# 1. 시리즈 정보 확인하기

[코드 3-1]

```
import pandas as pd
import numpy as np

# 자료의 입력
temp = pd.Series([-0.8, -0.1, 7.7, 13.8, 18.0, 22.4,
                  25.9, 25.3, 21.0, 14.0, 9.6, -1.4])
temp                                          # temp의 내용 확인

# 월이름을 레이블 인덱스로 지정하기
temp.index                                     # 인덱스 내용 확인
temp.index = ['1월','2월','3월','4월',          # 월이름을 인덱스로 지정
              '5월','6월','7월','8월',
              '9월','10월','11월','12월']
temp                                          # temp의 내용 확인
```

# 1. 시리즈 정보 확인하기

- 코드  입력해서 확인해보기(Series 정보)

```
temp.size                    # 배열의 크기(값의 개수)
len(temp)                    # 배열의 크기(값의 개수)
temp.shape                   # 배열의 형태
temp.dtype                   # 배열의 자료형
```

# 1. 시리즈 정보 확인하기

- 인덱싱과 슬라이싱
  - **인덱싱(Indexing):** 시리즈에 저장된 값의 위치를 가리키는 체계를 의미한다.
  - **슬라이싱(Slicing):** 인덱스와 조건문을 이용하여 시리즈에서 일부 값을 잘라내어 추출하는 연산이다.

| temp | -0.8 | -0.1 | 7.7 | 13.8 | 18.0 | 22.4 | 25.9 | 25.3 | 21.0 | 14.0 | 9.6 | -1.4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 절대 위치 인덱스 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 레이블 인덱스 | 1월 | 2월 | 3월 | 4월 | 5월 | 6월 | 7월 | 8월 | 9월 | 10월 | 11월 | 12월 |

그림 3-1 temp 객체의 절대 위치 인덱스와 레이블 인덱스

# 1. 시리즈 정보 확인하기

- 인덱싱과 슬라이싱

```
temp.iloc[2]                          # 인덱스 2의 값
temp.loc['3월']                       # 인덱스 '3월'의 값
temp.iloc[[3,5,7]]                    # 인덱스 3,5,7의 값
temp.loc[['4월','6월','8월']]          # 인덱스 '4월','6월','8월'의 값

temp.iloc[5:8]                        # 인덱스 5~7의 값
temp.loc['6월':'9월']                 # 인덱스 '6월'~'9월'의 값
temp.iloc[:4]                         # 인덱스 0~3의 값
temp.iloc[9:]                         # 인덱스 9~11의 값
temp.iloc[:]                          # 인덱스 0~11의 값
```

# 1. 시리즈 정보 확인하기

- 인덱싱과 슬라이싱

| 인덱스 | start | end | 예 |
|---|---|---|---|
| 절대 위치 인덱스 | ○ | × | `iloc[5:8]` → 5, 6, 7 |
| 레이블 인덱스 | ○ | ○ | `loc['6월':'9월']` → '6월', '7월', '8월', '9월' |

  - 범위 지정 연산(start:end)에서 start나 end를 생략할 수 있다.
  - start 생략: end 이전의 모든 인덱스를 포함
  - end 생략: start부터 이후의 모든 인덱스를 포함
  - start와 end 모두 생략: 전체 인덱스를 포함

# 1. 시리즈 정보 확인하기

- 조건문을  이용한슬라이싱
  - 조건문을 사용하여 특정 조건에 맞는 값들을 시리즈 객체에서 추출할 수 있다.
  - 조건문을 이용한 슬라이싱 시 .loc[] 메서드를 사용해야 한다.

```
# 월평균 기온이 15도 이상인 월들의 기온을 보이시오
temp.loc[temp >= 15]
# 월평균 기온이 15도 이상 25도 미만인 월들의 기온을 보이시오
temp.loc[(temp >= 15) & (temp < 25)]
# 월평균 기온이 5도 미만이거나 25도 이상인 월들의 기온을 보이시오
temp.loc[(temp < 5) | (temp >= 25)]
```

