

05 실무함수 구현하기



경고 : 본 강의자료는 연세대학교 학생들을 위해 수업 목적으로 제작.게시된 것이므로, 수업목적 이외의 용도로 사용할 수 없으며, 다른 사람과 공유할 수 없습니다. 위반에 따른 법적 책임은 행위자 본인에게 있습니다.



연세대학교
YONSEI MIRAE
CAMPUS

- ◆ 원하는 데이터 필터링 및 기준열로 정렬하기
- ◆ 찾기 및 참조함수로 원하는 값 찾기, 조건함수 사용하기

5.1 동적 배열 함수

5.2 찾기 및 참조 함수

5.3 논리 및 정보 함수

- 동적 배열 함수는 지정한 범위 내에 있는 데이터 중 원하는 행만 필터링하거나 기준 열에 맞춰 오름차순 또는 내림차순으로 정렬하는 기능을 제공

➢ 파이썬에서는 내장 함수와 pandas 함수를 활용하여 유사한 기능을 보다 더 빠르게 구현

- 실습데이터 불러오기

➢ “근무 유형.xlsx” 엑셀 파일을 work 데이터 프레임으로 저장

```
import pandas as pd    # pandas를 pd라는 이름으로 불러오기
# 동적 배열 함수 실습을 위해 엑셀 파일을 불러와 work에 저장하기
work = pd.read_excel(r"chapter05/근무 유형.xlsx", sheet_name = "Sheet1")
work.head(10)         # 총 16행으로 구성, 상위 10행만 출력
```

	사원번호	성명	부서	근무형태
0	54602	홍길동	제조팀	교대
1	39382	이영희	총무팀	상근
2	56925	김승우	품질관리팀	교대
3	51153	이승훈	회계팀	상근
4	66892	신성우	연구소	상근
5	73849	이철수	영업팀	상근
6	66301	김건호	제조팀	교대
7	90140	이구라	제조팀	교대
8	39814	박애라	총무팀	상근
9	60944	강인표	연구소	상근

원하는 데이터 필터링하기

● 데이터 테이블에서 조건에 맞는 행만 필터링하여 추출 가능

➢ 엑셀에서는 “필터”를 실행하여 필터링, 파이썬에서는 loc[] 또는 loc.isin() 함수를 이용하여 원하는 데이터 행을 추출

엑셀

➢ 데이터 테이블에서 “제조팀”만 필터링

The screenshot illustrates the steps to filter data in Excel. It shows a table with columns A (사원번호), B (성명), C (부서), and D (근무형태). The first table shows the initial data. The second table shows the result after filtering for '제조팀' (Manufacturing Team). The third table shows the result after filtering for '제조팀' only.

Table 1: Initial Data

A	B	C	D
사원번호	성명	부서	근무형태
54602	홍길동	제조팀	교대
39382	이영희	총무팀	상근
56925	김승우	품질관리팀	교대
51153	미승훈	회계팀	상근
66892	신성우	연구소	상근
73849	이철수	영업팀	상근
66301	김건호	제조팀	교대
90140	이구라	제조팀	교대
39814	박애라	총무팀	상근

Table 2: Data after filtering for '제조팀'

A	B	C	D
사원번호	성명	부서	근무형태
54602	홍길동	제조팀	교대
66301	김건호	제조팀	교대
90140	이구라	제조팀	교대
90000	김경호	제조팀	교대
66301	김건호	제조팀	교대

Table 3: Data after filtering for '제조팀' only

A	B	C	D
사원번호	성명	부서	근무형태
54602	홍길동	제조팀	교대
66301	김건호	제조팀	교대

Annotations:

- [A1] 셀을 선택한 다음 [데이터] - [정렬 및 필터]에서 필터를 클릭
- 열 제목에 필터가 적용되어 드롭다운 버튼이 생성되면 이를 클릭하고 “제조팀”만 선택

