Chapter 05 업무에 자주 쓰는 실무 함수 구현하기

5.1 동적 배열 함수

실습 데이터 불러오기

In [2]:

```
import pandas as pd      # pandas를 pd라는 이름으로 불러오기

# 동적 배열 함수 실습을 위해 액셀 파일을 불러와 work에 저장하기
work = pd.read_excel(r"c:\works\chapter05\근무 유형.xlsx", sheet_name = "Sheet1")
work
```

원하는 데이터 필터링하기

In [2]:

```
work = pd.read_excel(r"c:\works\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)          # w1, w2 출력
```

	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

	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
9	60944	강인표	연구소	상근
11	16736	한라산	영업팀	상근
12	88818	백운산	영업팀	상근
14	39382	이영희	총무팀	상근

In [3]:

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

Out[3]:

	사원번호	성명	부서	근무형태
0	54602	홍길동	제조팀	교대
2	56925	김승우	품질관리팀	교대

	사원번호	성명	부서	근무형태
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

In [4]:

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

Out[4]:

	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

기준 열로 정렬하기

In [5]:

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

Out[5]:

	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

In [6]:

```
work.sort_index().head()     # 인덱스 기준으로 오름차순 정렬
```

Out[6]:

	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

중복 행 제거하기

In [7]:

```
work = pd.read_excel(r"c:\works\chapter05\근무 유형.xlsx", sheet_name = "Sheet1")
work.duplicated()    # 중복 행을 True로 출력
```

Out[7]:

```
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
```

In [8]:

```
work.drop_duplicates()    # 중복 행 제거
```

Out[8]:

	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
10	90000	김경호	제조팀	교대
11	16736	한라산	영업팀	상근
12	88818	백운산	영업팀	상근
13	80185	이상민	품질관리팀	교대

5.2 찾기 및 참조 함수

실습 데이터 불러오기

In [9]:

```
# pandas를 pd라는 이름으로 불러오기
import pandas as pd

# 찾기 및 참조 함수 실습을 위해 "식자재 주문.xlsl"을 불러와 product, order에 저장하기
product = pd.read_excel(r"c:\works\chapter05\식자재 주문.xlsx", sheet_name = "product")
order = pd.read_excel(r"c:\works\chapter05\식자재 주문.xlsx", sheet_name = "order")
product
```

Out[9]:

	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	케찹	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

In [10]:

```
order
```

Out[10]:

	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

In [11]:

```
product = pd.read_excel(r"c:\works\chapter05\식자재 주문.xlsx", sheet_name = "product")
mapping = {"1":"청주", "2":"대구", "3":"광주"}  # 딕셔너리 자료형으로 Key : Value 정의
product["공장"] = product["제품코드"].str[4].map(mapping) # 제품코드 마지막 값을 Key로 사용
product.head()
```

Out[11]:

	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

In [12]:

```
product = pd.read_excel(r"c:\works\chapter05\식자재 주문.xlsx", sheet_name = "product")
order = pd.read_excel(r"c:\works\chapter05\식자재 주문.xlsx", sheet_name = "order" )
product.set_index("제품코드", inplace = True) # product["제품코드"]를 인덱스로 설정
order.set_index("제품코드", inplace = True)  # order["제품코드"]를 인덱스로 설정
order["제품명"] = product["제품명"]     # 동일 인덱스의 product["제품명"]을 order에 추가
order["단가"] = product["단가"]      # 동일 인덱스의 product["단가"]를 order에 추가
order["주문금액"] = order["주문수량"] * order["단가"] # 주문금액 계산하여 order에 추가
order
```

Out[12]:

제품코드	대리점명	주문수량	제품명	단가	주문금액
200-2	소망	200	생수	1000	200000
900-1	소망	350	과자1	1500	525000
400-3	소망	200	컵밥	2500	500000
600-1	희망	500	케찹	2500	1250000
800-2	희망	150	음료수	1500	225000
950-1	자유	200	과자2	2000	400000
100-1	자유	100	참치	2000	200000
400-3	자유	50	컵밥	2500	125000
200-2	희망	80	생수	1000	80000
500-1	희망	320	마요네즈	3000	960000

In [13]:

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

5.3 논리 및 정보 함수

실습 데이터 불러오기

In [14]:

```
import pandas as pd  # pandas를 pd라는 이름으로 불러오기
# 논리 함수 실습을 위해 "과일 주문.xlsx" 파일을 불러와 fruit에 저장
fruit = pd.read_excel(r"c:\works\chapter05\과일 주문.xlsx", sheet_name = "Sheet1")
fruit
```

Out[14]:

	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
6	포도	200	3500	700000	2021-12-17	2021-12-20	50만원 이상
7	딸기	100	1200	120000	2021-12-18	2021-12-20	20만원 미만
8	망고	200	3000	600000	2021-12-20	2021-12-31	50만원 이상
9	수박	300	6000	1800000	2021-12-20	2021-12-31	50만원 이상

