목차

1. 데이터의 분류
2. 데이터 분석 절차
3. 판다스 개요

01 데이터의 분류

- **데이터 분석의 시작**
  - 분석 대상이 되는 데이터(자료)의 종류를 확인해야 한다.
  - 데이터의 종류에 따라 적용할 수 있는 분석 방법이 달라진다.

■ 값, 데이터, 정보

- **값(value)**
  - 데이터 분석에서 값은 하나의 숫자 또는 문자열을 의미한다.
  - 파이썬에서 값은 자료형(data type)을 갖는다.

01 데이터의 분류

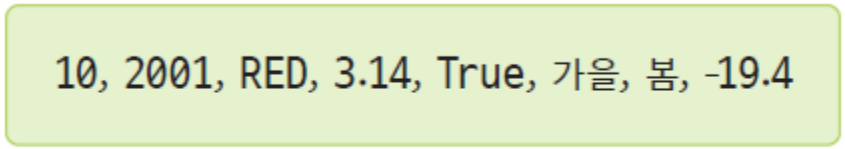


그림 2-1 값의 예

표 2-1 파이썬의 대표적인 자료형과 값의 예

자료형	값의 예
int	-5, 0, 10, 15
float	-4.5, 1.2, 3.14
bool	True, False
str	'RED', 'John', 'summer'

01 데이터의 분류

- **데이터(data)**
  - 동일한 성격을 갖는 값들을 체계적으로 모아놓은 것을 의미한다.
  - ex) 100명의 학생의 토익(TOEIC) 성적(값).
  - 데이터는 보통 1차원 배열 또는 2차원 배열 형태의 자료구조(data structure)에 저장되며, 분석에 활용된다.
  - 판다스(pandas): 자료구조(Series, DataFrame)

01 데이터의 분류

선호 계절
여름
봄
가을
겨울
봄
가을

(a) 1차원데이터

이름	성별	성적
김철수	M	95
최서영	F	84
최예림	F	76
하재경	M	90
정다훈	M	85
민유진	F	90

(b) 2차원 데이터

그림 2-2 데이터의 예

표 2-2 파이썬 환경의 자료구조

배열의 차원	파이썬	넘파이	판다스
1차원 데이터	list, tuple, dictionary	ndarray(1차원)	Series
2차원 데이터	list, tuple	ndarray(2차원)	DataFrame

01 데이터의 분류

▪ 정보(information)

- 데이터는 처리, 관찰, 분석을 거쳐 의미 있는 정보로 변환된다.
- 토익 성적 데이터를 분석하면 최저 점수, 최고 점수, 평균 점수, 800점 이상인 학생 수 등의 정보를 얻을 수 있다.
- 토익 성적 데이터에 성별 정보를 추가하면 남녀 간 TOEIC 평균 성적 차이와 같은 그룹 간 차이를 분석할 수도 있다.
- 데이터 분석의 목적은 데이터로부터 유용한 정보를 추출하는 것이다.

"학생들의 평균 몸무게는 61.5kg이다."
"남학생은 여학생보다 평균 12.4kg 더 많이 나간다."
"비만인 학생은 전체의 13%다."
"2%의 학생은 저체중이 의심된다."

그림 2-3 학생 200명의 몸무게 데이터를 분석하여 얻은 정보의 예

01 데이터의 분류

■ 데이터의 성격에 따른 분류

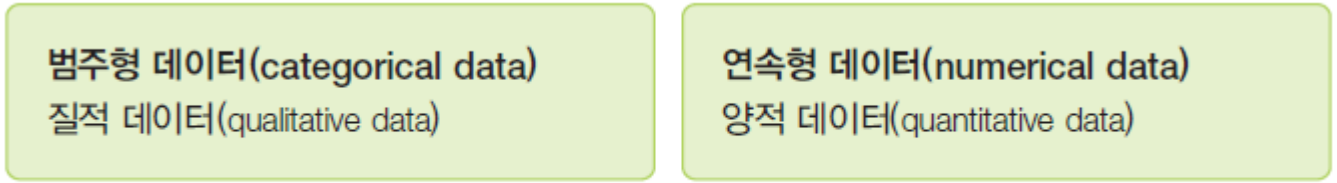


그림 2-4 데이터의 성격에 따른 분류

01 데이터의 분류

▪ 범주형 데이터

- 질적 데이터(qualitative data)라고도 불리며, 성별과 같이 범주 또는 그룹을 구분하는 값으로 구성된 데이터이다.
- 범주형 데이터의 값들은 기본적으로 숫자로 표현할 수 없으며, 대소 비교나 산술 연산이 적용되지 않는다.
- 예를 들어, 성별을 나타낼 때 남성은 0, 여성은 1로 표현할 수 있지만, 0과 1이 숫자로 표현되었다고 해서 산술 연산을 적용할 수 있는 것은 아니다.

표 2-3 범주형 데이터의 예

범주형 데이터	예
성별	M, F, F, M, M, M, F
혈액형	A, B, O, AB, B, A, O
선호하는 색깔	빨강, 파랑, 노랑, 빨강, 초록
찬성 여부	YES, NO, NO, YES, NO

표 2-4 숫자로 표현한 범주형 데이터의 예

범주형 데이터	예
성별	0, 1
혈액형	1, 2, 3, 4

01 데이터의 분류

■ 연속형 데이터

- 양적 데이터(quantitative data)라고도 불리며, 크기를 가진 숫자로 구성된 데이터이다.
- 연속형 데이터의 값들은 대소 비교가 가능하며, 평균, 최대값, 최소값 등의 산술 연산이 가능하므로 다양한 분석 방법이 적용될 수 있다.

