202404 from 202005

파이썬 머신러닝 실전 도장깨기

파이썬

머신러닝

실전 도장깨기

데이터의 수집부터 정제와
분석, 기계학습 분야
예측모델의 선정과 평가까지
실전으로 배우는 실습서

목 차

1장 머릿말

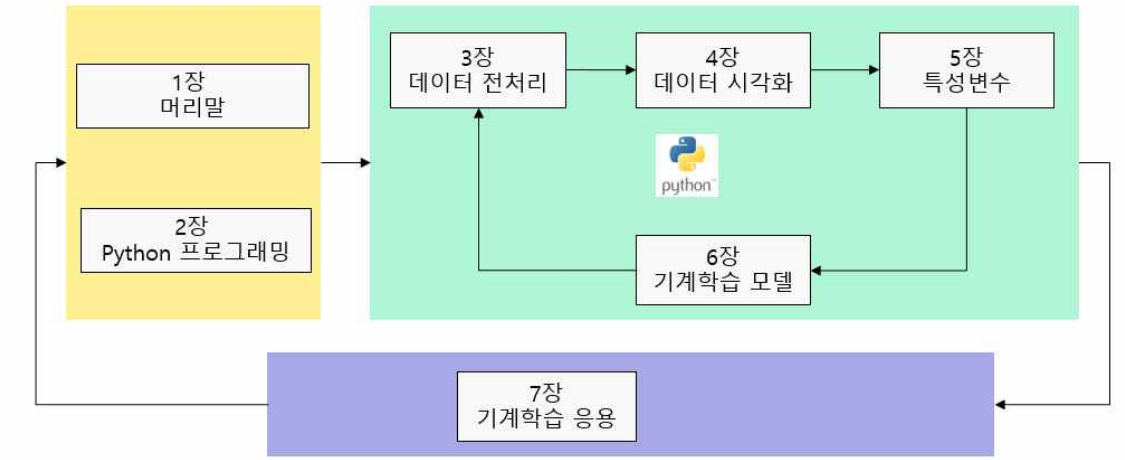
빅데이터와 인공지능을 핵심으로 하는 지능정보기술은 현재 우리 삶의 다양한 분야에 보편적으로 활용되어 새로운 가치를 창출하고 있다. 이 책의 특징은 실제 데이터를 수집, 획득하여 다듬는 과정부터 기계학습 모델을 적용하고 평가하는 전 단계에 필요한 핵심적인 파이썬의 명령과 문법을 설명하였다. 기초적인 파이썬의 문법과 명령어를 알거나 어느 정도 수준에 있는 독자를 대상으로, 기계학습 모형을 데이터에 적용하는 실전 지침서를 제시하고자 하였다. 제1장부터 제7장을 차례대로 실습하게 되면 데이터 분석역량, 통계 및 모델 이해력 등의 역량을 갖출 수 있게 될 것이다.

1.1. 책의 구성

필자가 직접 수행한 과제에서 얻은 경험, 시행착오, 해결 방안 등을 정리하여 핵심적인 사항을 골라 이 책의 내용을 구성하였습니다. 사전방식 나열형의 일방적 지식전달의 기존 방식과는 차별성을 두기 위해 하나의 데이터를 일관되게 사용하여 자료 처리와 예측까지 실전 데이터와 파이썬 코드로 구성하였습니다. 학교나 사무실에서 실제 데이터를 다루고 있는 독자는 이 책의 내용을 반복적으로 수행하고 익히게 되면 문제 해결 역량을 갖출 수 있다.

기계학습 각 단계에서 반드시 필요한 기초부터 심화수준의 사이킷런(scikit-learn) 모듈, 파이썬 코드와 명령어의 활용으로 단계적 학습이 가능하도록 하였다. 제1장부터 제6장까지 데이터를 일관되게 사용하여 기계학습의 전체 과정을 이론과 실무측면에서 이해할 수있게 스토리텔링(Stroy-telling)에 따라 구성하였다. 제1장과 2장에서는 파이썬의 기초적인 문법과 기능을 살펴보고 제3장부터 제6장은 데이터를 분석하고 예측에 영향을 미치는 특성변수를 최근접 이웃, 가우시안나이브, 로짓회귀분석, 랜덤포레스트, 서포트 벡터 머신 등을 최적화하여 적용하고 평가하는 과정으로 구성하였다.

예측모형 적용과 평가까지 데이터의 일관성을 유지하기 위해 한국방송진흥공사가 국내 소비자 모집단을 대상으로 무작위 층화표본 추출한 년도별 '소비자의 구매 행태조사' 자료를 바탕으로 구성하였다. 제6장까지 배운 지식을 활용하여 제7장에서는 소비자행태와 미국 시카고의 위생점검 불균형데이터에 대한 예측 방법을 제시하였다([그림 1]).



[그림 1. 책의 구성도]

1.2. Google colab

Jupyter Notebook은 명령어와 명령실행 결과 cell로 구분되며 명령어 cell 안에 커서가 있는 경우는 초록색으로 그 외에는 파란색으로 cell이 변하게 된다. 메뉴를 선택하여 Jupyter Nootebook의 명령어를 입력해도 되지만 Jupyter Notebook이 제공하는 단축키(hotkey)를 사용하면 아주 편리하다. 'Ctrl + Shift + P'를 치면 Jupyter notebook Hotkey를 모두 볼 수 있다. 단축키는 cell의 색깔이 파란색일 경우에만 실행 가능하다는 것을 기억하기 바란다. 필자가 자주 사용하는 단축키는 다음 몇 가지 정도이다.

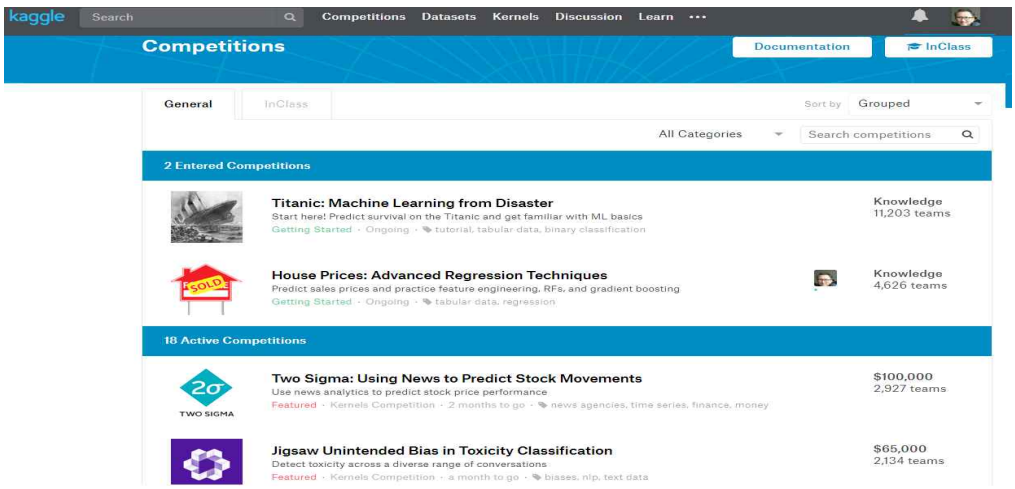
Jupyter notebook Hotkey	설명
A	현재 cell 위에 새로운 cell 을 추가
B	현재 cell 밑에 새로운 cell 을 추가
D + D	D를 연속해서 두번 누르면 현재 cell 을 삭제
X	해당 셀 오려두기
C	해당 셀 복사하기
V	오려거나 복사한 셀을 붙여넣기
M	Markdown 셀로 변환하며, Y는 Code 셀로 변환, R은 Raw Cell 로 변환
F	셀의 문자나 숫자를 찾아서 변경
O	실행결과 셀을 감추기(마우스를 짧게 두 번 클릭) 다시 한번 클릭하면 보여짐
Shift + M	여러 개의 셀을 합쳐버립니다.
Ctrl + Shift + minus(-)	현재 커서 위치를 기준으로 두 개의 셀로 구분
Ctrl + plus(+), Ctrl + minus(-)	화면의 크기를 조정