# 1. 시리즈 정보 확인하기

- 값을 추출하기 위한 조건 지정 시 하나의 조건이 아닌 여러 개의 조건을 지정할 수 있다.
- 각 조건은 '&' (and)나 ':' (or)로 연결한다.
- 주의할 점은 각 조건은 반드시 ( ) 로 묶어 주어야 한다는 것
- ( )가 없는 다중 조건문 temp >= 15 & temp < 25 은 에러 발생

temp >= 15 & temp < 25

(temp >= 15) & (temp < 25)

# 1. 시리즈 정보 확인하기

```
# 3월보다 기온이 낮은 월들의 기온을 보이시오
march = temp.loc['3월']
temp.loc[temp < march]
# where() 를 이용한 조건 검색
temp.where(temp >= 15)
temp.where(temp >= 15).dropna()
```

- where() 메서드는 조건문을 이용하여 값을 검색하는 기능을 제공한다.
- 조건에 맞지 않는 값들은 결측값(NaN)으로 표시된다.
- 결측값을 제외하고 조건에 맞는 값들만 보려면 dropna() 메서드를 where() 실행 결과에 추가한다.

# 1. 시리즈 정보 확인하기

```
>>> temp.where(temp >= 15)
1월      NaN
2월      NaN
3월      NaN
4월      NaN
5월     18.0
6월     22.4
7월     25.9
8월     25.3
9월     21.0
10월     NaN
11월     NaN
12월     NaN
dtype: float64
```

# 2. 시리즈 객체의 산술 연산

- 각각 값에 개별적으로 적용된다.
- 두 시리즈 객체 간의 산술 연산은 인덱스가 같은 값들끼리 수행된다.

```
temp + 1

2 * temp + 0.1

temp + temp

temp.loc[temp >= 15] + 1      # 기온이 15도 넘는 월들에 대해서만 1 증가
```

# 2. 시리즈 객체의 산술 연산

```
>>> temp + 1
1월      0.2
2월      0.9
(생략)
11월    10.6
12월    -0.4
dtype: float64
>>> 2 * temp + 0.1
1월     -1.5
2월     -0.1
(생략)
11월    19.3
12월    -2.7
dtype: float64
```

| temp | | temp + 1 | | temp + temp | |
|---|---|---|---|---|---|
| 1월 | -0.8 | 1월 | 0.2 | 1월 | -1.6 |
| 2월 | -0.1 | 2월 | 0.9 | 2월 | -0.2 |
| 3월 | 7.7 | 3월 | 8.7 | 3월 | 15.4 |
| 4월 | 13.8 | 4월 | 14.8 | 4월 | 27.6 |
| 5월 | 18.0 | 5월 | 19.0 | 5월 | 36.0 |
| 6월 | 22.4 | 6월 | 23.4 | 6월 | 44.8 |
| 7월 | 25.9 | 7월 | 26.9 | 7월 | 51.8 |
| 8월 | 25.3 | 8월 | 26.3 | 8월 | 50.6 |
| 9월 | 21.0 | 9월 | 22.0 | 9월 | 42.0 |
| 10월 | 14.0 | 10월 | 15.0 | 10월 | 28.0 |
| 11월 | 9.6 | 11월 | 10.6 | 11월 | 19.2 |
| 12월 | -1.4 | 12월 | -0.4 | 12월 | -2.8 |

# 2. 시리즈 객체의 산술 연산

- 통계 관련 메서드

| 메서드명 | 설명 |
|---|---|
| sum() | 합계 |
| mean() | 평균 |
| median() | 중앙값 |
| max() | 최댓값 |
| min() | 최솟값 |
| std() | 표준편차 |
| var() | 분산 |

```
temp.sum()          # 값들의 합계
temp.mean()         # 값들의 평균
temp.median()       # 값들의 중앙값
temp.max()          # 값들의 최대값
temp.min()          # 값들의 최소값
temp.std()          # 값들의 표준편차
temp.var()          # 값들의 분산
temp.abs()          # 값들의 절대값
temp.describe()     # 기초통계정보
```