원하는 데이터 필터링하기

파이썬

➤ loc[] 또는 loc.isin() 함수를 이용하여 원하는 데이터 행을 필터링

- loc[]에서는 열 이름과 필터값을 조건식으로 비교하여 원하는 행을 추출
- loc.isin() 함수에서는 loc[]에 열 이름을 지정하고 isin() 함수에 필터값을 지정하여 원하는 행을 추출

```
pd.loc[pd["열"] == 필터값]          # 열 이름과 필터값을 비교하여 행 추출
pd.loc[pd["열"].isin(["필터값"])]    # 열 이름과 isin 함수의 필터값을 비교하여 행 추출
```

```
work = pd.read_excel(r"chapter05/근무 유형.xlsx", sheet_name = "Sheet1")
w1 = work.loc[work["부서"] == "제조팀"]          # 부서명이 "제조팀"인 행 추출
w2 = work.loc[work["근무형태"].isin(["상근"])]    # 근무형태가 "상근"인 행 추출
display(w1, w2)                                  # 여러 개의 데이터 프레임 w1, w2 한번에 출력
```

w1

	사원번호	성명	부서	근무형태
0	54602	홍길동	제조팀	교대
6	66301	김건호	제조팀	교대
7	90140	이구라	제조팀	교대
10	90000	김경호	제조팀	교대
15	66301	김건호	제조팀	교대

w2, 총 9행 출력됨

	사원번호	성명	부서	근무형태
1	39382	이영희	총무팀	상근
3	51153	이승훈	회계팀	상근
4	66892	신성우	연구소	상근
5	73849	이철수	영업팀	상근
8	39814	박애라	총무팀	상근

원하는 데이터 필터링하기

파이썬

➤ 연산자를 이용하여 여러 개의 조건을 조합하여 필터링하기

"부서"가 "제조팀" 이거나 "근무형태" 가 "교대" 인 데이터 보기

`work.loc[(work["부서"] == "제조팀") | (work["근무형태"] == "교대")]`

	사원번호	성명	부서	근무형태
0	54602	홍길동	제조팀	교대
2	56925	김승우	품질관리팀	교대
6	66301	김건호	제조팀	교대
7	90140	이구라	제조팀	교대
10	90000	김경호	제조팀	교대
13	80185	이상민	품질관리팀	교대
15	66301	김건호	제조팀	교대

`work.loc[~work["근무형태"].isin(["상근"])]`

~(not)

기준 열로 정렬하기

● 데이터 테이블에서 특정 열 값을 기준으로 행 전체를 오름차순 또는 내림차순으로 정렬

➢ 엑셀에서는 “정렬”을, 파이썬에서는 `sort_index()`, `sort_values()` 함수를 이용

엑셀

➢ “직원번호” 기준으로 오름차순 정렬

Excel - 근무유형 - Excel

데이터

정렬

정렬 기준: 직원번호

정렬: 오름차순

직원번호	성명	부서	근무형태
54602	홍길동	제조팀	교대
39382	이영희	총무팀	상근
39382	이영희	총무팀	상근
39814	박애라	총무팀	상근
51153	이승훈	회계팀	상근
54602	홍길동	제조팀	교대
56925	김승우	품질관리팀	교대
60944	강인표	연구소	상근
66301	김건호	제조팀	교대
66301	김건호	제조팀	교대
66892	신성우	연구소	상근
73849	이철수	영업팀	상근
80185	이상민	품질관리팀	교대
88818	백운산	영업팀	상근
90000	김경호	제조팀	교대
90140	이구라	제조팀	교대

[A1] 셀을 선택한 후, [데이터] - [정렬 및 필터]에서 정렬 클릭
[정렬] 대화상자가 나타나면 [열], [정렬 기준], [정렬]에 기준 값을 지정하여 정렬

파이썬

- **sort_values()** 함수를 이용하면 데이터를 특정 열 기준으로 정렬할 수 있음
 - **by** 속성으로 정렬할 열 이름을 지정하고, **ascending** 속성을 **True**(오름차순), **False**(내림차순) 정렬
 - **sort.index()** 함수를 이용하면 데이터 프레임의 인덱스로 정렬할 수 있음

```
pd.sort_values(by = "열", ascending = True/False)    # ascending 생략 시 오름차순 정렬
pd.sort_index(ascending = True/False)
```

```
work = pd.read_excel(r"chapter05/근무 유형.xlsx", sheet_name = "Sheet1")
work.sort_values(by = "사원번호").head()             # 사원번호 기준으로 오름차순 정렬
```