Jupyter Notebook에는 다양한 magic(% run) 명령문이 있는데 cell이나 파이썬 스크립트 한 줄 단위로 적용할 수 있다. 그림 기능을 자동으로 그려주는 %matplotlib, 윈도우 명령 프롬프트 화면을 cell안에서 직접 수행할 수 있는 %pip, %dirs, %cd 등과 파이썬 스크립트 한 줄의 실행 시간을 알려주는 %time 등이 있\다. 필요한 경우 자세한 설명과 예시는 구글링을 통해 확인하면 자세히 구체적으로 설명되어 있어 이 책에서는 생략하기로 한다.

1.3 기계학습 예측

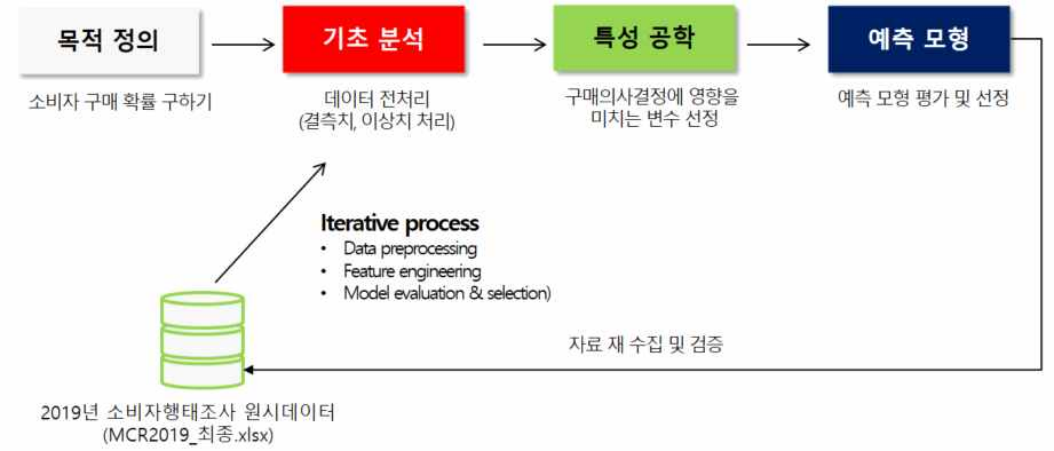
기계학습은 수학과 통계학(Math & Statistics), 컴퓨터과학(Computer science, Hacking Skills) 및 관련 부문 전문지식(Substantive Expertise) 등 개별적이고 독립적인 분야의 전문 역량이 필요하다. 빅데이터와 인공지능(Artificial Intelligence, AI)을 핵심으로 하는 지능정보 기술은 현재 우리 삶의 다양한 분야에 보편적으로 활용되어 새로운 가치를 창출하고 있다. 인공지능은 1956년에 컴퓨터가 인간과 같은 지능적인 방식으로 학습할 수 있도록 하는 것을 목적으로 학문의 한 분야로 생겨났다. 인공지능과 기존 컴퓨터 프로그램과의 차별성은 문제 해결방식에 있는데 인공지능은 사전에 설명이나 논리를 제시하지 않는 규칙기반 혹은 전문가 시스템(Rule-based or Expert systems)을 기반을 두고 있다, 1980년대부터 현재까지 주목을 받는 기계학습(Machine Learning)은 데이터를 바탕으로 컴퓨터가 학습할 수 있도록 하는 알고리즘과 관련 기술을 개발하는 분야로 대두되었다.

Kaggle(www.Kaggle.com)은 2010년에 설립된 플랫폼으로 기업과 단체가 당면한 문제를 상금과 함께 등록하면 참가자들이 예측모형을 개발하여 경쟁하는 포털이다. 최근 2017년에는 구글에 의해 인수되었으며, 타이타닉호 생존자 예측, 미국 아이오와주의 부동산가격 예측, 독일 카드회사 사기방지 등 관련 경쟁이 계속 진행 중에 있다([그림 4]). 데이터 전처리와 다양한 인공지능분야 모델의 적용 방법 등에 대한 파이썬 코드와 데이터가 모두 공개되어 있어 파이썬을 학습하는데 도움이 많이 된다.



[그림 4. Kaggle 홈페이지 화면]

인공지능의 일반적인 연구방법론

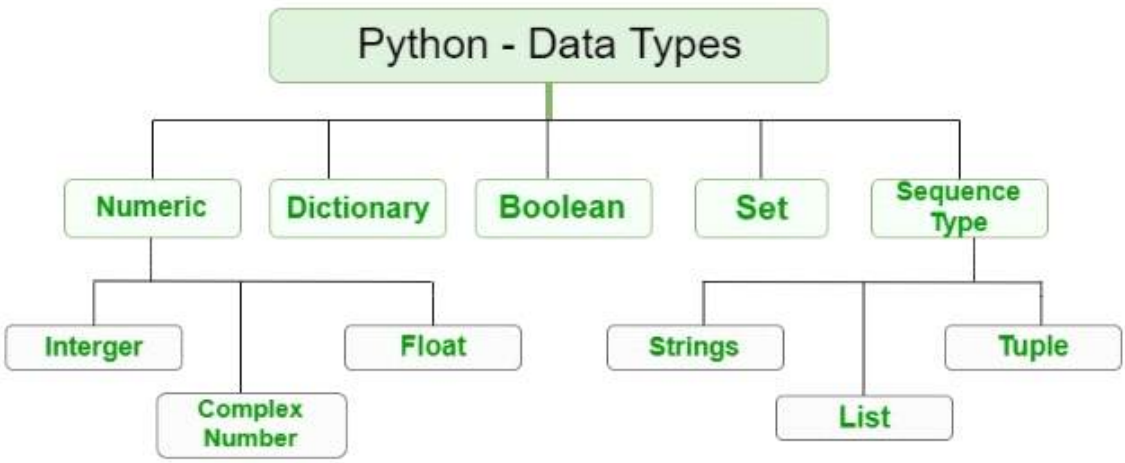


[그림 5. 기계학습 절차도]

2장 파이썬 프로그래밍

파이썬은 무료 오픈소스 프로그램 언어로 윈도우, 맥, 유닉스 등 다양한 OS(Operatin System)에서 작동 됩니다. 파이썬으로 작성한 소스 코드는 다른 프로그래밍 언어와는 다르게 컴파일이 필요 없이 직접 실행이 가능한 번역기(Intepreter)방식으로 아주 간편하다. 객체지향을 지원하며 전문가들이 사용하는 고급 언어로써 배우기 쉽고 사용하기 용이하다. 전 세계의 많은 사람들이 사용하며 구글, 페이스북, 인스타그램, 넷플랙스 등 주요 글로벌 기업에서 사용하고 있다. 기업에서 원하는 파이썬 언어의 수요[1]를 살펴보면 2019년도에 61,818개를 나타내고 2018년도 대비 124%가 증가하였으며 향후 매년 폭발적인 일자리 수요가 예상된다. 자 이제 핵심적인 파이썬 문법과 모듈(Numpy, Pandas)을 살펴보도록 하겠다.