조건 함수 사용하기

In [15]:

```
fruit = pd.read_excel(r"c:\works\chapter05\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:  # x값(금액)이 500000 이상이면, [비고] 열에 "50만원 이상" 저장
        fruit["비고"].loc[idx] = "50만원 이상"
    elif x >= 200000: # x값(금액)이 200000 이상이면, [비고] 열에 "20만원 이상" 저장
        fruit["비고"].loc[idx] = "20만원 이상"
    else:    # x값(금액)이 200000 미만이면, [비고] 열에 "20만원 미만" 저장
        fruit["비고"].loc[idx] = "20만원 미만"
fruit.head(6)       # fruit 출력
```

```
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:9: FutureWarning: Chai
nedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate obje
ct on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy

  fruit["비고"].loc[idx] = "20만원 미만"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:9: SettingWithCopyWarn
ing:
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_g
uide/indexing.html#returning-a-view-versus-a-copy
  fruit["비고"].loc[idx] = "20만원 미만"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:7: FutureWarning: Chai
nedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate obje
ct on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy

  fruit["비고"].loc[idx] = "20만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:7: SettingWithCopyWarn
ing:
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_g
uide/indexing.html#returning-a-view-versus-a-copy
  fruit["비고"].loc[idx] = "20만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:7: FutureWarning: Chai
nedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate obje
ct on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:
```

```
df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy

  fruit["비고"].loc[idx] = "20만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:7: SettingWithCopyWarn
ing:
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_g
uide/indexing.html#returning-a-view-versus-a-copy
  fruit["비고"].loc[idx] = "20만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:7: FutureWarning: Chai
nedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate obje
ct on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy

  fruit["비고"].loc[idx] = "20만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:7: SettingWithCopyWarn
ing:
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_g
uide/indexing.html#returning-a-view-versus-a-copy
  fruit["비고"].loc[idx] = "20만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:7: FutureWarning: Chai
nedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate obje
ct on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy

  fruit["비고"].loc[idx] = "20만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:7: SettingWithCopyWarn
ing:
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_g
uide/indexing.html#returning-a-view-versus-a-copy
  fruit["비고"].loc[idx] = "20만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:5: FutureWarning: Chai
nedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate obje
ct on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:
```

```
df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy

  fruit["비고"].loc[idx] = "50만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:5: SettingWithCopyWarn
ing:
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_g
uide/indexing.html#returning-a-view-versus-a-copy
  fruit["비고"].loc[idx] = "50만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:5: FutureWarning: Chai
nedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate obje
ct on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy

  fruit["비고"].loc[idx] = "50만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:5: SettingWithCopyWarn
ing:
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_g
uide/indexing.html#returning-a-view-versus-a-copy
  fruit["비고"].loc[idx] = "50만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:9: FutureWarning: Chai
nedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate obje
ct on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy

  fruit["비고"].loc[idx] = "20만원 미만"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:9: SettingWithCopyWarn
ing:
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_g
uide/indexing.html#returning-a-view-versus-a-copy
  fruit["비고"].loc[idx] = "20만원 미만"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:5: FutureWarning: Chai
nedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate obje
ct on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:
```

```
df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy

  fruit["비고"].loc[idx] = "50만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:5: SettingWithCopyWarn
ing:
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_g
uide/indexing.html#returning-a-view-versus-a-copy
  fruit["비고"].loc[idx] = "50만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:5: FutureWarning: Chai
nedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate obje
ct on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy

  fruit["비고"].loc[idx] = "50만원 이상"
C:\Users\poscouser\AppData\Local\Temp\ipykernel_14824\662987635.py:5: SettingWithCopyWarn
ing:
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_g
uide/indexing.html#returning-a-view-versus-a-copy
  fruit["비고"].loc[idx] = "50만원 이상"
```

Out[15]:

	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**

날짜 및 시간 함수

날짜 및 시간 함수 수정

기존 : `fruit["오늘 날짜"] = pd.datetime.now()`
수정 : `fruit["오늘 날짜"] = datetime.now()`

In [4]:

```
import pandas as pd
from datetime import datetime

fruit = pd.read_excel(r"c:\works\chapter05\과일 주문.xlsx", sheet_name = "Sheet1")
```

```
pd.to_datetime(fruit["판매일자"])    # 날짜형으로 변환
pd.to_datetime(fruit["유통기한"])
fruit["판매일"] = fruit["판매일자"].dt.day  # datetime 자료형에서 일자만 출력
fruit["오늘 날짜"] = datetime.now()  # [오늘 날짜] 열 생성
fruit["유통기한 경과일수"] = (fruit["오늘 날짜"] - fruit["유통기한"]).dt.days
fruit.head()
```

Out[4]:

	품목	수량	단가	금액	판매일자	유통기한	비고	판매일	오늘 날짜	유통기한 경과일수
0	사과	**100**	**1500**	**150000**	**2021-09-05**	**2021-10-31**	20만원 미만	**5**	**2024-06-28 14:44:21.116398**	**971**
1	귤	**500**	**500**	**250000**	**2021-09-12**	**2021-10-31**	20만원 이상	**12**	**2024-06-28 14:44:21.116398**	**971**
2	배	**150**	**2000**	**300000**	**2021-10-09**	**2021-11-10**	20만원 이상	**9**	**2024-06-28 14:44:21.116398**	**961**
3	참외	**250**	**1200**	**300000**	**2021-10-12**	**2021-11-10**	20만원 이상	**12**	**2024-06-28 14:44:21.116398**	**961**
4	한라봉	**100**	**2500**	**250000**	**2021-11-10**	**2021-12-10**	20만원 이상	**10**	**2024-06-28 14:44:21.116398**	**931**

In [17]:

```
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
```

```
2024
5
10
```

** 시간 변경

기존 : time1 = datetime(2019, 10, 1, 15, 30, 1)
변경 : time1 = datetime(2024, 2, 1, 15, 30, 1)

In [3]:

```
from datetime import datetime
time1 = datetime(2024, 2, 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
```

```
151 일
65526 초
18.201666666666668 시간
```

In [ ]:

```

```