표 2-5 연속형 데이터의 예

연속형 데이터	예
몸무게	57.4, 64.1, 71.0, 65.1, 90.1
키	169, 180, 174, 171, 181, 184
일평균 기온	19.1, 20.5, 20.5, 21.1, 22.0
자녀 수	0, 2, 1, 3, 0, 1, 2

01 데이터의 분류

■ 변수의 개수에 따른 분류

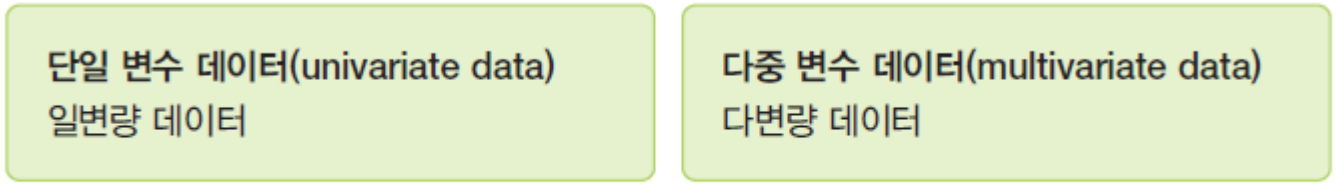


그림 2-5 변수의 개수에 따른 분류

■ 변수(variable)

- 데이터 분석에서의 변수는 프로그래밍 언어에서 말하는 변수와 의미상 차이가 있다.
- 데이터 분석에서 변수는 통계학 용어로, '연구, 조사, 관찰하고 싶은 대상의 특성'을 의미한다.
- ex) 키, 몸무게, 혈액형, 매출액, 습도, 미세먼지 농도 등

01 데이터의 분류

- **단일 변수 데이터(univariate data)**
  - 하나의 변수로만 구성된 데이터를 의미하며, 일변량 데이터라고도 한다.
- **다중 변수 데이터(multivariate data)**
  - 두 개 이상의 변수로 구성된 데이터를 의미하며, 다변량 데이터라고도 한다.
  - 특히, 두 개의 변수로 구성된 데이터는 이변량 데이터(bivariate data)라고 한다.

몸무게
62.4
65.3
59.8
46.5
49.8
58.7

(a) 단일 변수 데이터

키	몸무게	성별
168.4	62.4	M
169.5	63.2	F
172.1	70.1	F
185.2	78.5	M
173.7	68.2	M
175.2	72.0	F

(b) 다중 변수 데이터

그림 2-6 단일 변수 데이터와 다중 변수 데이터의 예

01 데이터의 분류

▪ 데이터의 특성 및 변수의 개수에 따른 분류

- 데이터의 특성(범주형/연속형)과 변수의 개수(단일/다중 변수)를 조합하면 총 3가지 경우의 수가 존재한다.

– 각 경우의 수에 따라 서로 다른 분석 방법이 적용된다.

– 다중 변수 데이터는 범주형 데이터와 연속형 데이터를 함께 포함할 수 있으므로 하나의 경우로 간주한다.

표 2-6 변수의 개수와 데이터의 특성에 따른 분류

데이터의 종류	학습할 장
단일 변수 범주형 데이터	5장 1절
단일 변수 연속형 데이터	5장 2절
다중 변수 데이터	6장

02 데이터 분석 절차

▪ 데이터 분석의 일반적 과정

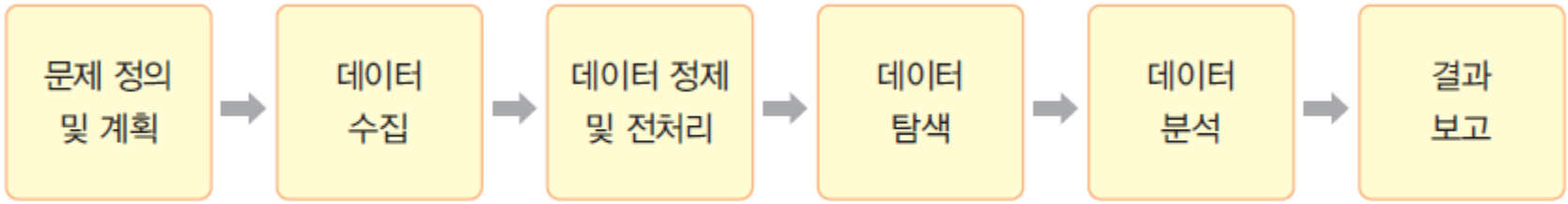


그림 2-7 데이터 분석 과정

02 데이터 분석 절차

■ 1단계: 문제 정의 및 계획(Problem Definition and Planning)

- 데이터 분석은 문제를 명확히 정의하는 것에서 시작한다.

  – 문제의 예

    » 미세먼지 농도는 어떻게 변화하는가?

    » 배추를 어느 규모로 경작하는 것이 최적일까?

    » 외국인 관광객들은 어떤 형태의 여행을 선호하는가?

    » 아파트 관리비는 적정한가?

- 문제가 명확해야 해당 문제를 해결하기 위한 데이터를 추정할 수 있으며, 적절한 분석 기법도 계획할 수 있다.
- 문제 정의가 구체적이고 명확할수록 데이터 분석이 방향을 잃지 않고 효과적으로 진행될 수 있다.

02 데이터 분석 절차

■ 2단계: 데이터 수집(Data Acquisition)

- 문제가 명확히 정의되면, 문제를 해결하기 위해 필요한 데이터가 무엇인지 파악하고, 이러한 데이터를 수집하는 과정을 거친다.
- 문제 해결에 필요한 데이터는 기존 시스템의 데이터베이스에 저장되어 있을 수도 있고, 엑셀 파일 형태로 관리되고 있을 수도 있다.

02 데이터 분석 절차

■ 3단계: 데이터 정제/전처리(Data Cleaning/Preprocessing)

- 여러 경로를 통해 수집된 데이터는 바로 분석에 사용할 수 없는 경우가 대부분이다.

- 예1) 한국과 미국 초등생의 발육 상태를 비교하기 위해 신체 검사 데이터를 수집한 경우, 한국의 데이터는 cm(센티미터), kg(킬로그램) 단위를 사용하지만, 미국의 데이터는 inch(인치), lb(파운드) 단위를 사용한다.