[Python Data Types]



1) https://www.codingdojo.com/blog/the-7-most-in-demand-programming-languages-of-2019

[Python Topic Structure]

Task	Single Value	Multiple Values	numpy, pandas
Presentation (value, variable)	int, float, string, boolean	list, tuple, dictionary, set	ndarray, Series, DataFrame
Operation (algebra)	expressions	operations, mutable operations	expressions, get, set, reshape, ...
Control flow		if for while	
Use and reuse		Functions Standard libraries Modules and Packages	
Input and output		Standard I/O File I/O	Standard I/O File I/O CSV, Excel

2.1 연산과 자료형태

1 수치형 자료 연산

연산에는 더하기, 빼기, 곱하기, 나누기의 사칙연산과 참과 거짓을 구분하는 논리연산이 있다. '9'을 x 변수(객체)에 저장하고❶ 프린트하여❷ '9'이라는 것을 확인하였다. 데이터 유형은❸ 정수형(int)이다. '9'값이 저장된 x 변수의 컴퓨터 내부의 메모리 주소 값이 '140725344772496'이다❹. Print, type, id 등의 파이썬 명령어의 괄호 표시는 함수를 의미한다. 엑셀의 함수식과 같다고 생각하면 된다. 이제 x 변수에 '9.0'을 저장하면 x 변수가 '9.0' 값을 갖는 실수형으로 변경되고❺ 컴퓨터의 내부 메모리 주소가 바뀌었음을 확인할 수 있다❻. 여러 명령어를 한 줄에 작성하고 싶다면 세미콜론으로 명령어를 구분할 수 있으며 코드가 아닌 메모를 작성할 경우 '#'를 입력한 후 작성하면 된다.

```
x = 9                                                                    ❶
print(x)                                                                 ❷
print(type(x))                                                           ❸
print(id(x))                                                             ❹
x = 9.0                                                                  ❺
print(x); print(type(x)); print(id(x)); # ';' 는 여러 명령어를 한 줄에 기록   ❻
```

```
9
<class 'int'>
140725344772752
9.0
<class 'float'>
2170898317936
```

기존변수 x에 '5'를 더하는 연산을 한 후❶ '3'을 가감승제하고❷❸ '3'으로 나눌 경우에 나머지, 몫 ❹, 거듭제곱을❺ 확인하였다.

```
x = x + 5                                                    ❶
print(x+3); print(x-3)                                       ❷
print(x*3); print(x/3)                                       ❸
print(x%3); print(x//3)                                      ❹
print(x**2)                                                  ❺
```

```
17.0
11.0
42.0
4.666666666666667
2.0
4.0
196.0
```

변수 y를 '10'으로 만들고 y 값에 '1'을 추가한 후❶(y +=1은 y = y+1의 축약형) x, y를 확인하면 '12'와 '11'가 된다❷. 이제 변수를 서로 변경하여❸❹❺ 확인하면 x, y 값이 서로 바뀐 것을❻ 확인할 수 있다. x, y 변수의 값을 간단하게 변경하여❼ 확인할 수도❽ 있다.

```
y = 10; y += 1                                               ❶
print(x,y)                                                   ❷
temp = x                                                     ❸
x = y                                                        ❹
y = temp                                                     ❺
print(x,y)                                                   ❻
x, y = y, x                                                  ❼
print(x,y)                                                   ❽
```

```
12.0 11
11 12.0
12.0 11
```

2 논리형 자료 연산

x, y 값의 부등호 관련❶❷ 비교 연산과 등호 연산❸❹ 결과를 참과 거짓의 불리언(Boolean) 결과로 확인할 수 있습니다.

```
print(x > y); print(x < y)                                    ❶
print(x <= y); print(x >= y)                                  ❷
print(x == y); print(x != y)                                  ❸
print(x is y); print(x is not y)                              ❹
```

```
True
False
False
True
False
True
False
True
```

파이썬의 'IPython.core.interactiveshell'의❶ 패키지(Package)에서 'InteractiveShell' 모듈(Module)을 불러오면 한 셀의 모든 코드의 모든 실행 결과를 프린트할 수 있다❷. 파이썬의 모듈은 다양한 기능을 모아 놓은 것이고 패키지는 특정한 주제를 가진 모듈을 따로 구성하여 유용하게 쓸 수 있게 한 것이다.

```
from IPython.core.interactiveshell import InteractiveShell    ❶
InteractiveShell.ast_node_interactivity = 'all'               ❷
```

True라고 x 변수에 입력하고❶ 출력하여❷ 살펴보면 참과 거짓을 나타내는 불(Bool) 자료형 태라는❸ 것을 알 수 있다.

```
x = True                                                      ❶
x                                                             ❷
type(x)                                                       ❸
```

```
True
bool
```

True와 False의 교집합은❶❸ False이며 합집합은❷❹ True이다.

```
print(True & False)                                                    ❶
print(True | False)                                                    ❷
print(True and False)                                                  ❸
print(True or False)                                                   ❹
```

```
False
True
False
True
```

파이썬 프로그램에서 True는 '1', False는 '0'의 정수를 갖는다. 따라서 True에 '2'를 더하면❶ 2가 되고 '10'을 곱하면❷ 10이 되며 True와 True를 합산하면 '2'가❸ 된다. False에 '1'를 더하면 '1'이 되지만❹ '10'을 곱하면❺ '0'이 된다.

```
True + 1                                                               ❶
True*10                                                                ❷
True + True                                                            ❸
False + 1                                                              ❹
False*10                                                               ❺
```

```
2
10
2
1
0
```

3 문자형 자료

'파이썬 프로그래밍 언어는 쉽고 간결합니다.'라는 텍스트를 x 변수에 저장한 후❶ 프린트하고❷ 자료형태를 살펴보면 문자열(str)이라는 것을❸ 확인할 수 있다.

```
x = '파이썬 프로그래밍 언어는 쉽고 간결합니다.'                    ❶
x                                                                ❷
type(x)                                                          ❸
```

```
파이썬 프로그래밍 언어는 쉽고 간결합니다.
<class 'str'>
```

x 변수에서 '썬' 글자를 '선'으로 문자를 변경하거나❶ 단어 '파이썬'을 '파송송'으로 변경할 수❷ 있다. '언어'라는 단어가 x에 있는지 확인하면 'True'이고❸ 더하는 것도❹ 가능하다. 텍스트 x의 전체 문장길이는 공백을 포함하여 23개이고❺ 전체 문장에서 '썬'은 3번째로❻ 나타난다. 파이썬 코드 순번은 '0'부터 시작하기 때문에 첫 번째 문자는 '파'❼ 마지막 문자는 '다'❽이다. 마지막문자는 '-1'번이고 바로 앞의 '니'는 '-2'로 역순이다.

```
❶ x.replace('썬', '선')
❷ x.replace('파이썬', '파송송')
❸ '언어' in x
❹ x + x
❺ len(x)
❻ x.find('썬')
❼ x[0]
❽ x[-1]
```

```
'파이선 프로그래밍 언어는 쉽고 간결합니다.'
'파송송 프로그래밍 언어는 쉽고 간결합니다.'
True
'파이썬 프로그래밍 언어는 쉽고 간결합니다.파이썬 프로그래밍 언어는 쉽고 간결합니다.'
23
2
'파'
'다'
```

