print(np.vstack([x,y]))
np.r_[[x],[y]]

[[ 1  2  3  4  5]
 [-1 -2 -3 -4 -5]]
array([[ 1,  2,  3,  4,  5],
       [-1, -2, -3, -4, -5]])
```

x\*x

```
array([ 1,  4,  9, 16, 25])
```

```
np.dot(x,y)
```

## ▼ 2.8.2 행렬 기본 연산

```
a = np.matrix([[1,2,3],
              [4,5,6]])
print(a)
print(a.transpose())

[[1 2 3]
 [4 5 6]]
[[1 4]
 [2 5]
 [3 6]]]

a = np.array([[1,0],[0,1]])
b = np.array([[1,2],[3,4]])
print(a+b)

[[2 2]
 [3 5]]

a = np.array([[1,0],[0,1]])
b = np.array([[1,2],[3,4]])

np.matmul(a,b)

array([[1, 2],
       [3, 4]])

a = np.matrix([[1,2,3],
              [4,5,6]])
a.shape

(2, 3)

a = np.array([[1,0],
              [0,1]])
b = np.array([1,2])
np.matmul(a,b)

array([1, 2])
```

## ▼ np.diag()

1차원, 2차원 array 만 허용

1차원 array를 받았을 경우 그 수를 가지고 대각선 행렬을 만들고

2차원 array를 받았을 경우 대각선에 위치한 수를 가지고 1차원 array를 만들 + k파라미터 값을 지정해서 대각선을 좌우로

```
np.diag([1 for _ in range(5)]) # np.diag() : 대각 원소를 추출하고 대각 행렬을 만들어낼 수 있는
array([[1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0],
       [0, 0, 0, 0, 1]])
```

## ▼ np.linalg.det 행렬식

역행렬이 존재하는지 여부를 확인하는 방법으로 행렬식(determinant, 줄여서 det)이라는 지표를 사용합니다.  
이 행렬식이 '0'이 아니면 역행렬이 존재하고, 이 행렬식이 '0'이면 역행렬이 존재하지 않습니다.

```
import numpy as np
a = np.matrix([[1,2,3],
               [4,5,6],
               [7,8,9]])
det = np.linalg.det(a)
print(det)
```

0.0

## ▼ 2.8.3 역행렬

정칙인 정방행렬 A에 대해 역행렬  $A^{-1}$ 가 유일하게 존재한다.

$$AA^{-1} = A^{-1}A = I$$

역행렬을 구하기 위해 numpy패키지의 np.linalg.inv()함수 사용

```
a = np.array([[1,2,3],
              [1,3,3],
              [1,2,4]])

print(a)
inv = np.linalg.inv(a) # 역행렬
print(inv)
```

```
[[1 2 3]
 [1 3 3]
 [1 2 4]]
 [[ 6. -2. -3.]
 [-1.  1.  0.]
 [-1.  0.  1.]]
```

```
s = np.array([[34.73, 15.66], [15.66, 378.60]])
print(s)
```

```
inv = np.linalg.inv(s) # 역행렬
inv

[[ 34.73 15.66]
 [ 15.66 378.6 ]]
array([[ 0.029, -0.001],
       [-0.001,  0.003]])
```

## ▼ 2.8.4 고유값과 고유벡터

```
G = np.array([[4,1],
              [1,2]])

w,v = np.linalg.eig(G)
print('Eigenvalues of matrix', w)
print('Eigenvalues of matrix', v)

Eigenvalues of matrix [ 4.414  1.586]
Eigenvalues of matrix [[ 0.924 -0.383]
 [ 0.383  0.924]]
```

행렬 공부

유튜브 : <https://www.youtube.com/watch?v=83UnOz6HiOY>

이상엽Math 강의록 :

[https://drive.google.com/file/d/1sAHAXH\\_IP1I3tUCXCF00ZPoBGbBC5kPj/view](https://drive.google.com/file/d/1sAHAXH_IP1I3tUCXCF00ZPoBGbBC5kPj/view)

## ▼ 2.9 배열

```
A = np.zeros((4,3,2))
print(A.shape)
print(A)

(4, 3, 2)
[[[0. 0.]
  [0. 0.]
  [0. 0.]]
 
 [[0. 0.]
  [0. 0.]
  [0. 0.]]
 