  – 직접 비교가 불가능하므로 단위를 통일해야 한다.

- 예2) 설문조사 데이터의 경우 무응답 항목(결측값)이 포함될 수 있다.

  – 비어 있는 데이터를 어떻게 처리할지 결정해야 한다.

02 데이터 분석 절차

- **데이터 정제 과정에서 고려해야 할 사항:**
  - 정상 범위를 벗어난 이상치(outlier)나 오류 데이터가 포함될 수 있으며, 이를 적절히 처리하지 않으면 통계 결과가 왜곡될 수 있다.
  - 연도별로 분산된 데이터를 하나로 병합해야 할 수도 있다.
  - 가로 방향으로 나열된 데이터를 분석이 용이하도록 세로 방향으로 변환할 수도 있다.
  - 수집된 데이터를 분석이 가능한 형태로 정돈하는 과정을 데이터 정제(Data Cleaning) 또는 전처리(Preprocessing)라고 하며, 데이터 분석에서 매우 중요한 단계 중 하나이다.

02 데이터 분석 절차

■ 4단계: 데이터 탐색(Data Exploration)

- 이 단계부터가 사실상 데이터 분석의 시작에 해당한다.
- 일반적으로 데이터 탐색(Exploration)과 데이터 분석(Analysis)을 구분한다.
- **데이터 탐색은 가벼운 데이터 분석**으로, 본격적인 분석에 앞서 정돈된 데이터 자체를 이해하고 파악하는 과정이다.
- 비교적 간단한 통계 기법을 적용하여 전체적인 데이터를 파악하는 단계이다.

– ex) 수집된 데이터의 건수는 얼마나 되는가? 남녀 데이터의 비율은 어떻게 되는가? 각 수집 항목별 평균값은 얼마인가?

- '탐색적 데이터 분석(EDA; Exploratory Data Analysis)'이라고도 한다.
- 가벼운 통계 분석이라고는 하지만 이 단계는 매우 중요하다.

– 탐색 단계에서 파악된 정보는 구체적인 분석 전략을 설계하는 데 활용된다.

– 예를 들어, 수도권과 비수도권 아파트 단지의 평균 관리비에 차이가 있다면, 분석 단계에서 그 원인을 규명해야 한다.

02 데이터 분석 절차

■ 5단계: 데이터 분석(Data Analysis)

- 데이터 탐색 단계에서 파악한 정보를 바탕으로 보다 심화된 분석을 수행하는 단계이다.
- 기본적인 통계 분석뿐만 아니라 다양한 고급 분석 기법이 적용된다.

– 군집 분석(Clustering Analysis), 분류 분석(Classification Analysis), 주성분 분석(Principal Component Analysis, PCA), 시계열 분석(Time-Series Analysis) 등

- 최근 주목받는 머신러닝(Machine Learning) 기술도 이 단계에서 활용된다.
- 데이터 분석은 단순한 분석을 넘어, 분석 결과에 대한 해석(Interpretation) 과정도 포함한다.

– 정보는 해석되어야 비로소 활용 방법을 결정할 수 있기 때문이다.

– 정보에 대한 해석 능력은 분석 대상이 되는 업무 분야에 대한 지식이 필요하므로, 데이터 분석가는 해당 업무 전문가와 협력하는 경우가 많다.

02 데이터 분석 절차

■ 6단계: 결과 보고(Reporting)

- 데이터 분석과 해석이 마무리되면, 그 내용을 정리하여 보고해야 한다.
- 정리의 핵심은 1단계에서 정의한 문제 해결에 도움이 되는 내용을 요약하는 것이다.
- 결과 보고 작성 단계에서 중요한 기술이 데이터 시각화(Data Visualization)이다.
- 데이터 시각화란 분석된 결과를 단순히 숫자로 나열하는 것이 아니라, 다양한 그래프나 그림을 활용하여 쉽게 이해할 수 있도록 표현하는 것을 의미한다.

그림 2-8 데이터 시각화의 예 © Slingshot

02 데이터 분석 절차

[여기서 잠깐!] 데이터 분석에 들어가는 시간

- 전체 분석 과정에서 약 80%의 시간이 분석을 위한 데이터 준비에 사용된다.
- 데이터 준비 기간을 얼마나 단축하느냐가 전체 분석 시간 단축의 핵심 요소가 된다.

