In [2]:

```
import pandas as pd

df = pd.DataFrame({
        'float' : [1.0, 2.0],
        'int' : [1, 2],
        'datetime' : [pd.Timestamp('20200101'), pd.Timestamp('20200102')],
        'bool' : [True, False],
        'object' : [1, '-'],
        'float2' : [1.0, 2]
    })
```

In [3]:

```
df
```

Out[3]:

	float	int	datetime	bool	object	float2
0	1.0	1	2020-01-01	True	1	1.0
1	2.0	2	2020-01-02	False	-	2.0

판다스 자료형 다루기

판다스에서 제공하는 기본 데이터 타입

- **int64 :** 정수형
- **float64 :** 실수형
- **bool :** 논리형
- **object :** 문자열
- **category :** 카테고리
- **datetime64 :** 날짜 / 시간

1. 자료형 확인

```
DataFrame.dyptes
Series.dtype
```

- 한 시리즈에 문자열과 숫자, 문자열과 **bool** 등으로 데이터타입이 혼합되어 있으면 ***object형으로 결정***된다.
- 한 시리즈에 정수와 실수가 혼합되어 있으면 ***float4형으로*** 결정된다.

1.1. 데이터프레임의 자료형 확인

In [4]:

```
# 데이터프레임의 자료형 확인
df.dtypes
```

Out[4]:

```
float              float64
int                  int64
datetime    datetime64[ns]
bool                   bool
object               object
float2              float64
dtype: object
```

1.2. 시리즈의 자료형 확인

In [5]:

```
df['int'].dtype
```

Out[5]:

```
dtype('int64')
```

1.3. 자료형이 혼합된 컬럼의 자료형 확인

In [6]:

```
# 숫자형과 문자형이 혼합되어 있는 경우 각 데이터의 자료형 확인
type(df.loc[0, 'object']) # type(1)
```

Out[6]:

```
int
```

In [7]:

```
print(df.loc[1, 'object']) # '-'
print(type(df.loc[1, 'object']))
```

```
-
<class 'str'>
```

2. 판다스 자료형 변환하기

`DataFrame.astype(자료형)`
`Series.astype(자료형)`

- 자료형 변경이 불가능한 경우 **error**

`pd.to_numeric(시리즈, errors = 에러 처리 옵션)`
에러 처리 옵션 :

- **ignore :** 숫자로 변경할 수 없는 값이 있으면 작업하지 않음
- **coerce :** 숫자로 변경할 수 없는 값은 **NaN**으로 설정
- **raise :** 숫자로 변경할 수 없는 값이 있으면 에러 발생**(default)**

In [8]:

```
# 샘플 데이터
df = pd.DataFrame(
    {
        'col1' : [1, 2],
        'col2' : [3, 4]
    }
)
df
```

Out[8]:

	col1	col2
0	1	3
1	2	4

In [9]:

```
df.dtypes
```

Out[9]:

```
col1    int64
col2    int64
dtype: object
```

2.1. 데이터프레임 전체 자료형 반환

In [10]:

```
# 실수형으로 변환
df.astype('float64')
```

Out[10]:

	col1	col2
0	1.0	3.0
1	2.0	4.0

In [11]:

```
df.astype('float64').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   col1    2 non-null      float64
 1   col2    2 non-null      float64
dtypes: float64(2)
memory usage: 160.0 bytes
```

In [12]:

```
# 문자열로 변환
df.astype('str')
```

Out[12]:

	col1	col2
0	1	3
1	2	4

In [13]:

```
df.astype('str').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   col1    2 non-null      object
 1   col2    2 non-null      object
dtypes: object(2)
memory usage: 160.0+ bytes
```

In [14]:

```
# 정수형으로 변환
df.astype('int64')
```

Out[14]:

	col1	col2
0	1	3
1	2	4

In [15]:

```
df.astype('int64').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   col1    2 non-null      int64
 1   col2    2 non-null      int64
dtypes: int64(2)
memory usage: 160.0 bytes
```