프린트(Print) 명령문과 '.format'을 사용하면 반복적인 키 입력을 줄일 수 있어 매우 유용하다. 자료 유형이 정수(9), 실수(1.5e5), 문자(파이썬)을❶ 각각 정수형태로 {0:d}, 소수점 이하 3자리의 실수형태로 {1:.3f}, 문자형태로 {2:s} format(9,1. 5e5, '파이썬') 괄호안의 순서대로 출력하였다❷. {0!s}는 print문에 전달된 값이 숫자 형식이라도 문자로 처리하여 출력하라는❸ 의미이다.

```
type(9), type(1.5e5), type('파이썬')                                          ❶
print('정수: {0:d}, 문자 : {1:.3f}, 실수 : {2:s}'.format(9, 5e5, '파이썬'))    ❷
print('수자이지만 문자로 출력 {0!s} 실수로 출력 {1:.2f}'.format(1.5e5, 1.5e5))  ❸
```

```
(int, float, str)
정수: 9, 문자 : 150000.000, 실수 : 파이썬
수자이지만 문자로 출력 150000.0 실수로 출력 150000.00
```

프린트문은 프로그램을 디버깅하는데 도움이 된다. 중간 중간의 결과물을 확인하기 위하여 임시로 코드 중간에 프린트문을 추가하여 적절히 사용할 수 있다. 초기 혹은 중간 계산 값의 결과 확인을 위해 필요한 위치에 print문을 추가하여 자주 사용하는 것을 권장한다. 코드가 정상적으로 작동하는지 확인했으면 해당 프린트문은 주석으로 처리하거나 삭제하면 된다.

유용한 표현식 f-string를 사용하여 문자열에 중괄호 {}를 이용해서 아주 간결하게 사용할 수 있다❶❷. 세미콜론(;)은 파이썬 명령어를 쭉 이어서 작성할 때 사용한다.

```
name ='조상구'; language = '영어'; sex ='남성'                                ❶
f'Hi, I am {name}, I like to use {language}. I am {sex}. Nice to meet you.'  ❷
```

```
'Hi, I am 조상구, I like to use 영어. I am 남성. Nice to meet you.'
```

정수와 실수형 상관없이 f-string를 사용하여 표현하였다❶❷.

```
x = 9; y = 10.24                                                ❶
print(f'{x}와 {y} 그리고 더하면 {x+y}')                          ❷
```

```
9와 10.24 그리고 더하면 19.240000000000002
```

4 리스트(List)형 자료

정수, 실수, 문자, 불(Bool)형태의 자료와 더불어 기계학습에서 주로 사용되는 리스트(List)형 자료를 살펴보자. 리스트(List)는 순서가 있는 자료의 나열이다. 자료가 없는 빈 리스트를❶ 만들 수도 있고 정수, 실수, 텍스트, 불(bool)형태의 어떠한 자료형태도 나열하여 리스트에 저장할 수❷❸❹ 있다.

```
x = []                                          ❶
x = [9, 9.0, '파이썬', True]                    ❷
x                                               ❸
type(x)                                         ❹
```

```
[9, 9.0, '파이썬', True]
list
```

문자열과 같이 리스트의 크기를❶ 확인할 수 있다. 리스트의 3번째 원소값을❷ 확인할 수 있는 인덱싱(indexing)과 두 번째 원소와 4번째 이전까지의 원소만을 골라❸ 리스트를 슬라이싱(Slincing)할 수 있다. '8'이라는 정수가 리스트에 있는지도 확인할 수❹ 있다.

```
len(x)                                          ❶
x[2]                                            ❷
x[1:3]                                          ❸
8 in x                                          ❹
```

```
4
'파이썬'
[9.0, '파이썬']
False
```

리스트에 자료를 추가할 수 있고❶❷ 새로운 리스트 변수 x를 만든 후에 리스트의 자료 값을 오름차순이나❸ 내림차순으로❹ 정렬할 수도 있다.

```
x.append('대한민국')                            ❶
x                                               ❷
x = [9,5,3,7,1,11]                              ❸
sorted(x)                                       ❹
sorted(x, reverse=True)
```

```
[9, 9.0, '파이썬', True, '대한민국']
[1, 3, 5, 7, 9, 11]
[11, 9, 7, 5, 3, 1]
```

5 딕셔너리(Dictionary)형 자료

사전(Dictionary)은 아주 강력하고 편리한 자료 구조이다. 자료가 없는 빈 사전을❶ 만들 수도 있고 키와 값의 쌍으로 구성하여 저장할 수❷❸❹ 있다.

```
x = {}   # x = dict()                                    ❶
x = {'a':1, 'b':3, 'c':5, 'd':7}                         ❷
x                                                        ❸
type(x)                                                  ❹
```

```
{'a': 1, 'b': 3, 'c': 5, 'd': 7}
dict
```

리스트와 같이 사전의 크기를❶ 확인할 수 있고 사전의 'a'키의 값과❷ 자료를 추가하여❸ 확인하였다❹. 사전형 자료의 전체 내용과❺ 키와❻ 해당 값을❼ 각각 확인할 수 있다.

```
len(x)                                                   ❶
x['a']                                                   ❷
x['e']= 9                                                ❸
x                                                        ❹
x.items()                                                ❺
x.keys()                                                 ❻
x.values()                                               ❼
```

```
5
1
{'a': 1, 'b': 3, 'c': 5, 'd': 7, 'e': 9}
dict_items([('a', 1), ('b', 3), ('c', 5), ('d', 7), ('e', 9)])
dict_keys(['a', 'b', 'c', 'd', 'e'])
dict_values([1, 3, 5, 7, 9])
```

사전형 자료의 키 값을 기준으로 내림차순으로 정렬할 수도❶ 있다.

```
sorted(x.items(), key=lambda kv: kv[0], reverse=True)    ❶
```

```
[('e', 9), ('d', 7), ('c', 5), ('b', 3), ('a', 1)]
```

다른 자료 형태인 세트(Set), 튜플(Tuple), 날짜형(time-series) 등의 자료 형태는 이 책을 학습하면서 이해하도록 하겠다.