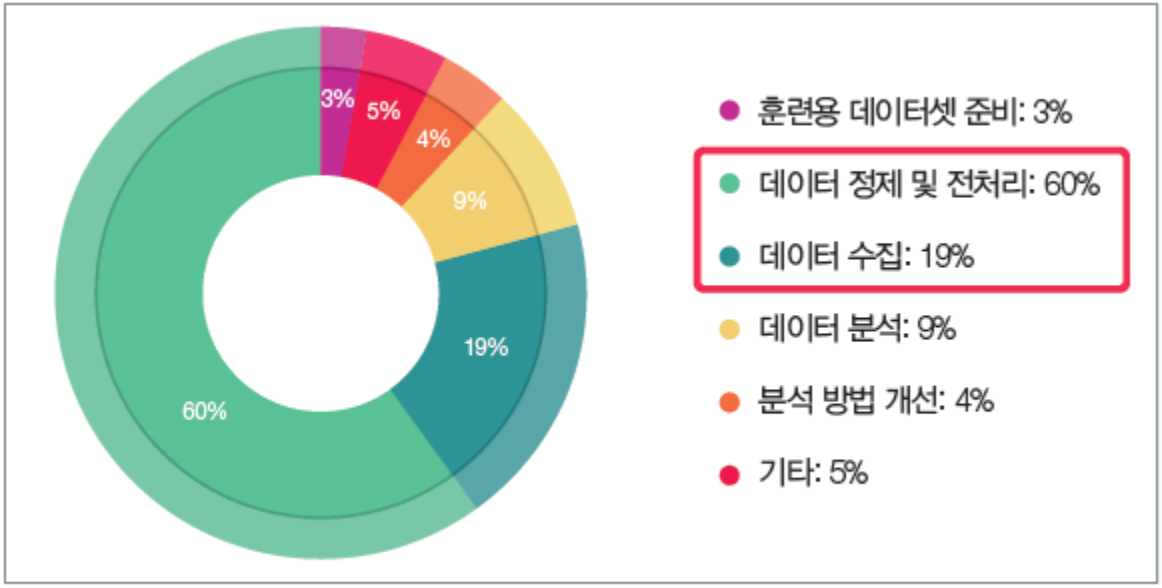


그림 2-9 데이터 분석 작업에 들어가는 시간 © Forbes

03 판다스 개요

■ 판다스 라이브러리

- 판다스(Pandas)는 파이썬에서 데이터 조작 작업을 단순하고 효율적으로 수행할 수 있도록 지원하는 강력한 라이브러리(패키지)이다.
- 소프트웨어 개발자이자 사업가인 웨스 맥키니(Wes McKinney)가 2008년에 개발하였다.
- "Pandas"는 "Panel Data"와 "Python Data Analysis"의 합성어이다.
- 판다스는 넘파이(NumPy) 라이브러리를 기반으로 구축되었으며, 1차원 배열 및 표 형태의 2차원 배열 데이터를 다룰 수 있도록 지원한다.

03 판다스 개요

▪ 판다스의 주요 기능

- 데이터셋에 포함된 결측값을 쉽게 처리할 수 있다.
- 데이터셋에서 행이나 열을 추가하거나 삭제하는 작업을 간편하게 수행할 수 있다.
- 데이터 객체에 레이블을 부여하고, 레이블을 활용하여 데이터를 다룰 수 있다.
- 데이터를 그룹화(Grouping)하거나, 그룹별로 집계(Aggregation)하는 작업이 가능하다.
- 엑셀 파일, 넘파이(NumPy) 배열 등을 손쉽게 판다스 객체로 변환할 수 있다.
- 데이터 레이블을 기반으로 데이터 자르기(Slicing), 인덱싱(Indexing), 부분집합(Subsetting) 구하기가 가능하다.
- 데이터의 형태를 쉽게 변환할 수 있다.
- 데이터 축(Axis)에 대한 계층적 레이블링(Hierarchical Labeling)을 지원한다.
- 판다스 객체를 CSV 파일, 엑셀 파일, 데이터베이스 형식으로 저장할 수 있다.
- 시계열(Time Series) 데이터를 다루는 데 유용한 기능을 제공한다.

03 판다스 개요

■ 시리즈(Series)와 데이터프레임(DataFrame) 맛보기

- **시리즈(Series):** 1차원 배열로서 단일 변수 데이터를 저장하는 데 사용된다.
- **데이터프레임(DataFrame):** 2차원 배열로서 다중 변수 데이터를 저장하는 데 사용된다.
  - 데이터프레임은 여러 개의 1차원 배열(시리즈)을 세로 방향으로 묶어 놓은 형태를 가진다.

표 2-7 데이터의 종류와 판다스 자료구조

데이터의 종류	배열의 차원	판다스 자료구조	비고
단일 변수 데이터	1차원	시리즈	–
다중 변수 데이터	2차원	데이터프레임	시리즈의 묶음

03 판다스 개요

```
>>> age
0       25
1       34
2       19
3       45
4       60
dtype: int64
```

(a) 시리즈 출력 결과의 예

```
>>> score
       KOR  ENG  MATH  SCI
John    85   96    40   95
Tom     73   69    45   80
Jane    78   50    60   90
Grace   95   60    80   95
```

(b) 데이터프레임 출력 결과의 예

그림 2-10 시리즈와 데이터프레임의 출력 결과 예

03 판다스 개요

▪ 판다스 시리즈 객체 생성하기

[코드 2-1]

```
import pandas as pd
# 직접 리스트 입력
age = pd.Series([25, 34, 19, 45, 60])
age                          # age의 내용 출력
type(age)                    # age의 데이터형 확인
# 이미 생성된 리스트 객체를 입력
data = ['spring', 'summer', 'fall', 'winter']
season = pd.Series(data)
season
season.iloc[2]               # 인덱스 2의 값
```

03 판다스 개요

```
>>> import pandas as pd
```

- 작업에 필요한 판다스 라이브러리를 불러온다.
  - 일반적으로 판다스는 pd라는 이름으로 사용된다.

03 판다스 개요

- 1차원 배열 리스트는 가로 방향, 시리즈는 세로 방향으로 값이 출력된다.

```
>>> # 직접 리스트 입력
>>>
>>> age = pd.Series([25, 34, 19, 45, 60])

>>> age                          # age의 내용 출력
0    25
1    34
2    19
3    45
4    60
dtype: int64
>>> type(age)                    # age의 자료형 확인
<class 'pandas.core.series.Series'>
```

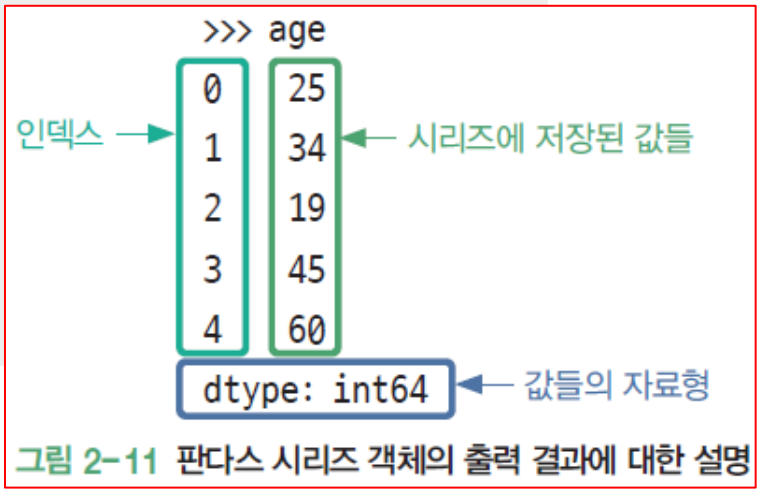


그림 2-11 판다스 시리즈 객체의 출력 결과에 대한 설명

03 판다스 개요

```
>>> # 이미 생성된 리스트 객체를 입력
>>>
>>> data = ['spring', 'summer', 'fall', 'winter']
>>> season = pd.Series(data)
>>> season
0    spring
1    summer
2      fall
3    winter
dtype: object
```