 [[0. 0.]
  [0. 0.]
  [0. 0.]]
 
 [[0. 0.]
  [0. 0.]
  [0. 0.]]]
```

```
A[0,:,:]
```

```
array([[0., 0.],
       [0., 0.],
       [0., 0.]])
```

```
em = np.empty((2,3))
print(em)
print(em.shape)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
(2, 3)
```

```
fu = np.full((2,4),3)
print(fu.shape)
fu
```

```
(2, 4)
array([[3, 3, 3, 3],
       [3, 3, 3, 3]])
```

```
b = np.arange(24).reshape(4,3,2)
b
```

```
array([[[ 0,  1],
        [ 2,  3],
        [ 4,  5]],

       [[ 6,  7],
        [ 8,  9],
        [10, 11]],

       [[12, 13],
        [14, 15],
        [16, 17]],

       [[18, 19],
        [20, 21],
        [22, 23]]])
```

```
b[1,1,:]
```

```
array([8, 9])
```

```
b[0,1,:]
```

```
array([2, 3])
```

```
e = np.eye(3,3) # 단위 행렬
e
```

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

## ▼ 2.10 반복문

### ▼ 2.10.1 for()를 이용한 루프

```
for i in range(1,5):
    print(i)
```

```
1
2
3
4
```

```
start = 100
end = 200
isum = 0
for i in range(start,end+1):
    isum+=i
print(isum)
```

```
15150
```

```
x = np.array([5,6,7,8])
n = len(x)
xx = np.tile(0,n)
for i in range(0,n):
    xx[i] = x[i] ** 2
    print(xx[i])
```

```
25
36
49
64
```

```
print(n)
```

```
4
```

```
np.tile(0,4)

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

```
transport = np.array(['bus','subway','car','bike'])
for vehicle in range(0,4):
    print(transport[vehicle])
```

```
bus
subway
car
bike
```

```
for i in range(1,3):
    for j in range(1,4):
        print(f'{i} * {j} = {i*j}')
```

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
```

## ▼ 2.10.2 while 이용한 루프

```
count = 0
while count < 5 :
    print(count, 'less than 5')
    count += 1
else:
    print(count, 'stop here')
```

```
0 less than 5
1 less than 5
2 less than 5
3 less than 5
4 less than 5
5 stop here
```

```
e = 30
```

```
while e > 1: # true
```

```
    e = e/3
```

```
    print(e)
```

```
10.0
3.333333333333335
1.111111111111112
0.3703703703703704
```

## ▼ 2.11 조건문

```

data = 5
if (data<10):
    print('it is less than 10')
else:
    print('it is equal or greater than 10')

it is less than 10

```

## ▼ 2.12 결측값

```

import pandas as pd
import numpy as np

b = np.arange(20).reshape(4,5)
df = pd.DataFrame(b,columns = [ 'a','b','c','d','e'])
df.isnull().values.any()

False

```

```

df.loc[1,['a','b']] = np.nan
df.loc[2,'c'] = np.nan
df.loc[3,:] = np.nan

```

```
df
```

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>
<b>0</b>	0.0	1.0	2.0	3.0	4.0
<b>1</b>	NaN	NaN	7.0	8.0	9.0
<b>2</b>	10.0	11.0	NaN	13.0	14.0
<b>3</b>	NaN	NaN	NaN	NaN	NaN

```

df_fillna = df.fillna(0)
print(df_fillna)

```

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>
0	0.0	1.0	2.0	3.0	4.0
1	0.0	0.0	7.0	8.0	9.0
2	10.0	11.0	0.0	13.0	14.0
3	0.0	0.0	0.0	0.0	0.0

```

df_missing = df.fillna('결측값')
df_missing

```

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>
<b>0</b>	0.0	1.0	2.0	3.0	4.0
<b>1</b>	NaN	NaN	7.0	8.0	9.0
<b>2</b>	10.0	11.0	7.0	13.0	14.0
<b>3</b>	10.0	11.0	7.0	13.0	14.0

```
df_f = df.fillna(method='ffill')
df_f
```

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>
<b>0</b>	0.0	1.0	2.0	3.0	4.0
<b>1</b>	0.0	1.0	7.0	8.0	9.0
<b>2</b>	10.0	11.0	7.0	13.0	14.0
<b>3</b>	10.0	11.0	7.0	13.0	14.0