6 조건문과 반복문

조건문은 특정한 조건이 충족된 경우에 프로그램이 작동되는 명령문이다. 일반적으로 Boolean를 사용하여 실행한다. 참(True)값을 변수 'is_opensource'저장하고❶ 참인 경우에만❷ '파이썬은 무료 오픈 소스이다.'라는 문장을 출력하고❸ 그렇지 않은 경우에는❹ '파이썬은 무료 오픈 소스가 아니다.'의 문장을❺ 출력하였다.

```
is_opensource = True                                    ❶
if is_opensource  :                                     ❷
    print('파이썬은 무료 오픈 소스이다.')                  ❸
else:                                                   ❹
    print('파이썬은 무료 오픈 소스가 아니다.')             ❺
```

```
파이썬은 무료 오픈 소스이다.
```

다른 자료형태를 사용하여 조건문을 수행할 수도 있는데 사전형 자료를 사용하여❶❷ A인 경우는 공집합으로 False로 간주하게 되어❸ 'The Object A is not empty'를 인쇄하고❹ 그렇지 않은 경우는 True로 간주하여❺ 'The Object A is empty'를❻ 프린트하게 된다. '파이썬'이 B의 원소에 있으면❼ 'B에 파이썬이 있다.'❽, A의 원소에 없으면❾ 'A에 파이썬이 없다.'를❿ 인쇄한다.

```
❶ A = {}
❷ B = {'파이썬':10, '자바':5}
❸ if A :
❹     print("The Object A is not empty")
❺ else :
❻     print("The Object A is empty.")

❼ if '파이썬' in B:
❽     print('B에 파이썬이 있다.')
❾ if '파이썬' not in A:
❿     print('A에 파이썬이 없다.')
```

```
The Object A is empty.
B에 파이썬이 있다.
A에 파이썬이 없다.
```

반복문은 순서가 있는 객체와 반복이 가능한 객체를 순차적으로 하나씩 읽으면서 순회 동작을 한다. 리스트(list), 문자(str), 튜플(tuple), range(), zip(), map() 등의 자료형태는 객체의 원소를 하나씩 차례 순서대로 반환과 반복이 가능하다(Iterable). '0'부터 '10' 미만의 연속된

정수를 만들어 주는 range()함수를 반복문을❶ 통해 하나씩 읽어와 출력할 수 있다❷.

```
for i in range(0,10):                                    ❶
    print(i)                                             ❷
```

```
0
1
2
3
4
5
6
7
8
9
```

리스트축약문(List Comprehension)은❶❷ 파이썬에서 For 반복문을 간결하고 간단하게 사용하는 명령문이다. 1부터 10까지의 정수의 합계를 구할 수 있다❸.

```
[i for i in range(0,10)]                                 ❶
[i for i in reversed([i for i in range(1,11)])]          ❷
sum([i for i in reversed([i for i in range(1,11)])])     ❸
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
55
```

Tip 1. 리스트 축약문(List comprehension)

① 일반적인 Python 코드

```
mylist =[]                    # list 형태의 빈 객체를 mylist로 저장
for x in np.arange(10) :      # 0부터 9까지 x로 차례로 읽어
    if x >=5:                 # x 값이 5이상이면
        mylist.append(x)      # mylist의 list형태에 차곡 차곡 저장
mylist                        # 결과값 [5, 6, 7, 8, 9]
```

② List comprehension형태의 Python 코드

```
[ x for in [1, 2, 3, 'A'] if x >=2] #list comprehension type 결과 값 [5, 6, 7, 8, 9]
```

반복문을 통해 1부터 10까지의 정수를 하나씩 읽어❶ 2로 나눈 값의 나머지가 0이면❷ 짝수를 인쇄하고❸ 그렇지 않은 경우는❹ 홀수라고 인쇄하였다❺.

```
for i in range(1, 11):                                  ❶
    if i%2 == 0:                                        ❷
        print('{}:짝수'.format(i))                       ❸
    else:                                               ❹
        print('{}:홀수'.format(i))                       ❺
```

```
1:홀수
2:짝수
3:홀수
4:짝수
5:홀수
6:짝수
7:홀수
8:짝수
9:홀수
10:짝수
```

리스트축약문을 사용하여 1부터 10까지의 짝수와❶ 홀수를❷ 간단하게 나타내었다.

```
[i for i in range(10) if i%2 == 0]                      ❶
[i for i in range(10) if i%2 != 0]                      ❷
```

```
[0, 2, 4, 6, 8]
[1, 3, 5, 7, 9]
```

리스트축약이외에 객체가 반복이 가능한(Iterable) 변수인 집합과 딕셔너리의 축약이 가능하다. 리스트 축약은 대괄호를 사용하지만 집합 및 딕셔너리 축약은 중괄호를 사용하며 모든 축약문은 if-else문 같은 조건부 논리를 포함하여 사용할 수 있다. 파이썬 프로그래밍을 제어하는 명령어는 if, for 이외에 while, break. continue 등이 있는데 사용 빈도가 적어 해당 제어문은 이 책의 내용을 실습하면서 이해하도록 하겠다.

2.2 Numpy와 Pandas

Numpy(Numerical Python)는 C언어로 구현된 파이썬 라이브러리로써, 고성능의 수치계산에 적합하여 벡터 및 행렬 연산에 있어서 매우 편리한 기능을 제공한다. 넘피(Numpy)는 기본적으로 array라는 단위로 데이터를 관리하고 연산을 수행하는 행렬(matrix)로 이해하면 된다. 우선 넘피를 사용하기 위해서는 모듈을 'np'라는 이름으로 읽으면❶ 관련 라이브러리를 사용할 수 있게 된다. 모듈과 패키지는 여러 코드를 묶어 재사용할 수 있도록 만든 코드 모음으로 일반적으로 비슷한 기능을 하는 함수나 이보다 큰 기능의 함수와 데이터를 포함한다.

```
import numpy as np                                                    ❶
```

'0'부터 '10'까지 '1'만큼씩 증가시켜 리스트를❶ 만들고 np.array()를 사용하여 numpy 행렬을❷ 생성하였다. Numpy의 arange() 함수를 사용하여 만든 행렬과❸ '2' 만큼씩 증가시킨 행렬을❹ 만들고 리스트 자료 형태로❺ 변경하였다.

```
list(range(10)) # list                                                ❶
np.array(range(0,10))                                                 ❷
np.arange(10)                                                         ❸
np.arange(0,10,2)                                                     ❹
list(np.arange(0,10,2))                                               ❺
```

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

파이썬의 리스트는 여러 가지 자료 형태의 원소를 넣을 수 있지만, 메모리 용량이 크고 속도가 느리며 전체적인 연산이 불가능하다. 반면에 Numpy array는 동일한 자료 형태의 원소를 갖고 메모리 최적화하여 연산 속도가 빠르며 전체 연산이 가능하다. 파이썬에는 랜덤시드(random seed)라는 것이 있는데 사실 랜덤처럼 보이지만 정해진 알고리즘에 의해 생성되는 규칙적인 순열로 시작점이 정해기기 때문에 랜덤 함수를 사용하면 정해진 숫자가 나온다.

필자와 독자의 코드 결과를 동일하게 재현하기 위해 무작위시드(seed)를❶ 설정한 후 '1' 부터 '100'까지 12개의 무작위 복원추출한 배열을 만들면❷ '52', '71', '72'등의 동일한 12개 자료가 무작위 추출된 것을❸ 확인할 수 있다. 최대와 최소값을 확인하고❹❺ 10개 자료의

합계❻, 평균❼, 표준편차를❽ 확인하였다. 최소값 '19'와 최대값 '80'를 가진 원소의 배열 인덱스는 각각 11번째와❾ 3번째인 것을❿ 알 수 있다.

```
❶ np.random.seed(1357)
❷ sample_matrix = np.random.randint(1,100,12)
❸ sample_matrix

❹ sample_matrix.max()
❺ sample_matrix.min()
❻ sample_matrix.sum()
❼ sample_matrix.mean()
❽ sample_matrix.std()
❾ sample_matrix.argmin()
❿ sample_matrix.argmax()
```

```
array([52, 71, 72, 80, 72, 53, 71, 46, 68, 59, 55, 19])
80
19
718
59.833333333333336
15.836841716565697
11
3
```