03 판다스 개요

```
>>> season.iloc[2]                    # 인덱스 2의 값
'fall'
```

- 판다스 시리즈는 리스트나 넘파이 배열과 동일하게 인덱스를 이용하여 원소를 다룰 수 있다.

– season.iloc[2]는 season 객체의 두 번째 값(0부터 시작)을 의미하므로 'fall'을 반환한다.

03 판다스 개요

- **판다스 데이터프레임 객체 생성하기**

```
import pandas as pd

# 2차원 리스트로 부터 데이터프레임 생성
score = pd.DataFrame([[85, 96, 40, 95],
                      [73, 69, 45, 80],
                      [78, 50, 60, 90]])
score                                   # score 내용 출력
type(score)                             # score 데이터형 출력

score.index                             # 행 방향 인덱스
score.columns                           # 열 방향 인덱스

score.iloc[1,2]                         # 인덱스 1행 2열의 값
```

03 판다스 개요

```
>>> # 2차원 리스트로부터 데이터프레임 생성
>>>
>>> score = pd.DataFrame([[85, 96, 40, 95],
...                       [73, 69, 45, 80],
...                       [78, 50, 60, 90]])
>>>
>>> score                                    # score 내용 출력
    0   1   2   3
0  85  96  40  95
1  73  69  45  80
2  78  50  60  90
>>> type(score)                              # score 자료형 출력
<class 'pandas.core.frame.DataFrame'>
```

- 행 방향과 열 방향으로 인덱스 번호가 붙어 있다는 점에 주목해야 한다.

03 판다스 개요

```
>>> score.index                                    # 행 방향 인덱스
RangeIndex(start=0, stop=3, step=1)
>>> score.columns                                  # 열 방향 인덱스
RangeIndex(start=0, stop=4, step=1)
```

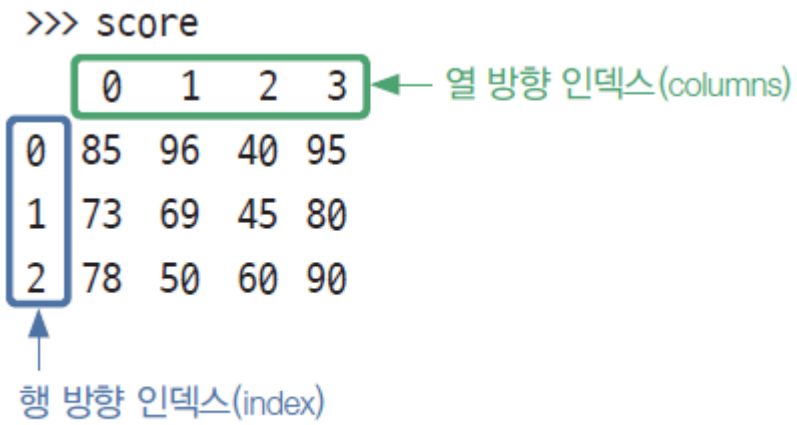


그림 2-12 데이터프레임 행과 열의 인덱스

03 판다스 개요

[참고] np.int64(45)

```
>>> score.iloc[1,2]
np.int64(45)
>>> print(score.iloc[1,2])
45
>>> score.iloc[1,2]+10
np.int64(45)
```

- np.int64(45): 값 45의 자료형이 정수형(int) 숫자임을 표시.
- 파이썬 버전 업그레이드 이후, 출력되는 단일 값에 대해 자료형이 표시됨.
- print() 문으로 출력하면 자료형 없이 값만 출력됨.
- 산술 연산에는 영향을 미치지 않음.
- np.float64(3.14): 3.14가 실수형(float) 숫자임을 의미.

03 판다스 개요

■ [하나 더 알기] 넘파이 배열 ↔ 판다스 배열

▪ 넘파이 1차원 배열 ↔ 판다스 시리즈

```
import pandas as pd
import numpy as np

# 넘파이 1차원 배열을 판다스로 변환
w_np = np.array([65.4, 71.3, np.nan, 57.8])   # 넘파이 1차원 배열
weight = pd.Series(w_np)                      # 넘파이 배열을 판다스 시리즈로
weight                                        # 판다스 시리즈

# 판다스 시리즈를 넘파이로 변환
w_np2 = pd.Index.to_numpy(weight)             # 판다스 시리즈를 넘파이 배열로
w_np2                                         # 넘파이 1차원 배열
```

03 판다스 개요

- **넘파이 2차원 배열 ↔ 판다스 데이터프레임**

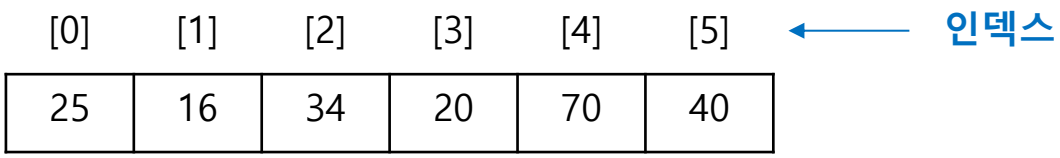
```
import pandas as pd
import numpy as np

# 넘파이 2차원 배열로 부터 데이터프레임 생성
s_np = np.array([[85, 96, 40, 95],
                 [73, 69, 45, 80],
                 [78, 50, 60, 90]])
s_np                                    # 넘파이 2차원 배열
score2 = pd.DataFrame(s_np)                 # 넘파이 배열을 데이터프레임으로
score2                                  # 판다스 데이터프레임
# 데이터프레임을 넘파이 2차원 배열로 변환
score_np = score2.to_numpy()                # 데이터프레임을 넘파이 배열로
score_np                                # 넘파이 2차원 배열
```

03 판다스 개요

■ 판다스 인덱싱 시스템

- 배열은 전통적인 자료 구조로서 데이터를 저장하는 데 사용된다.
- 배열에 저장된 원소(값)의 위치를 식별하기 위해 인덱스(index)를 사용한다.