```
df_b = df.fillna(method='bfill')
df_b
```

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>
<b>0</b>	0.0	1.0	2.0	3.0	4.0
<b>1</b>	10.0	11.0	7.0	8.0	9.0
<b>2</b>	10.0	11.0	NaN	13.0	14.0
<b>3</b>	NaN	NaN	NaN	NaN	NaN

df

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>
<b>0</b>	0.0	1.0	2.0	3.0	4.0
<b>1</b>	NaN	NaN	7.0	8.0	9.0
<b>2</b>	10.0	11.0	NaN	13.0	14.0
<b>3</b>	NaN	NaN	NaN	NaN	NaN

```
df_flimt = df.fillna(method='ffill', limit=1)
df_flimt
```

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>
<b>0</b>	0.0	1.0	2.0	3.0	4.0
<b>1</b>	0.0	1.0	7.0	8.0	9.0
<b>2</b>	10.0	11.0	7.0	13.0	14.0
<b>3</b>	10.0	11.0	NaN	13.0	14.0

```
df_fmean = df.fillna(df.mean())
df_fmean
```

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>
<b>0</b>	0.0	1.0	2.0	3.0	4.0
<b>1</b>	5.0	6.0	7.0	8.0	9.0
<b>2</b>	10.0	11.0	4.5	13.0	14.0
<b>3</b>	5.0	6.0	4.5	8.0	9.0

```
df_filla = df.fillna(df.mean()['a'])
df_filla
```

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>
<b>0</b>	0.0	1.0	2.0	3.0	4.0
<b>1</b>	5.0	5.0	7.0	8.0	9.0
<b>2</b>	10.0	11.0	5.0	13.0	14.0
<b>3</b>	5.0	5.0	5.0	5.0	5.0

```
df_fillab = df.fillna(df.mean()['a':'c']) # []안에 있는 열의 결측값들은 각 열의 평균으로 대체
df_fillab
```

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>
<b>0</b>	0.0	1.0	2.0	3.0	4.0
<b>1</b>	5.0	6.0	7.0	8.0	9.0
<b>2</b>	10.0	11.0	4.5	13.0	14.0
<b>3</b>	5.0	6.0	4.5	NaN	NaN

## ▼ 2.13 데이터프레임

python에서 모델링과 그래프 구현하는 함수를 사용하는데 필수적

각 열마다 다른 데이터 유형을 가질 수 있음

```
x = np.array([1,2,3,4,5])
sample_df = pd.DataFrame({
    'a':x,
    'b':x+2,
    'c':[str(x[0])+'j',str(x[1])+'j',str(x[2])+'j',str(x[3])+'j',str(x[4])+'j']
})
print(sample_df)
```

```
a   b   c  
0   1   3   1j  
1   2   4   2j  
2   3   5   3j  
3   4   6   4j  
4   5   7   5j
```

```
print(sample_df.a)
```

```
0    1  
1    2  
2    3  
3    4  
4    5  
Name: a, dtype: int64
```

```
print(sample_df.c)
```

```
0    1j  
1    2j  
2    3j  
3    4j  
4    5j  
Name: c, dtype: object
```

```
print(sample_df[['a','b']])
```

```
a   b  
0   1   3  
1   2   4  
2   3   5  
3   4   6  
4   5   7
```

```
print(sample_df.drop('b',axis=1))
```

```
a   c  
0   1   1j  
1   2   2j  
2   3   3j  
3   4   4j  
4   5   5j
```

```
print(sample_df.head(n=3))
```

```
a   b   c  
0   1   3   1j  
1   2   4   2j  
2   3   5   3j
```

```
print(sample_df.query('index == 0'))
```

```
a   b   c  
0   1   3   1j
```

```
print(sample_df.query('a==3'))
      a   b   c
2    3   5  3j

print(sample_df.query('c=="1j" | c=="3j"'))
      a   b   c
0    1   3  1j
2    3   5  3j

print(sample_df.query('a== 1 & c=="1j"'))
      a   b   c
0    1   3  1j

sample_df.query('a== 2')[['b', 'c']]
      b   c
1    4  2j
```

## ▼ 2.14 내장된 데이터셋 읽기

`load` 명령 : scikit-learn 설치 패키지에 같이 포함된 소량의 데이터