	사원번호	성명	부서	근무형태
11	16736	한라산	영업팀	상근
1	39382	이영희	총무팀	상근
14	39382	이영희	총무팀	상근
8	39814	박애라	총무팀	상근
3	51153	이승훈	회계팀	상근

```
work.sort_index().head()                            # 인덱스 번호순으로 오름차순 정렬해서 상위 5개의 자료 보기
```


중복 행 제거하기

- 행 전체가 중복되거나 일부 값이 중복되는 경우, 중복된 내용이 불필요하다면 제거
 - 엑셀에서는 “중복된 항목 제거”를, 파이썬에서는 duplicated(), drop_duplicates() 함수를 이용

엑셀

- “근무 유형.xlsx” 파일에서 중복되는 행을 제거하고 유니크한 행만 추출

중복된 항목 제거

중복 값을 삭제하려면 중복 값이 있는 열을 하나 이상 선택하십시오.

☒ 모두 선택(A) ☐ 모두 선택 취소(U) ☒ 내 데이터에 머리글 표시(M)

열

- ☒ 사원번호
- ☒ 성명
- ☒ 부서
- ☒ 근무형태

확인 취소

A	B	C	D
사원번호	성명	부서	근무형태
1	16736	한라산	영업팀
2	39382	이영희	총무팀
3	39814	박애라	총무팀
4	51153	이승훈	회계팀
5	54602	홍길동	제조팀
6	56925	김승우	품질관리팀
7	60944	강인표	연구소
8	66301	김건호	제조팀
9	66892	신성우	연구소
10	73849	이철수	영업팀
11	80185	이상민	품질관리팀
12	88818	백운산	영업팀
13	90000	김경호	제조팀
14	90140	이규라	제조팀

[A1] 셀을 선택한 다음 [데이터] - [데이터 도구]에서 중복된 항목 제거를 클릭
[중복된 항목 제거] 대화상자가 나타나면 중복 제거가 필요한 열을 체크하고 [확인] 버튼을 클릭

중복 행 제거하기



파이썬

- `drop_duplicates()` 함수로 중복행을 제거하거나, `uplicated()` 함수를 사용해 중복된 행을 TRUE/FALSE로 출력
 - `work.duplicated()`는 중복행을 True로 출력하고, `work.drop_duplicates()`는 중복행을 찾아 제거

`pd.duplicated()` # 중복되는 행이 있을 경우 **True** 출력
`pd.drop_duplicates()` # 중복되는 행을 제거

```
work = pd.read_excel(r"chapter05/근무 유형.xlsx", sheet_name = "Sheet1")
```

```
work.duplicated() # 중복행을 True로 출력
```

```
work.drop_duplicates() # 중복행 제거
```

	A	B	C	D
1	사원번호	성명	부서	근무형태
2	54602	홍길동	제조팀	교대
3	39382	이영희	총무팀	상근
4	56925	김승우	품질관리팀	교대
8	66301	김건호	제조팀	교대
15	80185	이상민	품질관리팀	교대
16	39382	이영희	총무팀	상근
17	66301	김건호	제조팀	교대

work.drop_duplicates() 결과

	사원번호	성명	부서	근무형태
0	54602	홍길동	제조팀	교대
1	39382	이영희	총무팀	상근
2	56925	김승우	품질관리팀	교대
3	51153	이승훈	회계팀	상근
4	66892	신성우	연구소	상근
5	73849	이철수	영업팀	상근

work.duplicated() 결과

```
0 False
1 False
2 False
3 False
4 False
5 False
6 False
7 False
8 False
9 False
10 False
11 False
12 False
13 False
14 True
15 True
dtype: bool
```