- **인덱스 사용 방식:**

– 1차원 배열: 하나의 인덱스 값으로 원소의 위치를 식별

– 2차원 배열: 행과 열을 지정하는 두 개의 인덱스 값이 필요

– 인덱스는 1이 아닌 0부터 시작

- 판다스 시리즈와 데이터프레임의 인덱싱 시스템은 전통적인 배열의 인덱스와 다소 차이가 있으므로 이해가 필요하다.

03 판다스 개요

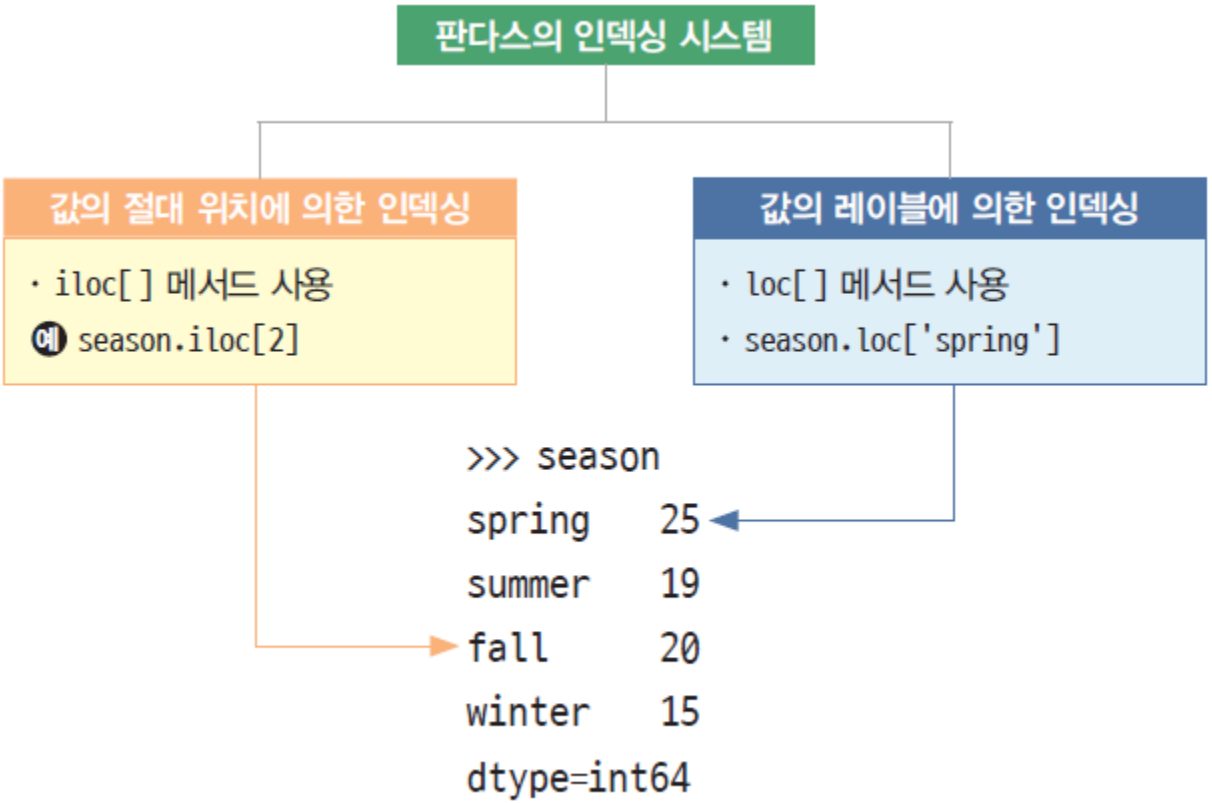


그림 2-13 판다스의 인덱싱 시스템

03 판다스 개요

▪ 절대 위치 인덱스

- 배열 안에 있는 원소의 절대 위치를 정수로 표현한다.
- 전통적인 인덱스 방식이며, 파이썬의 리스트나 넘파이 배열도 이 방식을 사용한다.
- 시리즈(Series)나 데이터프레임(DataFrame)을 출력해도 절대 위치 인덱스는 표시되지 않는다.
- 위치 지정 시 .iloc[] 사용

• ex) season.iloc[2]

절대위치 인덱스가 아닌 레이블 인덱스임

```
>>> season
0   25
1   19
2   20
3   15
dtype=int64
```

03 판다스 개요

▪ 레이블 인덱스

- 배열 안에 저장된 값에 특성을 나타내는 레이블(label)을 부여하고, 이를 인덱스처럼 사용한다.
- 시리즈(Series)나 데이터프레임(DataFrame)을 출력하면 레이블 인덱스가 표시된다.
- 위치 지정 시 .loc[] 사용
- ex) season.loc['spring']

레이블 인덱스

```
>>> season
spring    25
summer    19
fall      20
winter    15
dtype=int64
```

03 판다스 개요

▪ 판다스 객체의 index와 column 속성

- 판다스에서 index는 행(row)의 인덱스를 의미하며, 열(column)의 인덱스는 columns로 표현된다.
- 시리즈(Series)는 1차원 배열로 index만 존재한다.
- 데이터프레임(DataFrame)은 2차원 배열로, index와 columns의 조합을 통해 값의 위치를 식별한다.