```
load_내장된 데이터셋 이름
# ex) load_boston
```

`fetch` 명령 : 데이터 크기가 커서 패키지에 저장되어 있지 않고, 인터넷에서 다운로드하여 홈 디렉토리 아래의 `scikit_learn_data`라는 서브 디렉토리에 저장 후 추후 불러들이는 대량의 데이터들

```
fetch_내장된 데이터셋이름
# ex) fetch_covtype
```

- `data` : (필수) 독립변수 배열
- `target` : (필수) 종속변수 배열
- `feature_names` : (옵션) 독립변수 이름 리스트
- `target_names` : (옵션) 종속변수 이름 리스트
- `DESCR` : (옵션) 자료에 대한 설명

```
from sklearn import datasets
```

```

iris = datasets.load_iris()

display(iris.data.shape, iris.data[:5])
iris.feature_names

(150, 4)
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']

```

```

from sklearn.datasets import fetch_california_housing
california = fetch_california_housing()
print(california.DESCR)

```

**.. \_california\_housing\_dataset:**

California Housing dataset

\*\*Data Set Characteristics:\*\*

:Number of Instances: 20640

:Number of Attributes: 8 numeric, predictive attributes and the target

:Attribute Information:

- MedInc median income in block group
- HouseAge median house age in block group
- AveRooms average number of rooms per household
- AveBedrms average number of bedrooms per household
- Population block group population
- AveOccup average number of household members
- Latitude block group latitude
- Longitude block group longitude

:Missing Attribute Values: None

This dataset was obtained from the StatLib repository.

[https://www.dcc.fc.up.pt/~ltorgo/Regression/cal\\_housing.html](https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html)

The target variable is the median house value for California districts, expressed in hundreds of thousands of dollars (\$100,000).

This dataset was derived from the 1990 U.S. census, using one row per census block group. A block group is the smallest geographical unit for which the U.S Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average number of rooms and bedrooms in this dataset are provided per household, these columns may take surprisingly large values for block groups with few households.

and many empty houses, such as vacation resorts.

It can be downloaded/loaded using the  
`:func:`sklearn.datasets.fetch_california_housing`` function.

`.. topic:: References`

- Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions, Statistics and Probability Letters, 33 (1997) 291–297

```
import pandas as pd
df = pd.DataFrame(california.data, columns = california.feature_names)
df.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.02
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-121.88
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-121.88
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-121.88

python에서 파일 및 디렉토리 경로에 관한 함수는 모두 os 모듈을 사용하기 때문에 os 모듈을 임포트해야 함

```
getcwd() # 현재 사용하고 있는 디렉토리 확인
```

```
chdir() # 디렉토리 변경
```

```
path.getsize() # 파일 크기 확인
```

## ▼ 2.16 데이터셋 합치기

merge 함수

```
import pandas as pd
singers = pd.DataFrame({
    'name' : ['justin bieber', 'giveon'],
    'song' : ['holy', 'heartbreak anniversary']
})
print(singers)
```

	name	song
0	justin bieber	holy
1	giveon	heartbreak anniversary

```
info_singers = pd.DataFrame({
    'name' : ['justin bieber', 'giveon'],
    'nationality' : ['Canada', 'USA'],
    'career_beginnings' : [2009, 2018]
})
print(info_singers)
```

	name	nationality	career_beginnings
0	justin bieber	Canada	2009
1	giveon	USA	2018

```
mer = pd.merge(singers,info_singers, left_on='name', right_on='name')
print(mer)
```

	name	song	nationality	career_beginnings
0	justin bieber	holy	Canada	2009
1	giveon	heartbreak anniversary	USA	2018

```
info_singers = pd.DataFrame({
    'name_of_singers' : ['justin bieber', 'giveon'], # 변수명 변경했을 때
    'nationality' : ['Canada', 'USA'],
    'career_beginnings' : [2009, 2018]
})
print(info_singers)
```

	name_of_singers	nationality	career_beginnings
0	justin bieber	Canada	2009
1	giveon	USA	2018

```
mer = pd.merge(singers,info_singers, left_on='name', right_on='name_of_singers')
print(mer)
```

	name	song	name_of_singers	nationality	\
0	justin bieber	holy	justin bieber	Canada	
1	giveon	heartbreak anniversary	giveon	USA	