- 찾기 및 참조 함수는 특정 범위 내 정의된 값을 참조할 때 사용 (엑셀 : choose(), vlookup() 함수)
 - 파이썬에서는 pandas의 인덱스를 활용하여 구현
- 실습 데이터 불러오기
 - “식자재 주문.xlsx” 파일의 product, order 워크시트를 데이터 프레임으로 저장

```
import pandas as pd # pandas를 pd라는 이름으로 불러오기
# 찾기 및 참조 함수 실습을 위해 “식자재 주문.xlsx” 엑셀을 불러와 해당 시트를 각각 product, order에 저장하기
product = pd.read_excel(r"chapter05/식자재 주문.xlsx", sheet_name = "product")
order = pd.read_excel(r"chapter05/식자재 주문.xlsx", sheet_name = "order")
display(product, order)
```

product 시트

	제품코드	제품명	단위	단가
0	100-1	참치	캔	2000
1	200-2	생수	개	1000
2	300-3	컵라면	개	800
3	400-3	컵밥	개	2500
4	500-1	마요네즈	그램	3000
5	600-1	케첩	그램	2500
6	700-1	식용유	리터	4000
7	800-2	음료수	병	1500
8	900-1	과자1	봉지	1500
9	950-1	과자2	봉지	2000

order 시트

	대리점명	제품코드	주문수량
0	소망	200-2	200
1	소망	900-1	350
2	소망	400-3	200
3	희망	600-1	500
4	희망	800-2	150
5	자유	950-1	200
6	자유	100-1	100
7	자유	400-3	50
8	희망	200-2	80
9	희망	500-1	320

인덱스로 값 확인하기

- 특정 위치의 문자나 숫자를 인덱스로 지정하고, 인덱스를 원래의 값으로 변환(문자열로 대체)
 - 엑셀에서는 choose() 함수를, 파이썬에서는 map() 함수를 이용

엑셀

- 제품코드의 마지막 자리 숫자가 제조공장의 지역을 나타내는 값일 때 right() 함수로 제품코드 중 일부를 추출한 후, choose() 함수로 인덱스에 대응하는 값을 지정

식자재주문 - Excel

파일 홈 삽입 페이지 레이아웃 수식 데이터 검토 보기 수행할 작업을 알려 주세요. 로그인 공유

클립보드 글꼴 맞춤 표시 형식 스타일 정렬 및 찾기 및

E2 =CHOOSE(RIGHT(A2,1), "청주", "대구", "광주")

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	제품코드	제품명	단위	단가									
2	100-1	참치	캔	2000	청주								
3	200-2	생수	개	1000	대구								
4	300-3	컵라면	개	800	광주								
5	400-3	컵밥	개	2500	광주								
6	500-1	마요네즈	그램	3000	청주								
7	600-1	케찹	그램	2500	청주								
8	700-1	식용유	리터	4000	청주								
9	800-2	음료수	병	1500	대구								
10	900-1	과자1	봉지	1500	청주								
11	950-1	과자2	봉지	2000	청주								
12													
13													
14													
15													
16													
17													
18													
19													

product order

[E2] 셀에

=choose(right(A2,1), "청주", "대구", "광주")를
입력하면 [A2] 셀에서 우측 숫자 1자리를
추출하여 그 값이 1일 경우 청주, 2일 경우 대구,
3일 경우 광주를 반환

파이썬

- dictionary 자료형에 key와 해당되는 value를 미리 정의하고 map() 함수를 이용하여 참조
 - map() 함수는 두번째 인자(iterable)를 첫번째 인자인 함수(Key)에 전달하여 그 결과(Value)를 반환
 - map 함수의 반환 값은 map 객체이기 때문에 해당 자료형을 list 혹은 tuple로 형 변환시켜주어야 함

```
pd.map(function, iterable)    # 반복 가능한 자료형을 함수로 전달, 결과 반환
```

```
product = pd.read_excel(r"chapter05/식자재 주문.xlsx", sheet_name = "product")
mapping = {"1":"청주", "2":"대구", "3":"광주"}    # dictionary 자료형으로 key : value 정의
product["공장"] = product["제품코드"].str[4].map(mapping)    # 제품코드 마지막 값을 key로 사용. 연결된 Value 반환
product.head()
```

	제품코드	제품명	단위	단가	공장
0	100-1	참치	캔	2000	청주
1	200-2	생수	개	1000	대구
2	300-3	컵라면	개	800	광주
3	400-3	컵밥	개	2500	광주
4	500-1	마요네즈	그램	3000	청주