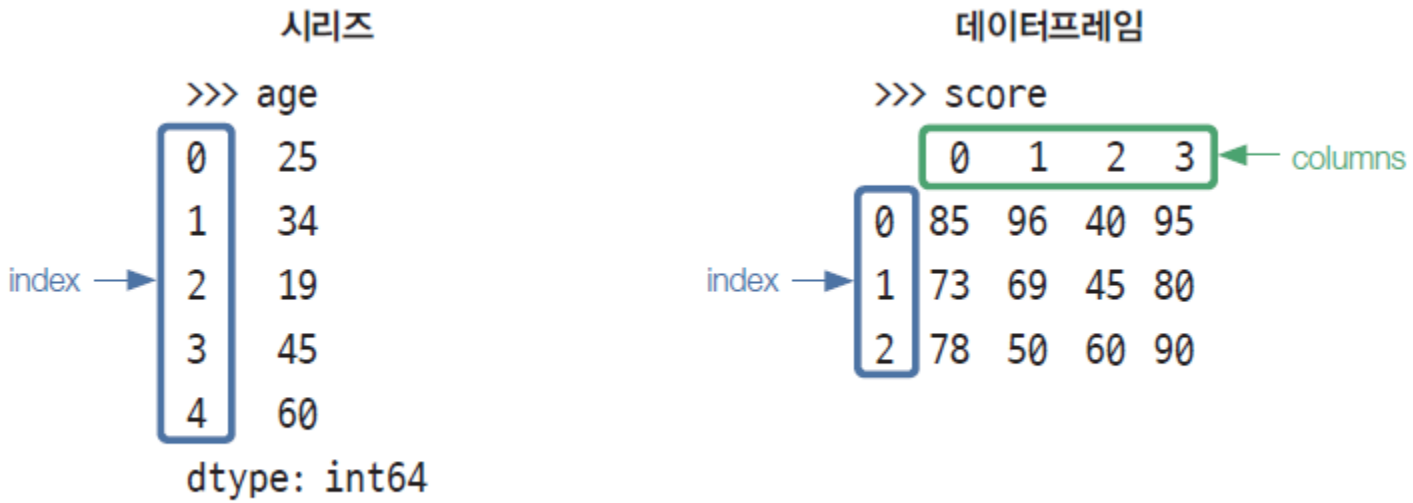


그림 2-14 판다스의 index와 columns

03 판다스 개요

- **행과 열에 레이블을 부여하는 방법**

[코드 2-3]

```
import pandas as pd

# 시리즈에 레이블 부여

age = pd.Series([25, 34, 19, 45])
age                                    # 레이블 부여 전
age.index = ['John','Jane','Tom','Luka']   # 행에 레이블 부여
age                                    # 레이블 부여 후

age.iloc[2]                            # 절대위치에 의한 인덱싱
age.loc['Tom']                           # 레이블에 의한 인덱싱
```

03 판다스 개요

```
>>> import pandas as pd
>>> # 시리즈에 레이블 부여
>>>
>>> age = pd.Series([25, 34, 19, 45])
>>> age                                    # 레이블 부여 전
0   25
1   34
2   19
3   45
dtype: int64
```

절대 위치 인덱스가 아니라 레이블 인덱스임. 만일 두번째 원소 삭제 후 age 를 출력하면

```
>>> age
0   25
2   19
3   45
dtype: int64
```

- 왼쪽에 0~3의 정수 인덱스가 표시되는 것은 레이블을 부여하지 않았기 때문이며, 시스템에서 자동으로 정수 인덱스를 레이블로 할당한 것이다.

03 판다스 개요

```
>>> age.index = ['John','Jane','Tom','Luka']        # 행에 레이블 부여
>>> age                                             # 레이블 부여 후
John    25
Jane    34
Tom     19
Luka    45
dtype: int64
```

레이블 인덱스

값에 레이블을 부여한 후에도 절대 위치 인덱스는 내부적으로 유지가 되어 사용가능하다.

```
>>> age.iloc[2]                                     # 절대위치에 의한 인덱싱
np.int64(19)
>>> age.loc['Tom']                                  # 레이블에 의한 인덱싱
np.int64(19)
```

03 판다스 개요

- [코드 2-3] (계속)

```
# 데이터프레임에 레이블 부여
score = pd.DataFrame([[85, 96, 40, 95],
                      [73, 69, 45, 80],
                      [78, 50, 60, 90]])
score                                    # 레이블 부여 전
score.index = ['John','Jane','Tom']          # 행에 레이블 부여
score.columns = ['KOR','ENG','MATH','SCI']  # 열에 레이블 부여
score                                    # 레이블 부여 후

score.iloc[2,1]                           # 절대위치에 의한 인덱싱
score.loc['Tom','ENG']                       # 레이블에 의한 인덱싱
```

03 판다스 개요

```
>>> # 데이터프레임에 레이블 부여
>>>
>>> score = pd.DataFrame([[85, 96, 40, 95],
... [73, 69, 45, 80],
... [78, 50, 60, 90]])
>>>
>>> score                                    # 레이블 부여 전
   0  1  2  3
0 85 96 40 95
1 73 69 45 80
2 78 50 60 90
```

- 아직 레이블 부여 전이므로 정수 인덱스가 행과 열의 레이블로 자동 부여되어 화면에 표시 (절대위치 인덱스 == 레이블 인덱스)

03 판다스 개요

```
>>> score.index = ['John','Jane','Tom']                    # 행에 레이블 부여
>>> score.columns = ['KOR','ENG','MATH','SCI']             # 열에 레이블 부여
>>> score                                                  # 레이블 부여 후
      KOR ENG MATH SCI
John   85  96   40  95
Jane   73  69   45  80
Tom    78  50   60  90
```

- 레이블을 부여 후에는 절대위치 인덱스 ≠ 레이블 인덱스

```
>>> score.iloc[2,1]                                        # 절대위치에 의한 인덱싱
np.int64(50)
>>> score.loc['Tom','ENG']                                 # 레이블에 의한 인덱싱
np.int64(50)
```

03 판다스 개요

- **절대위치 인덱스와 레이블 인덱스 정리**
  - 사용자 눈에는 레이블 인덱스만 보인다.