	career_beginnings
0	2009
1	2018

```
mer.drop(['name_of_singers'], axis=1, inplace=True)
mer
```

	name	song	nationality	career_beginnings
0	justin bieber	holy	Canada	2009
1	giveon	heartbreak anniversary	USA	2018

```
actors = pd.DataFrame({
    'name' : ['이병헌', '송중기'],
    'movie' : ['오징어게임', '늑대인간']
})
```

```
print(actors)

actors_more_info = pd.DataFrame({
    'name_of_actors' : [ '이병현', '송중기', '최우식' ], # 변수명 변경했을 때
    'nationality' : [ '한국', '한국', '캐나다' ],
    'drama_series' : [ '미스터 선샤인', '재벌집 막내 아들', '그해 우리는' ]
})
print(actors_more_info)
```

	name	movie	
0	이병현	오징어게임	
1	송중기	늑대인간	
	name_of_actors	nationality	drama_series
0	이병현	한국	미스터 선샤인
1	송중기	한국	재벌집 막내 아들
2	최우식	캐나다	그해 우리는

```
mer1 = pd.merge(actors,actors_more_info, left_on=[ 'name' ],right_on=[ 'name_of_actors' ]
mer1
```

	name	movie	name_of_actors	nationality	drama_series
0	이병현	오징어게임	이병현	한국	미스터 선샤인
1	송중기	늑대인간	송중기	한국	재벌집 막내 아들
2	NaN	NaN	최우식	캐나다	그해 우리는

```
mer2 = pd.merge(actors,actors_more_info, left_on=[ 'name' ],right_on=[ 'name_of_actors' ]
mer2
```

	name	movie	name_of_actors	nationality	drama_series
0	이병현	오징어게임	이병현	한국	미스터 선샤인
1	송중기	늑대인간	송중기	한국	재벌집 막내 아들

## ▼ 2장 연습 문제

```
alphabet = [ 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n' ]

for i in alphabet:
    print('##( ' + i + ' )')

##(a)
##(b)
##(c)
##(d)
##(e)
##(f)
##(g)
##(h)
##(i)
```

```
##(j)
##(k)
##(l)
##(m)
##(n)
```

1번

```
##(a) 15
print(1+2*(3+4))
##(b)
print(1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6)
##(c) 7
print(np.sqrt((4+3)*(2+5)))
##(d)
print(((1+2)/(4+5))**3)
##(e)
print(122 + 12*23)
```

```
15
2.449999999999997
7.0
0.03703703703703703
398
```

```
##(f)
mulall = 1
for j in range(1,11):
    mulall *= j
print(mulall)
##(g)
print(np.sqrt(6**2+4))
##(h)
print(np.sin(60) + np.cos(30))
```

```
3628800
6.324555320336759
-0.15055917121463264
```

```
##(i)
print(np.log10(24)+np.log(10))
##(j)
print(np.sin(np.pi/4))
##(k)
print(np.cos(np.pi/3))
##(l)
print((1+2+3)/(4+5+6))
##(m)
tot = 0
for i in range(5,12):
    tot += i ** 2
print(tot)
```

```

##(n)
x = 15
y = 3

x2 = x **2
y3 = y **3

tot_2 = x+y
minus_2 = x -y

print(np.sqrt((3*x2+2*y3)/(tot_2*minus_2)))

3.682796334705652
0.7071067811865475
0.5000000000000001
0.4
476
1.8371173070873836

```

2번

```

x = np.array([2,3,5,7,9,10])
x2 = x**2
print('(a) ', x)
print('(b) ', x2)

tot = 0
for i in x2:
    tot += i
print('(c) ', tot)

print('(d) ', x-2)

print('(e) ', max(x),min(x))

x_up = x[x>5]
print('(f) ', x_up)

print('(g) ', len(x))
print('(h) ', np.matmul(x.transpose(),x))
print('(i) ', np.outer(x,x.transpose()))

xc=np.column_stack([x,x2])
print('(j) ', xc )

xr=np.vstack([x,x2])
print('(k) ', xr)

(a) [ 2  3  5  7  9 10]
(b) [ 4   9  25  49  81 100]
(c) 268
(d) [0  1  3  5  7  8]

