01_판다스 시작하기

• 데이터 집합 불러오기

> read_csv('경로/파일명', sep='₩t', encoding='utf-8')

```
import pandas as pd
df = pd.read_csv('data/gapminder.tsv',sep='₩t')
```

※ 판다스에서 사용하는 자료형 :

시리즈(Series)

```
0    25
1     5
2     5
3    15
4     2
dtype: int64
```

```
서울    25
대전     5
광주     5
부산    15
제주     2
dtype: int64
```

데이터프레임(DataFrame)

	구	국번
서울	25	02
대전	5	042
광주	5	062
부산	15	051
제주	2	064

• 데이터 집합 불러오기

> head(n)
  - 데이터프레임의 데이터를 확인하는 용도로 사용
  - 가장 앞에 있는 5개의 행을 출력 (n 숫자 지정 시 개수만큼 출력)

```
df.head()
```

Out :

	country	continent	year	lifeExp	pop	gdpPercap
0	Afghanistan	Asia	1952	28.801	8425333	779.445314
1	Afghanistan	Asia	1957	30.332	9240934	820.853030
2	Afghanistan	Asia	1962	31.997	10267083	853.100710
3	Afghanistan	Asia	1967	34.020	11537966	836.197138
4	Afghanistan	Asia	1972	36.088	13079460	739.981106

• 데이터 집합 불러오기

> type() – 자료형 확인

```
type(df)
```

Out : pandas.core.frame.DataFrame

> shape – 데이터의 행과 열의 크기 정보

```
df.shape
```

Out : (1704, 6)

> columns – 데이터프레임의 열 이름 정보

```
df.columns
```

Out : Index(['country', 'continent', 'year', 'lifeExp', 'pop',
'gdpPercap'], dtype='object')

• 데이터 집합 불러오기

> dtypes – 데이터프레임의 모든 열 자료형 확인

```
df.dtypes
```

Out :

```
country       object
continent     object
year           int64
lifeExp      float64
pop            int64
gdpPercap    float64
dtype: object
```

> info() – 데이터프레임의 상세 정보 확인

```
df.info()
```

Out :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1704 entries, 0 to 1703
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   country    1704 non-null   object
 1   continent  1704 non-null   object
 2   year       1704 non-null   int64
 3   lifeExp    1704 non-null   float64
 4   pop        1704 non-null   int64
 5   gdpPercap  1704 non-null   float64
dtypes: float64(2), int64(2), object(2)
memory usage: 80.0+ KB
None
```

• 데이터 집합 불러오기

> 판다스와 파이썬 자료형 비교

판다스 자료형	파이썬 자료형	설명
object	string	문자열
int64	int	정수
float64	float	소수점을 가진 숫자
datetime64	datetime	파이썬 표준 라이브러리인 datetime이 반환하는 자료형

• 데이터 추출하기

> df['열 이름']

```
country_df = df['country']
type(country_df)
```

Out : pandas.core.series.Series

> tail(n)

- 데이터프레임의 데이터를 확인하는 용도로 사용
- 가장 뒤에 있는 5개의 행을 출력 (n 숫자 지정 시 개수만큼 출력)

```
country_df.tail()
```

Out :
```
1699    Zimbabwe
1700    Zimbabwe
1701    Zimbabwe
1702    Zimbabwe
1703    Zimbabwe
Name: country, dtype: object
```

• 데이터 추출하기

> 여러 개의 열 데이터 추출하기

```
subset=df[['country', 'continent', 'year']]
type(subset)
```

Out : pandas.core.frame.DataFrame

```
subset.tail()
```

Out :

	country	continent	year
1699	Zimbabwe	Africa	1987
1700	Zimbabwe	Africa	1992
1701	Zimbabwe	Africa	1997
1702	Zimbabwe	Africa	2002
1703	Zimbabwe	Africa	2007

• 데이터 추출하기

> 행 단위 데이터 추출하기
  - loc : 인덱스를 기준으로 행 데이터 추출
  - iloc : 행 번호를 기준으로 행 데이터 추출

	country	continent	year	lifeExp	pop	gdpPercap
0	Afghanistan	Asia	1952	28.801	8425333	779.445314
1	Afghanistan	Asia	1957	30.332	9240934	820.853030
2	Afghanistan	Asia	1962	31.997	10267083	853.100710
3	Afghanistan	Asia	1967	34.020	11537966	836.197138
4	Afghanistan	Asia	1972	36.088	13079460	739.981106

인덱스

> 인덱스는 보통 0부터 시작하지만, 특정 열을 지정하여 사용 가능
  행 번호는 0부터 시작하는 정수형태의 데이터 순서

• 데이터 추출하기

> loc 속성으로 행 데이터 추출하기