2.2. 컬럼의 자료형 변환

In [16]:

```
# col2의 자료형만 str로 변환
df['col2'] = df['col2'].astype('str')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   col1    2 non-null      int64
 1   col2    2 non-null      object
dtypes: int64(1), object(1)
memory usage: 160.0+ bytes
```

In [17]:

```
# col2의 자료형만 float64로 변환
df['col2'] = df['col2'].astype('float64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   col1    2 non-null      int64
 1   col2    2 non-null      float64
dtypes: float64(1), int64(1)
memory usage: 160.0 bytes
```

In [18]:

```
# col2의 자료형만 int로 변환
df['col2'] = df['col2'].astype('int64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   col1    2 non-null      int64
 1   col2    2 non-null      int64
dtypes: int64(2)
memory usage: 160.0 bytes
```

2.3. 자료형이 혼합된 컬럼의 자료형 변환

2.3. 자료형이 혼합된 칼럼의 자료형 변환

In [19]:

```
# 샘플 데이터
df = pd.DataFrame(
    {
        'col1' : [1, 2],
        'col2' : [3, 4],
        'col3' : [5, '-']
    }
)
df
```

Out[19]:

	col1	col2	col3
0	1	3	5
1	2	4	-

In [20]:

```
df.dtypes
```

Out[20]:

```
col1     int64
col2     int64
col3    object
dtype: object
```

In [21]:

```
# col3 칼럼을 str로 변환
df['col3'] = df['col3'].astype('str')
df.dtypes
```

Out[21]:

```
col1     int64
col2     int64
col3    object
dtype: object
```

In [22]:

```
# 변경할 수 없는 자료가 섞여있으면 error
df['col3'] = df['col3'].astype('int64')
df.dtypes
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [22], line 2
      1 # 변경할 수 없는 자료가 섞여있으면 error
----> 2 df['col3'] = df['col3'].astype('int64')
      3 df.dtypes

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\generic.py:5912, in
NDFrame.astype(self, dtype, copy, errors)
   5905     results = [
   5906         self.iloc[:, i].astype(dtype, copy=copy)
   5907         for i in range(len(self.columns))
   5908     ]
   5910 else:
   5911     # else, only a single dtype is given
-> 5912     new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   5913     return self._constructor(new_data).__finalize__(self, method="astype")
   5915 # GH 33113: handle empty frame or series

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\internals\managers.
py:419, in BaseBlockManager.astype(self, dtype, copy, errors)
```

```
    418 def astype(self: T, dtype, copy: bool = False, errors: str = "raise") -> T:
--> 419     return self.apply("astype", dtype=dtype, copy=copy, errors=errors)

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\internals\managers.
py:304, in BaseBlockManager.apply(self, f, align_keys, ignore_failures, **kwargs)
    302         applied = b.apply(f, **kwargs)
    303     else:
--> 304         applied = getattr(b, f)(**kwargs)
    305 except (TypeError, NotImplementedError):
    306     if not ignore_failures:

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\internals\blocks.py
:580, in Block.astype(self, dtype, copy, errors)
    562 """
    563 Coerce to the new dtype.
    564
   (...)
    576 Block
    577 """
    578 values = self.values
--> 580 new_values = astype_array_safe(values, dtype, copy=copy, errors=errors)
    582 new_values = maybe_coerce_values(new_values)
    583 newb = self.make_block(new_values)

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\dtypes\cast.py:1292
, in astype_array_safe(values, dtype, copy, errors)
   1289     dtype = dtype.numpy_dtype
   1291 try:
-> 1292     new_values = astype_array(values, dtype, copy=copy)
   1293 except (ValueError, TypeError):
   1294     # e.g. astype_nansafe can fail on object-dtype of strings
   1295     #  trying to convert to float
   1296     if errors == "ignore":

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\dtypes\cast.py:1237
, in astype_array(values, dtype, copy)
   1234     values = values.astype(dtype, copy=copy)
   1236 else:
-> 1237     values = astype_nansafe(values, dtype, copy=copy)
   1239 # in pandas we don't store numpy str dtypes, so convert to object
   1240 if isinstance(dtype, np.dtype) and issubclass(values.dtype.type, str):

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\dtypes\cast.py:1154
, in astype_nansafe(arr, dtype, copy, skipna)
   1150 elif is_object_dtype(arr.dtype):
   1151
   1152     # work around NumPy brokenness, #1987
   1153     if np.issubdtype(dtype.type, np.integer):
-> 1154         return lib.astype_intsafe(arr, dtype)
   1156     # if we have a datetime/timedelta array of objects
   1157     # then coerce to a proper dtype and recall astype_nansafe
   1159     elif is_datetime64_dtype(dtype):

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\_libs\lib.pyx:668, in pa
ndas._libs.lib.astype_intsafe()

ValueError: invalid literal for int() with base 10: '-'
```