원하는 값 찾기

● 데이터 테이블에서 특정 위치에 해당하는 값을 참조

➢ 엑셀은 vlookup() 함수 사용, 파이썬은 set_index() 함수로 인덱스 열 지정 후 동일한 인덱스를 갖는 열의 값 참조

엑셀

➢ order의 제품코드와 동일한 제품코드가 있는 product 행에서 제품명, 단가를 참조하여 가져오고 주문금액을 계산

product

	A	B	C	D	E	F
1	제품코드	제품명	단위	단가		
2	100-1	참치	캔	2000		
3	200-2	생수	개	1000		
4	300-3	컵라면	개	800		
5	400-3	컵밥	개	2500		
6	500-1	마요네즈	그램	3000		
7	600-1	케찹	그램	2500		
8	700-1	식용유	리터	4000		
9	800-2	음료수	병	1500		
10	900-1	과자1	봉지	1500		
11	950-1	과자2	봉지	2000		

order

	A	B	C	D	E	F
1	대리점명	제품코드	주문수량	제품명	단가	주문금액
2	소망	200-2	200	생수	1000	200000
3	희망	300-3	350	과자1	1500	525000
4	희망	400-3	200	컵밥	2500	500000
5	희망	500-1	500	케찹	2500	1250000
6	희망	600-1	150	음료수	1500	225000
7	희망	700-1	200	과자2	2000	400000
8	자유	800-2	100	참치	2000	200000
9	자유	400-3	50	컵밥	2500	125000
10	희망	200-2	80	생수	1000	80000
11	희망	500-1	320	마요네즈	3000	960000

"order" 시트의 [D1:F1]에 제품명, 단가, 주문금액을 입력

[D2] 셀에 =VLOOKUP(B2, product!\$A\$2:\$D\$11, 2, 0)를 입력하면 [B2] 셀에 있는 제품코드를 "product" 시트의 [A2:D11] 범위에서 찾아 해당 행의 두 번째 값인 제품명을 가져옴

[D5] 셀의 =VLOOKUP(B5, product!\$A\$2:\$D\$11, 4, 0) 도 마찬가지로 "order" 시트의 제품코드를 "product" 시트에서 찾아 해당 행의 네 번째 값인 단가를 가져옴

원하는 값 찾기



파이썬

- 데이터 프레임의 인덱스를 활용하면 다른 데이터 프레임에서 원하는 값을 찾을 수 있음
 - `set_index()` 함수로 데이터 프레임의 특정 열을 인덱스로 설정

```
pd.set_index("열 이름", inplace = True) # index를 "열 이름"으로 설정하고 데이터 프레임에 반영
inplace : 원본 객체를 변경할지 여부
```

```
pd.reset_index(inplace = True) # 데이터 프레임 인덱스 리셋
```

제품코드	제품명	단위	단가
100-1	참치	캔	2000
200-2	생수	개	1000
300-3	컵라면	개	800
400-3	컵밥	개	2500
500-1	마요네즈	그램	3000
600-1	케찹	그램	2500
700-1	식용유	리터	4000
800-2	음료수	병	1500
900-1	과자1	봉지	1500
950-1	과자2	봉지	2000

대리점명	제품코드	주문수량
소망	200-2	200
소망	900-1	350
소망	400-3	200
희망	600-1	500
희망	800-2	150
자유	950-1	200
자유	100-1	100
자유	400-3	50
희망	200-2	80
희망	500-1	320

파이썬

- [제품코드]를 인덱스로 설정하면 두 시트의 “제품코드”가 동일한 행의 데이터를 서로 참조할 수 있다.
- order에 [제품명]이 없어도 동일한 [제품코드]를 가진 product의 “제품명”을 추출해 올 수 있다.

제품코드	제품명	단위	단가
	product	order	+

대리점명	제품코드	주문수량
	product	order

```
product = pd.read_excel(r"chapter05/식자재 주문.xlsx", sheet_name = "product")
```

```
order = pd.read_excel(r"chapter05/식자재 주문.xlsx", sheet_name = "order" )
```

```
product.set_index("제품코드", inplace = True)
```

```
order.set_index("제품코드", inplace = True)
```

```
order["제품명"] = product["제품명"]
```

```
order["단가"] = product["단가"]
```

```
order["주문금액"] = order["주문수량"] * order["단가"]
```

```
order
```