```
df.loc[0]
```

Out :

```
country      Afghanistan
continent           Asia
year                1952
lifeExp           28.801
pop              8425333
gdpPercap        779.445
Name: 0, dtype: object
```

```
df.loc[99]
```

Out :

```
country      Bangladesh
continent           Asia
year                1967
lifeExp           43.453
pop             62821884
gdpPercap        721.186
Name: 99, dtype: object
```

```
df.loc[-1]
```

```
-------
KeyError                                  Traceback (most recent cal
l last)
```

• 데이터 추출하기

> 마지막 행 데이터 추출하기

```
number_of_rows = df.shape[0]
last_row_index = number_of_rows - 1
df.loc[last_row_index]
```

Out :

```
country      Zimbabwe
continent      Africa
year             2007
lifeExp        43.487
pop          12311143
gdpPercap     469.709
Name: 1703, dtype: object
```

```
df.tail(1)
```

Out :

	country	continent	year	lifeExp	pop	gdpPercap
1703	Zimbabwe	Africa	2007	43.487	12311143	469.709298

• 데이터 추출하기

> 여러 행 데이터 추출하기

```
df.loc[[0, 99, 999]]
```

Out :

	country	continent	year	lifeExp	pop	gdpPercap
0	Afghanistan	Asia	1952	28.801	8425333	779.445314
99	Bangladesh	Asia	1967	43.453	62821884	721.186086
999	Mongolia	Asia	1967	51.253	1149500	1226.041130

```
df.loc[0:2]
```

Out :

	country	continent	year	lifeExp	pop	gdpPercap
0	Afghanistan	Asia	1952	28.801	8425333	779.445314
1	Afghanistan	Asia	1957	30.332	9240934	820.853030
2	Afghanistan	Asia	1962	31.997	10267083	853.100710

• 데이터 추출하기

> iloc 속성으로 행 데이터 추출하기

```
df.iloc[0]
```

Out :

```
country      Afghanistan
continent           Asia
year                1952
lifeExp           28.801
pop              8425333
gdpPercap        779.445
Name: 0, dtype: object
```

```
df.iloc[99]
```

Out :

```
country      Bangladesh
continent           Asia
year                1967
lifeExp           43.453
pop             62821884
gdpPercap        721.186
Name: 99, dtype: object
```

• 데이터 추출하기

> iloc 속성으로 행 데이터 추출하기

```
df.iloc[-1]
```

Out :

```
country      Zimbabwe
continent      Africa
year             2007
lifeExp        43.487
pop          12311143
gdpPercap     469.709
Name: 1703, dtype: object
```

```
df.iloc[1710]
```

```
---------------------------------------------------------------------
-------
IndexError                                Traceback (most recent cal
l last)
<ipython-input-28-e91d411323c7> in <module>()
----> 1 print(df.iloc[1710])
```

• 데이터 추출하기

> loc, iloc 속성 자유자재로 사용하기
  - 슬라이싱 구문으로 데이터 추출하기

```
subset = df.loc[:, ['year', 'pop']]
subset.head()
```

Out :

	year	pop
0	1952	8425333
1	1957	9240934
2	1962	10267083
3	1967	11537966
4	1972	13079460

• 데이터 추출하기

> loc, iloc 속성 자유자재로 사용하기
  - 슬라이싱 구문으로 데이터 추출하기

```
subset = df.iloc[:, [2, 4, -1]]
subset.head()
```

Out :

	year	pop	gdpPercap
0	1952	8425333	779.445314
1	1957	9240934	820.853030
2	1962	10267083	853.100710
3	1967	11537966	836.197138
4	1972	13079460	739.981106

• 데이터 추출하기

> loc, iloc 속성 자유자재로 사용하기
  - 슬라이싱 구문으로 데이터 추출하기