```

```
(e) 10 2
(f) [ 7  9 10]
(g) 6
(h) 268
(i) [[ 4   6   10  14  18  20]
     [ 6   9   15  21  27  30]
     [ 10  15  25  35  45  50]
     [ 14  21  35  49  63  70]
     [ 18  27  45  63  81  90]
     [ 20  30  50  70  90 100]]
(j) [[ 2   4]
     [ 3   9]
     [ 5  25]
     [ 7  49]
     [ 9  81]
     [ 10 100]]
(k) [[ 2   3   5   7   9   10]
     [ 4   9  25  49  81 100]]
```

3번

```
A = np.matrix([[1,-1,4],
              [-1,1,3],
              [4,3,2]])

B = np.matrix([[3, -2, 4],
              [-2,1,0],
              [4,0,5]])

x = np.array([1,-2,4])

y = np.array([3,2,1])

print('(a) ', A + B)

print('(b) ', A.transpose())

matmul1 = np.matmul(x.transpose(),A)
matmul2 = np.matmul(matmul1,y)
print('(c) ', matmul2)

print('(d) ', np.matmul(x.transpose(),x))

x_1 = np.matmul(x.transpose(),A)
print('(e) ', np.matmul(x_1,x))

print('(f) ', np.matmul(x.transpose(),y))

print('(g) ', np.matmul(A.transpose(),A))

print('(h) ', np.matmul(A,B))
```

```
print('(i) ', np.matmul(y.transpose(),B))

print('(j) ', np.outer(x,x.transpose()))

print('(k) ', x+y)

print('(l) ', x-y)

k = x-y
print('(m) ', k.transpose())

print('(n) ', np.outer(x,y.transpose()))
print('(o) ', A - B)
print('(p) ', A.transpose() + B.transpose())

print('(q) ', (A + B).transpose())

print('(r) ', 3 * x)

print('(s) ', np.matmul(x.transpose(),y) **2 )
print('(t) ', np.matmul(B,A))

print('(u) ', np.linalg.inv(A))

(a) [[ 4 -3  8]
 [-3  2  3]
 [ 8  3  7]]
(b) [[ 1 -1  4]
 [-1  1  3]
 [ 4  3  2]]
(c) [[81]]
(d) 21
(e) [[25]]
(f) 3
(g) [[18 10  9]
 [10 11  5]
 [ 9  5 29]]
(h) [[21 -3 24]
 [ 7  3 11]
 [14 -5 26]]
(i) [[ 9 -4 17]]
(j) [[ 1 -2  4]
 [-2  4 -8]
 [ 4 -8 16]]
(k) [4 0 5]
(l) [-2 -4  3]
(m) [-2 -4  3]
(n) [[ 3  2  1]
 [-6 -4 -2]
 [12  8  4]]
(o) [[-2  1  0]
 [ 1  0  3]
 [ 0  3 -3]]
(p) [[ 4 -3  8]
 [-3  2  3]]
```

```
[ 8  3  7]
(q) [[ 4 -3  8
      [-3  2  3]
      [ 8  3  7]]
(r) [ 3 -6 12]
(s) 9
(t) [[21  7 14]
      [-3  3 -5]
      [24 11 26]]
(u) [[ 0.14285714 -0.28571429  0.14285714]
      [-0.28571429  0.28571429  0.14285714]
      [ 0.14285714  0.14285714  0.          ]]
```

**4번**

```
np.tile('a', 8)

array(['a', 'a', 'a', 'a', 'a', 'a', 'a', 'a'], dtype='<U1')

one = np.tile(1,3)
two = np.tile(2,3)
three = np.tile(3,3)
four = np.tile(4,3)
five = np.tile(5,3)

np.concatenate((one,two,three,four,five),axis=0)

array([1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5])

np.arange(1,100,step=2)

array([ 1,   3,   5,   7,   9,  11,  13,  15,  17,  19,  21,  23,  25,  27,  29,  31,  33,
       35,  37,  39,  41,  43,  45,  47,  49,  51,  53,  55,  57,  59,  61,  63,  65,  67,
       69,  71,  73,  75,  77,  79,  81,  83,  85,  87,  89,  91,  93,  95,  97,  99])
```

```
np.array([1,5,19,30])

array([ 1,   5,  19, 30])

np.arange(-10,11)

array([-10,   -9,   -8,   -7,   -6,   -5,   -4,   -3,   -2,   -1,    0,    1,    2,
       3,    4,    5,    6,    7,    8,    9,   10])
```