3. 자료형이 혼합된 컬럼을 숫자형으로 변경

`pd.to_numeric(컬럼, errors='변경 방식')`

errors 변경 방식 :

- **'ignore' :** 숫자로 변경할 수 없는 값이 있으면 작업하지 않음
- **'coerce' :** 숫자로 변경할 수 없는 값이 있으면 **NaN**으로 설정
- **'raise' :** 숫자로 변경할 수 없는 값이 있으면 에러 발생 **(default)**

3.1. astype으로 변환

In [23]:

```
# 모든 값을 숫자로 변경할 수 있음
s1 = pd.Series(['1.0', '2', -3])
s1.astype('float64')
```

Out[23]:

```
0    1.0
1    2.0
2   -3.0
dtype: float64
```

In [24]:

```
# 숫자로 변경할 수 없는 값이 섞여 있음
s2 = pd.Series(['-1.0', '2', '-3', 'a'])
s2.astype('float64')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [24], line 3
      1 # 숫자로 변경할 수 없는 값이 섞여 있음
      2 s2 = pd.Series(['-1.0', '2', '-3', 'a'])
----> 3 s2.astype('float64')

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\generic.py:5912, in
NDFrame.astype(self, dtype, copy, errors)
   5905     results = [
   5906         self.iloc[:, i].astype(dtype, copy=copy)
   5907         for i in range(len(self.columns))
   5908     ]
   5910 else:
   5911     # else, only a single dtype is given
-> 5912     new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   5913     return self._constructor(new_data).__finalize__(self, method="astype")
   5915 # GH 33113: handle empty frame or series

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\internals\managers.
py:419, in BaseBlockManager.astype(self, dtype, copy, errors)
    418 def astype(self: T, dtype, copy: bool = False, errors: str = "raise") -> T:
--> 419     return self.apply("astype", dtype=dtype, copy=copy, errors=errors)

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\internals\managers.
py:304, in BaseBlockManager.apply(self, f, align_keys, ignore_failures, **kwargs)
    302         applied = b.apply(f, **kwargs)
    303     else:
--> 304         applied = getattr(b, f)(**kwargs)
    305 except (TypeError, NotImplementedError):
    306     if not ignore_failures:

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\internals\blocks.py
:580, in Block.astype(self, dtype, copy, errors)
    562 """
    563 Coerce to the new dtype.
    564
   (...)
    576 Block
    577 """
    578 values = self.values
--> 580 new_values = astype_array_safe(values, dtype, copy=copy, errors=errors)
    582 new_values = maybe_coerce_values(new_values)
    583 newb = self.make_block(new_values)

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\dtypes\cast.py:1292
, in astype_array_safe(values, dtype, copy, errors)
   1289     dtype = dtype.numpy_dtype
   1291 try:
-> 1292     new values = astype array(values, dtype, copy=copy)
```

```
   1293 except (ValueError, TypeError):
   1294     # e.g. astype_nansafe can fail on object-dtype of strings
   1295     #  trying to convert to float
   1296     if errors == "ignore":

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\dtypes\cast.py:1237
, in astype_array(values, dtype, copy)
   1234     values = values.astype(dtype, copy=copy)
   1236 else:
-> 1237     values = astype_nansafe(values, dtype, copy=copy)
   1239 # in pandas we don't store numpy str dtypes, so convert to object
   1240 if isinstance(dtype, np.dtype) and issubclass(values.dtype.type, str):

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\dtypes\cast.py:1181
, in astype_nansafe(arr, dtype, copy, skipna)
   1177     raise ValueError(msg)
   1179 if copy or is_object_dtype(arr.dtype) or is_object_dtype(dtype):
   1180     # Explicit copy, or required since NumPy can't view from / to object.
-> 1181     return arr.astype(dtype, copy=True)
   1183 return arr.astype(dtype, copy=copy)

ValueError: could not convert string to float: 'a'
```