```
subset = df.iloc[:, range(5)]
subset.head()
```

Out :

	country	continent	year	lifeExp	pop
0	Afghanistan	Asia	1952	28.801	8425333
1	Afghanistan	Asia	1957	30.332	9240934
2	Afghanistan	Asia	1962	31.997	10267083
3	Afghanistan	Asia	1967	34.020	11537966
4	Afghanistan	Asia	1972	36.088	13079460

• 데이터 추출하기

> loc, iloc 속성 자유자재로 사용하기
  - 여러 개의 행, 열 데이터 추출하기

```
df.loc[[0, 99, 999], ['country', 'lifeExp', 'gdpPercap']]
```

Out :

	country	lifeExp	gdpPercap
0	Afghanistan	28.801	779.445314
99	Bangladesh	43.453	721.186086
999	Mongolia	51.253	1226.041130

```
df.iloc[[0, 99, 999], [0, 3, 5]]
```

Out :

	country	lifeExp	gdpPercap
0	Afghanistan	28.801	779.445314
99	Bangladesh	43.453	721.186086
999	Mongolia	51.253	1226.041130

• 기초적인 통계 계산하기

> **그룹화한 데이터의 평균 구하기**
  - groupby('그룹화 열 이름')['대상 열'].mean()

```
df.groupby('year')['lifeExp'].mean()
```

Out :

```
year
1952    49.057620
1957    51.507401
1962    53.609249
1967    55.678290
1972    57.647386
1977    59.570157
1982    61.533197
1987    63.212613
1992    64.160338
1997    65.014676
2002    65.694923
2007    67.007423
Name: lifeExp, dtype: float64
```

• 기초적인 통계 계산하기

> 그룹화한 데이터의 평균 구하기
  - groupby() 메소드 살펴보기

```
grouped_year_df = df.groupby('year')
type(grouped_year_df)
```

Out : pandas.core.groupby.generic.DataFrameGroupBy

```
grouped_year_df_lifeExp = grouped_year_df['lifeExp']
type(grouped_year_df_lifeExp)
```

Out : pandas.core.groupby.generic.SeriesGroupBy

• 기초적인 통계 계산하기

> 그룹화한 데이터의 평균 구하기
  - groupby() 메소드 살펴보기

```
mean_lifeExp_by_year = grouped_year_df_lifeExp.mean()
mean_lifeExp_by_year.head(2)
```

Out :

```
year
1952    49.057620
1957    51.507401
Name: lifeExp, dtype: float64
```

```
type(mean_lifeExp_by_year)
```

Out : pandas.core.series.Series

• 기초적인 통계 계산하기

> 그룹화한 데이터의 평균 구하기
  - lifeExp, gdpPercap 열의 평균값을 연도, 지역별로 그룹화하여 계산

```
df.groupby(['year', 'continent'])[['lifeExp', 'gdpPercap']].mean()
```

Out :

		lifeExp	gdpPercap
year	continent		
1952	Africa	39.135500	1252.572466
	Americas	53.279840	4079.062552
	Asia	46.314394	5195.484004
	Europe	64.408500	5661.057435
	Oceania	69.255000	10298.085650
1957	Africa	41.266346	1385.236062
	Americas	55.960280	4616.043733
	Asia	49.318544	5787.732940
	Europe	66.703067	6963.012816
	Oceania	70.295000	11598.522455

• 기초적인 통계 계산하기

> **그룹화한 데이터의 개수 세기**
  - nunique() : 빈도수
  - continent를 기준으로 그룹화한 후 country 개수 세기

```
df.groupby('continent')['country'].nunique()
```

Out :

```
continent
Africa      52
Americas    25
Asia        33
Europe      30
Oceania      2
Name: country, dtype: int64
```

• 그래프 그리기

> **년도별 수명 데이터를 그룹화하여 그래프로 표현**
  - year 열을 기준으로 그룹화한 데이터프레임에서 lifeExp 열의 평균 계산

```
year_life_expectancy = df.groupby('year')['lifeExp'].mean()
year_life_expectancy
```

Out :

```
year
1952    49.057620
1957    51.507401
1962    53.609249
1967    55.678290
1972    57.647386
1977    59.570157
1982    61.533197
1987    63.212613
1992    64.160338
1997    65.014676
2002    65.694923
2007    67.007423
Name: lifeExp, dtype: float64
```

• 그래프 그리기

> **년도별 수명 데이터를 그룹화하여 그래프로 표현**
  - matplotlib 라이브러리 사용

```
%matplotlib inline
import matplotlib.pyplot as plt
year_life_expectancy.plot()
```

Out : `<AxesSubplot:xlabel='year'>`