배열 sample_matrix의 차원은 12개의 원소를 가진 1차원 배열이며❶ 3개의 행과 4개의 열로❷ 재구성된 배열의 차원은 (3, 4)인 것을❸ 알 수 있'다.

```
sample_matrix.shape                                    ❶
sample_matrix.reshape(3,4)                             ❷
sample_matrix.reshape(3,4).shape                       ❸
```

```
(12,)
array([[52, 71, 72, 80],
       [72, 53, 71, 46],
       [68, 59, 55, 19]])
(3, 4)
```

벡터를❶ 재배열하여(reshape(-1,1)) 원소의 개수만큼의 행렬로❷ 만든다. 기계학습 분야의 모든 모델이 행렬형태의 입력변수를 요구하기 때문에 기억하여야 할 기능이다.

```
sample_matrix.shape                                    ❶
sample_matrix.reshape(-1,1).shape                      ❷
```

```
(12,)
(12, 1)
```

배열의 3번째와❶ 마지막 원소의❷ 값은 각각 '80', '19'이다. 행과 열의 (3,4)로 재배열된 첫 번째와 두 번째 행의 행렬과❸ 전체 행과 두 번째 세 번째 열의 행렬❹을 인덱싱(Indexing)

하여 확인할 수 있다. 전체 행과 세 번째 열까지의 행렬❺, 두 번째와 세 번째 행과 열의 원소값❻, 처음 행부터 첫 번째 행과 두 번째 열부터 마지막 열까지의 행렬을❼ 슬라이싱(Slicing)할 수 있다.

```
❶ sample_matrix[3]
❷ sample_matrix[-1]
❸ sample_matrix.reshape(3,4)[0:2] # sample_matrix.reshape(3,4)[[1,2]]
❹ sample_matrix.reshape(3,4)[:, [1,2]]
❺ sample_matrix.reshape(3,4)[:, 0:3]
❻ sample_matrix.reshape(3,4)[1][2]
❼ sample_matrix.reshape(3,4)[:2, 2:]
```

```
80
19
array([[52, 71, 72, 80],
       [72, 53, 71, 46]])
array([[52, 71, 72],
       [72, 53, 71],
       [68, 59, 55]])
71
array([[72, 80],
       [71, 46]])
array([[72, 53, 71, 46],
       [68, 59, 55, 19]])
array([[71, 72],
       [53, 71],
       [59, 55]])
```

판다스(Pandas)는 데이터 정제, 분석 등에 가장 많이 사용되는 데이터분석 라이브러리로, 행과 열로 이루어진 데이터 객체를 만들어 마치 엑셀을 사용하는 것과 유사하게 데이터를 처리할 수 있다. 판다스는 대용량의 데이터를 안정적이고 빠르게 처리하는데 매우 편리한 장점을 갖고 있다. 판다스 모듈을 pd라는 이름으로 읽어❶ 행렬을 객체화하여❷ 판다스의 데이터프레임(DataFrame)으로❸ 만들었다.

```
import pandas as pd                                  ❶
data = sample_matrix.reshape((3,4))                  ❷
pd.DataFrame(data)                                   ❸
```

	0	1	2	3
0	52	71	72	80
1	72	53	71	46
2	68	59	55	19

엑셀의 행과 열의 이름을❶❷ 지정하여❸ 데이터프레임을❹ 만들면 엑셀 형태와 똑같이 쉽게 사용할 수 있다. 대용량의 엑셀파일을 파이썬 코드로 효과적으로 다룰 수 있게 되었다고 생

각하면 된다.

```
index = range(1,4)                                                    ❶
columns = ['A','B','C','D']                                           ❷
df = pd.DataFrame(data, index = index, columns = columns)             ❸
df                                                                    ❹
```

	A	B	C	D
1	52	71	72	80
2	72	53	71	46
3	68	59	55	19

엑셀과는 달리 눈에 보이지 않는 상태에서 데이터를 핸들링하면 처음에는 낯설겠지만, 주요 기능을 연습하여 익히면 아주 유용한 도구임을 알 수 있게 된다. 판다스의 주요 기능은 엑셀의 주요 기능과 같이 조건문(filitering), 피봇팅(pivoting)이다. 세부 기능은 2.3절에서 다루어 보기로 하고 여기서는 인덱싱과 조건문에 대해서만 살펴보도록 하겠다.

특정 행의 데이터만 보고 싶으면 loc() 명령을 사용하여 '2번' 인덱싱 행을 지정하거나❶ 인덱싱을 행렬로 간주하여 iloc() 명령을 사용하여 '1'을 지정하면❷ 두 번째 행의 값을 확인할 수 있다. 특정한 자료 값을 확인하기 위해 loc()와 iloc()를❸❹ 모두 사용할 수 있다.

```
df.loc[2]                                                             ❶
df.iloc[1]                                                            ❷
df.loc[2, 'B']                                                        ❸
df.iloc[1, 1]                                                         ❹
```

```
A    72
B    53
C    71
D    46
Name: 2, dtype: int32

A    72
B    53
C    71
D    46
Name: 2, dtype: int32

53

53
```

특정 열만 지정하여❶ 데이터를 확인하면 자료 형태가 pandas 시리즈(Series)인 것을❷ 알 수 있다. 하나 이상의 열을 지정하여❸ 자료형태를 확인하면 데이터프레임이라는❹ 것을 확

인할 수 있다. 데이터프레임은 각각의 시리즈 자료가 세로로 쌓아져 생성된다고 이해하면 된다. ‘A’와 ‘B’ 두 개 열의 ‘2’번째 행 인덱스의 데이터를❺ 확인하였다.

```
df['A']                                  ❶
type(df['A'])                            ❷
df[['A','B']]                            ❸
type(df[['A','B']])                      ❹
df[['A','B']].loc[2]                     ❺
```

```
1    52
2    72
3    68
Name: A, dtype: int32

pandas.core.series.Series
```

	A	B
1	52	71
2	72	53
3	68	59

```
pandas.core.frame.DataFrame

A    72
B    53
Name: 2, dtype: int32
```

Tip 1. 판다의 자료 구조인 시리즈(Series)와 데이터프레임(Dataframe)

판다(Panda)의 자료구조는 성별, 연령, 직업 등 각 열의 데이터 형태인 시리즈(Series)가 복수 개로 합쳐져 엑셀의 스프레드시트 형식의 데이터프레임(Dataframe)

데이터프레임의 각 열(column),은 서로 다른 종류의 값(숫자, 문자열, Boolean 등)만을 가질 수 있지만 하나의 Series는 동일한 자료형

데이터프레임은 Series인 벡터(Vector)가 옆으로 차곡 차곡 쌓인 행렬(Matrix)

판다스의 시리즈(Series)는 1차원 배열의 값에 대응되는 인덱스를 부여할 수 있는 구조이고 데이터프레임(DataFrame)은 2차원으로 행방향 인덱스와 열방향 컬럼을 갖는 레이블이 있는 엑셀 형태의 자료 구조이다.