**5번**

```
x = np.arange(1,11)

len(x)
```

```
sum(x)
55

np.mean(x)
5.5

print(np.var(x,ddof=1))
print(np.std(x,ddof=1))

9.16666666666666
3.0276503540974917

odd = x[x%2!=0]
print(odd)

[1 3 5 7 9]

tot = 0
for i in range(len(x)):
    tot += x[i]/(i+1)

print(tot)

10.0
```

7번

```
x = np.array([-4.123,-3.556,1.634,2.213,3.875])
x

array([-4.123, -3.556,  1.634,  2.213,  3.875])

y = np.round(x,2)
y

array([-4.12, -3.56,  1.63,  2.21,  3.88])
```

x-y

```
array([-0.003,  0.004,  0.004,  0.003, -0.005])
```

```
np.round(x,1)
array([-4.1, -3.6,  1.6,  2.2,  3.9])
```

```
np.ceil(x)

array([-4., -3.,  2.,  3.,  4.])

np.trunc(x)

array([-4., -3.,  1.,  2.,  3.])
```

8번

```
data1={'name' : ['kim','lee','park','oh','yang','min','jung','moon'],
       'Korean': [93,76,87,92,98,75,82,92]}
data1=pd.DataFrame(data1)

data2={'name2' : ['kim','lee','park','oh','yang','min','jung','choi'],
       'English': [90,94,88,75,79,87,88,90]}
data2=pd.DataFrame(data2)

mer = pd.merge(data1,data2, left_on=['name'],right_on=['name2'],how='outer')
mer.drop('name2',axis=1,inplace=True)
mer.loc[8,'name']='choi'
mer
```

	name	Korean	English
0	kim	93.0	90.0
1	lee	76.0	94.0
2	park	87.0	88.0
3	oh	92.0	75.0
4	yang	98.0	79.0
5	min	75.0	87.0
6	jung	82.0	88.0
7	moon	92.0	NaN
8	choi	NaN	90.0

```
mer.sort_values('name')
```

	name	Korean	English
8	choi	NaN	90.0
6	jung	82.0	88.0
0	kim	93.0	90.0
1	lee	76.0	94.0

9번

```
/   1110011    92.0      11111
```

```
class1={'class': ['1','1','1','1','1','1','1','1'],
        'name' : ['kim','lee','park','oh','ang','min','jung','moon'],
        'Korean': [93,84,87,95,98,77,82,92]}
class1=pd.DataFrame(class1)
```

```
class2={'class': ['2','2','2','2','2','2','2','2'],
        'name' : ['kang','yun','park','cho','yang','min','jung','choi'],
        'Korean': [90,95,88,75,79,87,90,90]}
class2=pd.DataFrame(class2)
```

```
print(np.mean(class1.loc[:, 'Korean']))
print(np.sqrt(np.var(class1.loc[:, 'Korean'])))
```

```
print(np.mean(class2.loc[:, 'Korean']))
print(np.sqrt(np.var(class2.loc[:, 'Korean'])))
```

```
88.5
6.726812023536855
86.75
6.118619125260208
```

```
korean = np.concatenate([class1, class2], axis=0)
korean
```

```
korean = pd.DataFrame(korean, columns = ['class','name','score'])
korean
print(np.mean(korean.score))
print(np.sqrt(korean.score.var()))
```

```
87.625
6.701989754294367
```

```
korean = korean.sort_values('name')
korean
```

class	name	score
4	ang	98
11	cho	75
15	choi	90
6	jung	82
14	jung	90
8	kang	90
0	kim	93
1	lee	84
5	min	77
13	min	87
7	moon	92
3	oh	95
2	park	87

10번

```

for i in range(20,31):
    F = (9/5) * i + 32
    print(F)

68.0
69.80000000000001
71.6
73.4
75.2
77.0
78.80000000000001
80.6
82.4
84.2
86.0

```

11번

```

radius = [1,2,4,6,10]

for r in radius:
    a = np.pi * (r**2)
    print(a)

3.141592653589793
12.566370614359172
50.26548245743669

```

```
113.09733552923255  
314.1592653589793
```

12번

```
radius = [1.5, 3]  
  
for r in radius:  
    v = (4/3)*np.pi*(r**2)
```