# 2. 시리즈 객체의 산술 연산

- describe() 에 포함된 통계값

| 통곗값 | 설명 |
|---|---|
| count | 값들의 개수 |
| mean | 값들의 평균 |
| std | 값들의 표준편차 |
| min | 값들의 최솟값 |
| 25% | 1사분위수 |
| 50% | 2사분위수(중앙값) |
| 75% | 3사분위수 |
| max | 값들의 최댓값 |

# 3. 시리즈 객체 내용 변경

- code2

```
salary = pd.Series([20, 15, 18, 30])        # 레이블 인덱스가 없는 시리즈 객체
score = pd.Series([75, 80, 90, 60],
                  index=['KOR', 'ENG', 'MATH', 'SOC'])
salary
score


# 값의 변경
score.iloc[0] = 85                          # 인덱스 0 의 값을 변경
score
score.loc['SOC'] = 65                       # 인덱스 'SOC' 의 값을 변경
score
score.loc[['ENG','MATH']] = [70,80]         # 인덱스 'ENG','MATH' 의 값을 변경
Score
```

# 3. 시리즈 객체 내용 변경

- 값의 추가는 loc[]를 통해서만 가능하다

```
Score.loc[ ' PHY ' ] = 50                    # 없는 인덱스 추가
Score
Score.iloc[5] = 90                          # 에러 발생
# 값의 추가 (레이블 인덱스가 없는 경우)
Next_idx = salary.size
Salary.iloc[next_idx] = 33                  # 에러 발생
Salary.loc[next_idx] = 33                   # 정상 수행
Salary
```

# 3. 시리즈 객체 내용 변경

- _append() 메서드를 이용한 추가

```
New = pd.Series({ ' MUS ' : 95})

Score._append(new)                          # score 변경 없음

Score

Score = score._append(new)                  # score 변경됨

Score



Salary._append(pd.Series([66]), ignore_index=True)

Salary = salary._append(pd.Series([66]), ignore_index=True)

Salary
```

# 3. 시리즈 객체 내용 변경

- 레이블을 지정하지 않은 시리즈 객체에 새로운 값 추가
  - ignore_index = True ➔ 추가되는 객체에 맞춰 새로운 레이블 인덱스 부여

```
>>> salary._append(pd.Series([66]), ignore_index=True)
0    20
1    15
2    18
3    30
4    33
5    66
dtype: int64
>>> salary = salary._append(pd.Series([66]), ignore_index=True)
>>> salary
0    20
1    15
2    18
3    30
4    33
5    66
dtype: int64
```

프리젠테이션을 마지막으로 수정한 날짜: 2월 12일

레이블을 지정하지 않은
시리즈 객체의 경우
새로운 값을 추가하는 예

# 3. 시리즈 객체 내용 변경

- 시리즈 객체 삭제 : drop() 메서드 사용,레이블 인덱스를 매개변수로 입력
- 실제 시리즈를 삭제 하지 않고, 결과를 미리 보여준다
- 실제 삭제하기 위해서는 inplace 매개 변수 값을 True 로 설정.

**[코드 3-10]**

```
# 코드3-8에 이어서 실행

# 값의 삭제
salary = pd.Series([20, 15, 18, 30])  # 레이블 인덱스가 없는 시리즈 객체
score = pd.Series([75, 80, 90, 60],
                  index=['KOR', 'ENG', 'MATH', 'SOC'])
score.drop('PHY')                        # 레이블 인덱스가 있는 경우
score                                   # score의 내용 변동 없음
score = score.drop('PHY')
score                                   # score의 내용 변경

salary = salary.drop(1)                   # 레이블 인덱스가 없는 경우
Salary
```

# 3. 시리즈 객체 내용 변경

```
>>> salary = salary.drop(1)                    # 레이블 인덱스가 숫자인 경우
>>> salary
0    20
2    18
3    30
4    33
5    66
dtype: int64
```

```
>>> score = score.drop('PHY')
```