product["제품코드"]를 인덱스로 설정

order["제품코드"]를 인덱스로 설정

동일 인덱스의 product["제품명"]을 order에 추가

동일 인덱스의 product["단가"]를 order에 추가

주문금액을 계산하여 order에 추가

order 데이터 프레임에 [제품명, 단가, 주문금액] 열이 추가됨

inplace = True 를 생략하면 원본 인덱스 번호 순으로 출력됨

	대리점명	주문수량	제품명	단가	주문금액
제품코드	-	-	-	-	-
200-2	소망	200	생수	1000	200000
900-1	소망	350	과자 1	1500	525000
400-3	소망	200	커피	2500	500000
600-1	희망	500	커피	2500	1250000
800-2	희망	150	음료수	1500	225000
950-1	자유	200	과자 2	2000	400000
100-1	자유	100	참치	2000	200000

- 논리 함수는 조건에 따라 작업을 수행하며 정보 함수는 날짜나 시간을 계산할 수 있는 기능을 제공
 - 대표적인 논리함수로는 조건문이 있으며 엑셀이나 파이썬 모두 사용방법은 비슷함
- 실습 데이터 불러오기
 - “과일 주문.xlsx” 엑셀 파일을 불러와 fruit 데이터 프레임으로 저장

```
import pandas as pd                # pandas를 pd라는 이름으로 불러오기
fruit = pd.read_excel(r"chapter05/과일 주문.xlsx", sheet_name = "Sheet1")
fruit.head()
```

	품목	수량	단가	금액	판매일자	유통기한
0	사과	100	1500	150000	2021-09-05	2021-10-31
1	귤	500	500	250000	2021-09-12	2021-10-31
2	배	150	2000	300000	2021-10-09	2021-11-10
3	참외	250	1200	300000	2021-10-12	2021-11-10
4	한라봉	100	2500	250000	2021-11-10	2021-12-10

조건 함수 사용하기

● 조건 함수는 주어진 조건식의 결과에 따라 별도의 작업을 수행하는 기능을 제공

➤ 엑셀, 파이썬 모두 if() 함수를 사용하며, 조건이 다양할 경우 if() 함수를 중첩해서 사용하거나 elif, else 문을 추가

엑셀

➤ if() 함수를 이용해 품목별 금액에 따라 지정한 금액대를 입력

과일주문 - Excel

파일 홈 삽입 페이지 레이아웃 수식 데이터 검토 보기 수행할 작업을 알려 주세요. 로그인 공유

삽입 텍스트 줄 바꿈 일반 조건부 서식 표 서식 스타일 삽입 삭제 서식

G2 : =IF(D2>=200000, IF(D2>=500000, "50만원이상", "20만원이상"), "20만원미만")

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	품목	수량	단가	금액	판매일자	유통기한							
2	사과	100	1500	150000	2021-09-05	2021-10-31	20만원미만						
3	귤	500	500	250000	2021-09-12	2021-10-31	20만원이상						
4	배	150	2000	300000	2021-10-09	2021-11-10	50만원이상						
5	참외	250	1200	300000	2021-10-12	2021-11-10	20만원이상						
6	한라봉	100	2500	250000	2021-11-10	2021-12-10	20만원이상						
7	토마토	800	2000	1600000	2021-11-23	2021-12-10	50만원이상						
8	포도	200	3500	700000	2021-12-10	2021-12-10	20만원이상						
9	딸기	100	1200	120000	2021-12-10	2021-12-10	20만원미만						
10	망고	200	3000	600000	2021-12-10	2021-12-10	50만원이상						
11	수박	300	5000	1500000	2021-12-10	2021-12-10	50만원이상						

=IF(D2>=200000, IF(D2>=500000, "50만원이상", "20만원이상"), "20만원미만")

[G2] 셀에 =IF(D2>=200000, IF(D2>=500000, "50만원이상", "20만원이상"), "20만원미만")을 입력하면 [D2] 셀의 값이 500000원 이상일 때 "50만원이상", 200000원 이상일 때 "20만원이상", 그 외는 "20만원미만"으로 결과값을 구할 수 있음