이제 ‘60’ 이상인 자료를 확인하여❶ True인 경우의 자료만 가져와 False인 경우는 NaN값을❷ 가져온다. ‘60’ 이상인 Bool형태의 행렬에 1을 곱하여 ‘1’과 ‘0’의 ‘행렬을 만든 후❸ 인덱스와 열을 지정하여 데이터프레임으로❹ 만들었다.

```
df>=60                                                                  ❶
df[df>=60]                                                              ❷
np.array(df>=60)*1  #  (df>=60).values*1                                ❸
pd.DataFrame(np.array(df>=60)*1, index = index, columns = columns)      ❹
```

	A	B	C	D
1	False	True	True	True
2	True	False	True	False
3	True	False	False	False

	A	B	C	D
1	NaN	71.0	72.0	80.0
2	72.0	NaN	71.0	NaN
3	68.0	NaN	NaN	NaN

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

	A	B	C	D
1	0	1	1	1
2	1	0	1	0
3	1	0	0	0

C열의 자료가 '60'이 넘고❶ B와 C의 두 개 열의 값이 각각 50과 70을 동시에 넘는 조건을 충족하는 데이터프레임을❷ 살펴보았다.

```
df[df['C']>60]                                                          ❶
df[(df['B']>50)&(df['C']<70)]                                           ❷
```

	A	B	C	D
1	52	71	72	80
2	72	53	71	46

	A	B	C	D
3	68	59	55	19

DataFrame을 복사할 때는 '='기호를 이용해서 복사하면 실제 데이터의 내용이 복사되는 것이 아니라 데이터의 메모리 위치만 복사되는 것이기 때문에 원본 자료는 하나만 있게 된다. 데이터의 내용까지 복사하여 저장한 후❶ 새로운 열 'D'을 만든 후에❷❸ 컬럼별로 NaN의 개수❹, 고유한 값의 개수❺, 자료 형태를❻ 살펴보았다. A열 결측치인 NaN의 개수는 1 개

이고, 고유 값은 '72.0'과 '68.0' 2개의 실수형 자료이다. 열 'E'는 결측치가 없고 고유값은 'owl'과 'fox'인 object형 자료이다.

```
df = df[df>60].copy()                                    ❶
df['E'] = ['owl', 'fox', 'owl']                          ❷
df                                                       ❸
df.isnull().sum()                                        ❹
df.nunique()                                             ❺
df.dtypes                                                ❻
```

	A	B	C	D	E
1	NaN	71.0	72.0	80.0	owl
2	72.0	NaN	71.0	NaN	fox
3	68.0	NaN	NaN	NaN	owl

```
A    1
B    2
C    1
D    2
E    0
dtype: int64

A    2
B    1
C    2
D    1
E    2
dtype: int64

A    float64
B    float64
C    float64
D    float64
E     object
dtype: object
```

데이터프레임이 생성될 때마다 결측치와 고유값의 개 수, 자료형태 등을 매번 확인하려면 번거롭다. 이런 경우에는 사용자가 직접 사용자 정의함수를 만들어 재사용할 수 있다. 이 책에서 이미 다루었던 type(), sum(), reverse() 등은 이미 파이썬에 내장되어있는 함수입니다. 동일한 코드를 반복해서 사용하고자 한다면 관련 코드를 사용자가 직접 함수로 정의하여 사용한다. 데이터프레임의 결측치 수❶, 고유값의 개수❷, 자료 형태❸ 등을 확인하면 모두 시리즈로 각각 5개의 원소를 가진다는 사실을 알 수 있다.

```
type(df.isnull().sum()); df.isnull().sum().shape ❶
type(df.nunique()); df.nunique().shape ❷
type(df.dtypes); df.dtypes.shape ❸
```

```
pandas.core.series.Series

(5,)

pandas.core.series.Series

(5,)

pandas.core.series.Series

(5,)
```

사용자 정의함수를 만들기 위해서는 코드 맨 앞에 함수를 만드는 명령어 'def'로 함수의 이름을 show(df)라고 선언한 후❶ 괄호를 열고 닫은 후, 콜론을 입력한다. 자료 형태❷, 결측치 수❸, 고유값 개수❹의 시리즈를 'tf'라는 데이터프레임으로 만들어 반환한다❺. 사용자 정의함수 show()는 데이터프레임을 입력 인자로 받아 각 열의 자료 형태, 결측치와 고유값의 개수를 데이터프레임 형태로❻ 반환한다.

```
def show(df) : ❶
    tf = pd.DataFrame({'데이터형태(dtypes)' : df.dtypes, ❷
                       '결측치 수(null)' : df.isnull().sum(), ❸
                       '고유값 수(nunique)' : df.nunique()}) ❹
    return tf ❺

show(df) ❻
```

	데이터형태(dtypes)	결측치 수(null)	고유값 수(nunique)
A	float64	1	2
B	float64	2	1
C	float64	1	2
D	float64	2	1
E	object	0	2

사용자정의함수의 괄호안의 인자(argument, arg)는 일반인자(*arg)와 키워드인자(**kwarg)로 나누어진다. 일반인자(*arg)[2]는 인자의 개수를 사전에 알 수가 없는 경우 와일드(*)를 달고 인자(arg)를 사용한다❶~❺. '제6장4절 의사결정트리와 앙상블'에서 무작위 서치의 결과인 매개변수를 갱신할 때 사용하였다.

2) https://book.pythontips.com/en/latest/args_and_kwargs.html

```
def test_var_args(f_arg, *argv):                                    ❶
    print("first normal arg:", f_arg)                               ❷
    for arg in argv:                                                ❸
        print("another arg through *argv:", arg)                    ❹

test_var_args('sanggoo', 'python', 'eggs', 'apple')                 ❺
```

```
first normal arg: sanggoo
another arg through *argv: python
another arg through *argv: eggs
another arg through *argv: apple
```

키워드인자(**karg)는 인자의 개수를 사전에 알 수가 없으나 키워드는 알고 있는 경우에 사용한다❶~❺.

```
def greet_me(**kwargs):                                             ❶
    for key, value in kwargs.items():                               ❷
        print("{0} = {1}".format(key, value))                       ❸

greet_me(name="sanggoo", fruit ="apple")                            ❹
```

```
name = sanggoo
fruit = apple
```

사용자정의함수는 x, y, z를❶ 인자로 받아 합산하는 기능이다❷. 다만 'z' 는 키워드 인자로 '1'이 지정되었다. 'z'를 50으로 입력하여 함수를 실행하면 총 합산은 80이다❸.

```
def add(x,y,z=1) :                                                  ❶
    return (x + y + z)                                              ❷
add(10, 20, z=50)                                                   ❸
```

```
80
```

사용자정의함수는❶ 인자를 받아 일반인자(*arg)❷, 키워드인자(**kwarg)❸, add함수를 출력하고 일반인자(*arg)와 키워드인자(**kwarg)를 모두 사용자정의함수 add()의 입력인자로 잔달하는 기능이다❺.

```
def dispaly(add, *arg, **kwargs) :                                  ❶
    print(arg)                                                      ❷
    print(kwargs)                                                   ❸
    print(add)                                                      ❹
    return add(*arg, **kwargs)                                      ❺
```

사용자정의함수 display를❶ 실행한 결과이다.

```
dispaly(add, 10, 20, z=50)                                    ❶
(10, 20)
{'z': 50}
<function add at 0x0000028B32719E58>
80
```

자료 결측치 값에 '0'을 채우고❶ 확인해보면 데이터프레임에 바로 적용이 되지 않았지만 명령어를❷ 입력하면 데이터프레임에 직접 적용된 것을❸ 확인할 수 있다. 데이터프레임에 직접 반영하려면 'inplace=True' 명령어를 입력하면 된다.

```
df.fillna(0)                                                  ❶
df.fillna(0, inplace = True)  # df = df.fillna(0)             ❷
df                                                            ❸
```