3.2. to_numeric으로 변환

In [25]:

```
# ignore : 숫자로 변경할 수 없는 값이 있으면 작업하지 않음
pd.to_numeric(s2, errors = 'ignore')
```

Out[25]:

```
0    -1.0
1       2
2      -3
3       a
dtype: object
```

In [26]:

```
# coerce : 숫자로 변경할 수 없는 값이 있으면 NaN으로 변환
pd.to_numeric(s2, errors = 'coerce')
```

Out[26]:

```
0   -1.0
1    2.0
2   -3.0
3    NaN
dtype: float64
```

In [27]:

```
# raise : 숫자로 변경할 수 없는 값이 있으면 에러 발생 ( default )
pd.to_numeric(s2, errors = 'raise')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\_libs\lib.pyx:2315, in p
andas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string "a"

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
Cell In [27], line 2
      1 # raise : 숫자로 변경할 수 없는 값이 있으면 에러 발생 ( default )
----> 2 pd.to_numeric(s2, errors = 'raise')

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\core\tools\numeric.py:18
```

```
4, in to_numeric(arg, errors, downcast)
    182 coerce_numeric = errors not in ("ignore", "raise")
    183 try:
--> 184     values, _ = lib.maybe_convert_numeric(
    185         values, set(), coerce_numeric=coerce_numeric
    186     )
    187 except (ValueError, TypeError):
    188     if errors == "raise":

File c:\Users\com\anaconda3\envs\vscode\lib\site-packages\pandas\_libs\lib.pyx:2357, in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string "a" at position 3
```

4. 시계열 데이터로 변경

`pd.to_datetime(컬럼, errors = 에러처리옵션)`

In [28]:

```
df = pd.read_csv('./data/birth_die.csv')
df.head()
```

Out[28]:

	이름	주요경력	출생	사망
0	스티븐 호킹	이론 물리학자	1942-01-08	2018-03-14
1	마이클잭슨	가수	1958-08-29	2009-06-25
2	스티브잡스	CEO	1955-02-24	2011-10-05
3	로빈윌리엄스	배우	1951-07-21	2014-08-11
4	앨빈토플러	미래학자	1928-10-04	2016-06-27

In [29]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   이름      5 non-null      object
 1   주요경력    5 non-null      object
 2   출생      5 non-null      object
 3   사망      5 non-null      object
dtypes: object(4)
memory usage: 288.0+ bytes
```

4.1. astype으로 변환

In [34]:

```
df['출생'] = df['출생'].astype('datetime64')
df.dtypes
```

Out[34]:

```
이름              object
주요경력            object
출생      datetime64[ns]
사망              object
dtype: object
```

4.2. to_datetime으로 변환

In [35]:

```
# 사망
df['사망'] = pd.to_datetime(df['사망'])
df.dtypes
```

Out[35]:

```
이름              object
주요경력            object
출생      datetime64[ns]
사망      datetime64[ns]
dtype: object
```