파이썬

- 데이터 프레임 내 값을 불러와서 조건을 판단한 후 특정 위치의 값을 변경하기 위해 for 문과 if 문을 활용
 - for 문에 `enumerate()` 함수를 같이 사용하면 데이터 프레임에 있는 자료를 인덱스와 함께 하나씩 불러올 수 있음

```
if ~ elif ~ else      # 조건에 따라 코드 실행
enumerate(변수)        # 변수 내 값과 인덱스를 반환
```

```
fruit = pd.read_excel(r"과일 주문.xlsx", sheet_name = "Sheet1")
fruit["비고"] = ""                                     # [비고] 열 생성
for idx, x in enumerate(fruit["금액"]):               # [금액] 열 값과 인덱스를 하나씩 반환
    if x >= 500000:                                     # x 값(금액)이 500000 이상이면, [비고] 열에 "50만원 이상" 저장
        fruit["비고"].loc[idx] = "50만원 이상"
    elif x >= 200000:                                   # x 값(금액)이 200000 이상이면, [비고] 열에 "20만원 이상" 저장
        fruit["비고"].loc[idx] = "20만원 이상"
    else:                                                # x 값(금액)이 200000 미만이면, [비고] 열에 "20만원 미만" 저장
        fruit["비고"].loc[idx] = "20만원 미만"
fruit.head(6)
```

fruit 데이터 프레임에 [비고]열이 추가됨

	품목	수량	단가	금액	판매일자	유통기한	비고
0	사과	100	1500	150000	2021-09-05	2021-10-31	20만원 미만
1	귤	500	500	250000	2021-09-12	2021-10-31	20만원 이상
2	배	150	2000	300000	2021-10-09	2021-11-10	20만원 이상
3	참외	250	1200	300000	2021-10-12	2021-11-10	20만원 이상
4	한라봉	100	2500	250000	2021-11-10	2021-12-10	20만원 이상
5	토마토	800	2000	1600000	2021-11-23	2021-12-10	50만원 이상

pandas 경고 발생과 해결 방법

- ◆ pandas에서 원본 Dataframe의 일부를 복사하거나 인덱싱 후 값을 수정할 때 발생할 수 있는 "SettingWithCopyWarning or "SettingWithCopyError"의 원인과 해결 방법

```
C:\Users\wnicen\AppData\Local\Temp\ipykernel_11676\2779826000.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
fruit["비고"].loc[idx]="20만원 미만"
C:\Users\wnicen\AppData\Local\Temp\ipykernel_11676\2779826000.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

- ◆ 경고는 발생하지만 결과는 나타난다. 이러한 현상을 Chained Assignment라 한다. 비정상적인 코드 사용이나 패키지의 버전 문제일 수 있으므로 아래와 같은 방법을 사용한다.

1) 경고를 끈다.

```
# 오류(SettingWithCopyError 발생)
pd.set_option('mode.chained_assignment', 'raise') # Setting WithCopyError

# 경고(SettingWithCopyWarning 발생. 기본값)
pd.set_option('mode.chained_assignment', 'warn') # Setting WithCopyWarning

# 무시
pd.set_option('mode.chained_assignment', None) # 경고를 꺼버린다.
```

2) 명시적으로 복사해서 사용한다.

- 원본 Dataframe의 일부를 잘라서 편집해야 하는 경우 명시적으로 복사해서 사용하는 것이 좋다.

날짜 및 시간 함수

● 날짜 및 시간을 연산하거나 표기 방법을 변경

➤ 엑셀에서는 today(), year(), month(), day() 함수를, 파이썬에서는 datetime.now() 함수를 사용

엑셀

➤ 판매일자에서 “일” 을 추출, 유통기한과 오늘날짜의 차이를 계산하여 유통기한 경과일수 산정

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	품목	수량	단가	금액	판매일자	유통기한	판매일	오늘날짜	유통기한 경과일수				
2	사과	100	1500	150000	2021-09-05	2021-10-31	5	2022-01-24	85				
3	귤	500	500	250000	2021-09-12	2021-10-31	12	2022-01-24	85				
4	배	150	2000	300000	2021-10-09	2021-11-30	9	2022-01-24	5				
5	참외	250	1200	300000	2021-10-12	2021-11-30	12	2022-01-24	5				
6	한라봉	100	2500	250000	2021-11-10	2021-12-31	10	2022-01-24	4				
7	토마토	800	2000	160000	2021-11-10	2021-12-31	23	2022-01-24	45				
8	포도	200	3500	700000	2021-11-10	2021-12-31	17	2022-01-24	35				
9	딸기	100	1200	120000	2021-12-18	2021-12-20	18	2022-01-24					
10	망고	200	3000	600000	2021-12-20	2021-12-31	20	2022-01-24					
11	수박	300	6000	1800000	2021-12-20	2021-12-31	20	2022-01-24					

[G2] 셀에 =DAY(B2)를 입력하여 판매일자를 추출

[H2] 셀에 =TODAY()를 입력하여 오늘날짜 추출

[I2] 셀에 =DATEDIF(F2, H2, "D")를 입력하면 유통기한과 오늘날짜의 차이를 이용해 유통기한 경과일수를 계산할 수 있음

파이썬

- 날짜를 출력하거나 계산 하려면 자료형을 날짜형으로 변경해야 함
 - 날짜형 포맷으로는 출력 시 사용하는 datetime 자료형, 날짜계산시 사용하는 timedelta 자료형이 있음