	A	B	C	D	E
1	0.0	71.0	72.0	80.0	owl
2	72.0	0.0	71.0	0.0	fox
3	68.0	0.0	0.0	0.0	owl

	A	B	C	D	E
1	0.0	71.0	72.0	80.0	owl
2	72.0	0.0	71.0	0.0	fox
3	68.0	0.0	0.0	0.0	owl

데이터프레임의 특정 조건에 해당되는 행과 열을 Bool 값으로 찾아내어 삭제할 수 있다. 데이터프레임을 만든 후에❶❷❸ 행 'c', 행 'a'와 'c'를 삭제하여❹❺ 데이터를 확인한다.

```
data = sample_matrix.reshape((3,4))                                    ❶
df = pd.DataFrame(data, index = [i for i in 'abc'], columns = ['A','B','C','D'])   ❷
df                                                                     ❸
df.drop('c')                                                           ❹
df.drop(['a', 'c'])                                                    ❺
```

	A	B	C	D
a	52	71	72	80
b	72	53	71	46
c	68	59	55	19

	A	B	C	D
a	52	71	72	80
b	72	53	71	46

	A	B	C	D
b	72	53	71	46

열 ‘C’와 열 ‘A’와 ‘C’를 데이터프레임에서 삭제하려면❶❷ 데이터프레임의 열을 특정해주는 ‘axis=1’을 입력하면 된다.

```
df.drop('C', axis=1)                    ❶
df.drop(['A', 'C'], axis=1)             ❷
```

	A	B	D
a	52	71	80
b	72	53	46
c	68	59	19

	B	D
a	71	80
b	53	46
c	59	19

두 번째 행과 같은 ‘d’인덱스를❶ 만들어 자료를 확인하고❷ 중복이 있는지 확인한 후❸ 중복이 없는 인덱스의 자료만 추출하였다❹.

```
df.loc['d'] = df.loc['b']               ❶
df                                      ❷
df.duplicated()                         ❸
df[~df.duplicated()]                    ❹
```

	A	B	C	D
a	52	71	72	80
b	72	53	71	46
c	68	59	55	19
d	72	53	71	46

```
a    False
b    False
c    False
d     True
dtype: bool
```

	A	B	C	D
a	52	71	72	80
b	72	53	71	46
c	68	59	55	19

2.3 텍스트 처리

방탄소년단의 노래 가사 일부를 3개의 텍스트 변수로 읽어❶❷❸ 자료형태를 확인하여 문자(str)라는 것을 확인하였다.

```
txt = '      필 땐 장미꽃처럼, 흩날릴 땐 벚꽃처럼, 질 땐 나팔꽃처럼'  ❶
txt2 = '아름다운 그 순간처럼, 어짜피 이것 밖에 난 못해'  ❷
txt3 = '너를 사랑하는 것 밖엔 못해'  ❸
type(txt)  ❹
```

```
str
```

문자열의 공백을 기준으로 분할 구분하여 리스트로 변환하고❶❷ 리스트 형태의 문자열을 공백을 기준으로 구분하여 다시 문자열로 변환한다❸.

```
txt.split()  ❶
type(txt.split())  ❷
' '.join(txt.split(' '))  ❸
```

```
['필', '땐', '장미꽃처럼,', '흩날릴', '땐', '벚꽃처럼,', '질', '땐', '나팔꽃처럼']
list
'      필 땐 장미꽃처럼, 흩날릴 땐 벚꽃처럼, 질 땐 나팔꽃처럼'
```

문자열의 양 끝에 존재하는 공백과 '\n'을 제거하였습니다❶. 중간에 존재하는 것은 제거해주지 않는다.

```
txt.strip()  ❶
```

```
'필 땐 장미꽃처럼, 흩날릴 땐 벚꽃처럼, 질 땐 나팔꽃처럼'
```

문자열의 처음부터 9번째 글자를 보여주고❶ '장미'라는 단어가 처음 나나타는 문자열의 순서를❷ 찾아내었다.

```
txt[0:10]  ❶
txt.find('장미')  ❷
```

```
'      필 땐 장미'
8
```

문자열에 있는 '장미'를 '해바라기'로 대체하였다❶.

```
txt.replace('장미', '해바라기')                                                    ❶
'      필 땐 해바라기꽃처럼, 흩날릴 땐 벚꽃처럼, 질 땐 나팔꽃처럼'
```

3개의 텍스트 변수를 판다스 시리즈에서❶ 데이터프레임으로❷ 변환하여 확인하였다❸.

```
doc = pd.Series([txt, txt2, txt3], dtype="string")                                 ❶
doc = doc.to_frame()                                                               ❷
doc                                                                                ❸
```

	0
0	필 땐 장미꽃처럼, 흩날릴 땐 벚꽃처럼, 질 땐 나팔꽃처럼
1	아름다운 그 순간처럼, 어짜피 이것 밖에 난 뭇해
2	너를 사랑하는 것 밖엔 뭇해

데이터프레임에 '장미'라는 단어가 포함되어 있는지 확인해보면❶ 첫 번째 행에만 있고 나머지에는 없다.

```
doc[0].str.contains('장미')                                                        ❶
0         True
1        False
2        False
장미         NaN
dtype: object
```

반복문을 통해 데이터프레임에 '장미', '벚꽃', '사랑'이라는 단어를 하나씩 불러❶ 각 행에 해당 단어가 있으면 '1', 없으면 '0'을 표시하여 데이터프레임을 만들었다❷. 특정 단어의 출현 여부를 수치화하였다.

```
for choice in ['장미', '벚꽃', '사랑'] :                                           ❶
    doc[choice] = doc[0].str.contains(choice).fillna(0).astype(int)                ❷
```

	0	장미	벚꽃	사랑
0	필 땐 장미꽃처럼, 흩날릴 땐 벚꽃처럼, 질 땐 나팔꽃처럼	1	1	0
1	아름다운 그 순간처럼, 어짜피 이것 밖에 난 뭇해	0	0	0
2	너를 사랑하는 것 밖엔 뭇해	0	0	1

파이썬의 os 모듈은❶ ‘Operating System’의 약자로서 운영체제에서 제공되는 여러 기능을 수행할 수 있다. 현재 폴더를 확인하고❷ 하위 폴더(‘../’)와 현재 폴더(‘./’)의 파일과 폴더를 모두 보여준다❸. 필자는 ‘Book’ 폴더의 하위 폴더인 ‘script’에 앞에서 2장에서 실습한 파이썬 코드를 확인할 수 있다. 파이썬 노트북에서 자동으로 저장한 코드를 ‘.ipynb_checkpoints’폴더를 자동생성하여 보관하고 있다.

```
import os                                                    ❶
%pwd                                                         ❷
os.listdir('../') # 파일과 폴더 모두 보여줌                  ❸
os.listdir('./')                                             ❹
```

```
'D:₩₩ml₩₩Book₩₩script'
['script']
['.ipynb_checkpoints',
 '2장1절_연산과자료형태.ipynb',
 '2장2절_Numpy_pandas.ipynb',
 '2장3절_텍스트처리.ipynb',
 '2장_파이썬프로그래밍.ipynb']
```