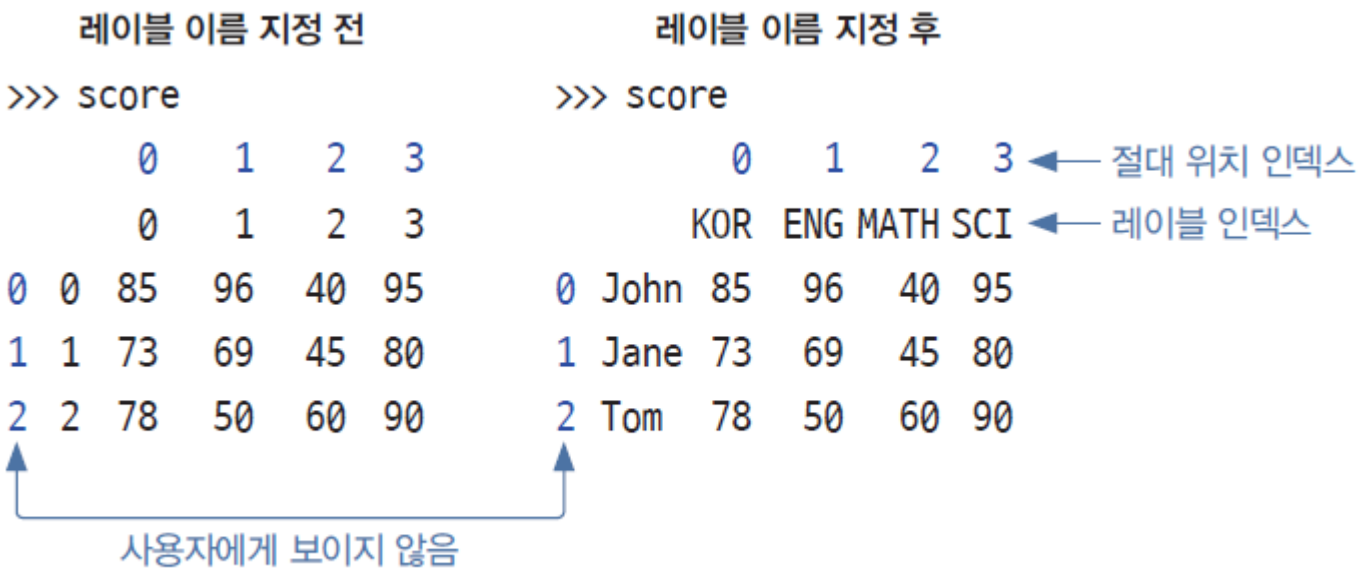


그림 2-15 절대 위치 인덱스와 레이블 인덱스

03 판다스 개요

- **중복된 레이블 부여**
  - 절대 위치 인덱스는 중복되지 않으며, 시스템에 의해 자동 관리된다.
  - 레이블 인덱스는 사용자가 임의로 지정할 수 있으며, 중복이 존재할 수 있다.

[코드 2-4]

```
import pandas as pd


age = pd.Series([25, 34, 19, 45, 60])
age.index = ['John','Jane','Tom','Micle','Tom']
age


age.iloc[3]
age.loc['Tom']
```

03 판다스 개요

```
>>> age
John      25
Jane      34
Tom       19
Micle     45
Tom       60
dtype: int64

>>> age.iloc[3]
np.int64(45)
>>> age.loc['Tom']
Tom 19
Tom 60
dtype: int64
```

03 판다스 개요

- **숫자 레이블의 부여**
  - 레이블은 일반적으로 문자열로 지정하지만, 연도와 같이 숫자로 지정하는 것도 가능하다.
  - 레이블이 숫자든 문자열이든, 레이블 값으로 인덱싱할 때는 반드시 .loc[] 메서드를 사용해야 한다.

[코드 2-5]

```
import pandas as pd


population = pd.Series([523, 675, 690, 720, 800])
population.index = [10, 20, 30, 40, 50]          # 숫자 레이블 지정
population
population.iloc[20]                              # 에러발생
population.loc[20]
```

03 판다스 개요

```
>>> population
10 523
20 675
30 690
40 720
50 800
dtype: int64
>>> population.iloc[20]                                    # 에러발생
IndexError: single positional indexer is out-of-bounds
>>> population.loc[20]
np.int64(675)
```

- 레이블이 숫자이든 문자열이든, 레이블 값으로 인덱싱할 때는 반드시 .loc[] 메서드를 사용해야 한다.

실습해보기

1. 다음의 월평균기온 통계표를 판다스 데이터프레임에 저장하시오.

	전북	전주	군산	부안
1월	-0.1	0.0	-0.1	-0.2
2월	1.8	2.0	1.6	1.6
3월	6.4	6.8	5.8	5.9
4월	12.3	12.9	11.5	11.5
5월	17.9	18.5	17.1	17.1
6월	22.2	22.8	21.6	21.5

회색 음영 부분은
행과 열의 인덱스로 저장

2. 절대위치 인덱스를 이용하여 전주의 3월 평균 기온을 출력하시오.
3. 절대위치 인덱스를 이용하여 부안의 4월 평균 기온을 출력하시오.
4. 레이블 인덱스를 이용하여 군산의 1월 평균 기온을 출력하시오.
5. 레이블 인덱스를 이용하여 전북의 6월 평균 기온을 출력하시오.