```
import pandas as pd
```

<code>pd.to_datetime()</code>	# 문자열 데이터를 날짜형 데이터로 변환
<code>pd.dt.day</code>	# datetime 자료형에서 일자만 출력
<code>pd.dt.days</code>	# timedelta 자료형에서 일자만 출력
<code>pd.datetime.now()</code>	# 오늘 날짜 출력 (deprecated 경고 발생)

파이썬

➤ 날짜 패키지 datetime

```
from datetime import datetime
```

```
# datetime 패키지 import
```

```
today = datetime.now()
```

```
# 현재 날짜 시간을 today 변수에 저장
```

```
print(today.year)
```

```
# today 변수에서 년 데이터만 출력
```

```
print(today.month)
```

```
# today 변수에서 월 데이터만 출력
```

```
print(today.day)
```

```
# today 변수에서 일 데이터만 출력
```

```
time1 = datetime(2019, 10, 1, 15, 30, 1)
```

```
# time1에 임의의 날짜 데이터 저장
```

```
time2 = datetime.now()
```

```
# time2에 현재 날짜/시간 저장
```

```
print((time2 - time1).days, "일")
```

```
# 두 변수 간 일 차이 출력
```

```
print((time2 - time1).seconds, "초")
```

```
# 두 변수 간 초 단위 차이 출력
```

```
print((time2 - time1).seconds / 3600, "시간")
```

```
# 두 변수 간 시간 단위 차이 출력
```

파이썬

➤ pandas 패키지와 datetime 패키지를 이용한 날짜형 변경

```
import pandas as pd
from datetime import datetime

fruit = pd.read_excel(r"chapter05/과일 주문.xlsx", sheet_name = "Sheet1")
pd.to_datetime(fruit["판매일자"])          # 문자형 데이터를 날짜형으로 변환
pd.to_datetime(fruit["유통기한"])
fruit["판매일"] = fruit["판매일자"].dt.day  # datetime 자료형에서 일자만 출력

#fruit["오늘 날짜"] = pd.datetime.now()      # [오늘 날짜] 열 생성 --> 경고 발생
fruit["오늘 날짜"] = datetime.now()          # [오늘 날짜] 열 생성
fruit["유통기한 경과일수"] = (fruit["오늘 날짜"] - fruit["유통기한"]).dt.days
fruit.head()
```

	품목	수량	단가	금액	판매일자	유통기한	판매일	오늘날짜	유통기한 경과일수
0	사과	100	1500	150000	2021-09-05	2021-10-31	5	2022-07-01 14:10:30.786232	243
1	귤	500	500	250000	2021-09-12	2021-10-31	12	2022-07-01 14:10:30.786232	243
2	배	150	2000	300000	2021-10-09	2021-11-10	9	2022-07-01 14:10:30.786232	233
3	참외	250	1200	300000	2021-10-12	2021-11-10	12	2022-07-01 14:10:30.786232	233
4	한라봉	100	2500	250000	2021-11-10	2021-12-10	10	2022-07-01 14:10:30.786232	203