=

```
>>> score.drop('PHY', inplace=True)
```

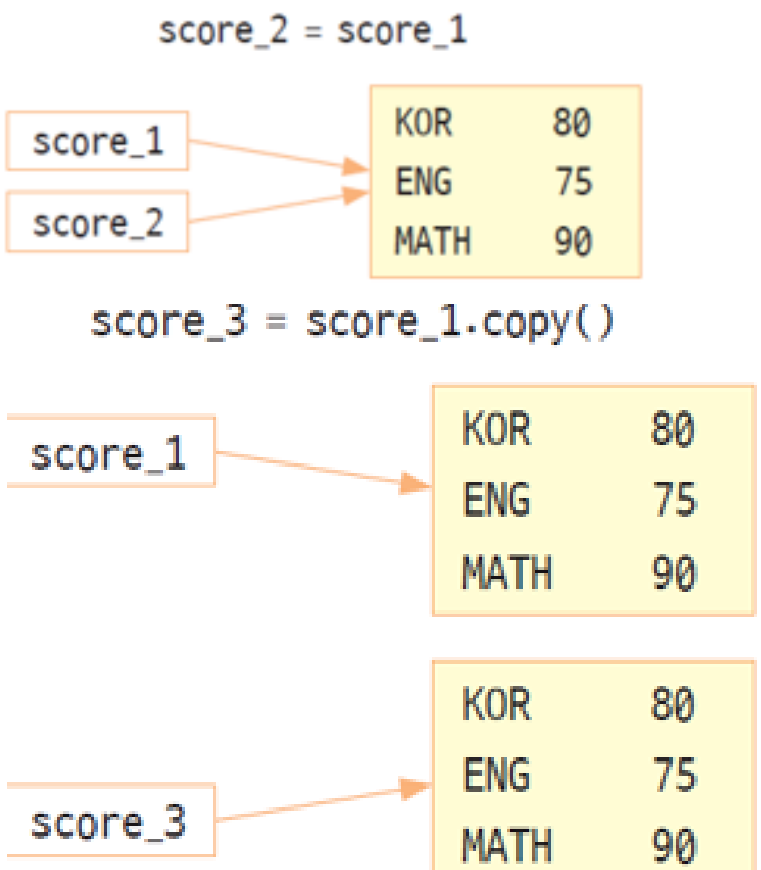
```
>>> salary = salary.drop(1)
```

=

```
>>> salary.drop(1, inplace=True)
```

# 3. 시리즈 객체 내용 변경

- 시리즈 객체 복사
  - = : 동일한 객체를 참조
  - .copy() 메서드를 사용하여 복사




```
score_1 = pd.Series([80, 75, 90],
                    index=['KOR','ENG','MATH'])

score_2 = score_1           # score_1 과 score_2 는 동일한 객체
score_2.loc['KOR'] = 95
score_2
score_1

score_3 = score_1.copy()    # score_1 과 score_3 은 독립된 객체
score_3.loc['KOR'] = 70
score_3
score_1
```

# 실습해보기

- **※ 다음은 어느 보험회사 8개 영업팀(A~H)의 1분기 매출 실적 데이터이다. 각 문제를 해결하기 위한 파이썬 코드를 작성하시오.**

| A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|
| 781 | 650 | 705 | 406 | 580 | 450 | 550 | 640 |

1. 매출액이 500 미만이거나 700을 초과하는 팀들의 이름과 매출액을 출력하시오
1. B팀보다 매출액이 많은 팀들의 이름과 매출액을 출력하시오.
2. 매출액이 600 미만인 팀들의 이름을 출력하시오.
3. 매출액이 600 미만인 팀들의 매출액을 20% 증가시켜 출력하시오.
4. 8개 팀의 매출액 평균, 합계, 표준편차를 출력하시오.
5. A, C팀의 매출액을 각각 810, 820으로 변경한 후 변경 여부를 확인하시오.
6. 신생팀 J팀의 매출액 400을 sales에 추가하시오.
7. J팀의 매출액 정보를 삭제하시오.
8. sales의 내용을 sales2에 복사한 후, sales2의 매출액을 각각 500씩 증가시키고, sales와 sales2의 내용을 비교하시오.