Python의 Pandas를 이용한 데이터 처리 실습

경남대학교 신병주 교수

동남권 LINC 3.0 사업단 부산광역시 BUSAN METROPOLITAN CITY NRF 한국연구재단


파이썬 자료구조
-리스트(List)-

파이썬 자료구조

◆ 리스트

- ❖ 변수는 하나의 값 혹은 다른 변수의 값을 가질 수 있음
- ❖ 여러 개의 값이나 변수를 가지기 위해서는 집합으로 지정
- ❖ 집합의 종류에는 리스트(List), 딕셔너리(Dictionary), 튜플(Tuple) 등이 있음
- ❖ 리스트
  - ➢ 여러 개의 변수, 값을 묶어서 집합으로 관리하는 자료 형태
  - ➢ 형식: 리스트명 = [원소1, 원소2, 원소3, ...]
  - ➢ 리스트로 정의된 변수의 이름을 사용하면 리스트 내 모든 원소 지정 가능

```
In [7]: # 예제 1-7 리스트 자료 입력
        print( '1번','2번','3번','4번','5번','6번','7번','8번','9번','10번')
        class_2_1= [ '1번','2번','3번','4번','5번','6번','7번','8번','9번','10번' ]
        print(class_2_1)

        1번 2번 3번 4번 5번 6번 7번 8번 9번 10번
        ['1번', '2번', '3번', '4번', '5번', '6번', '7번', '8번', '9번', '10번']
```

파이썬 자료구조

◆ 리스트

- ❖ 리스트 내에서 하나의 원소를 선택할 때는 인덱스를 이용하여 몇 번째 원소인지를 지정해서 선택 가능
- ❖ 형식: 리스트변수이름[인덱스 번호]
- ❖ 리스트의 첫 번째 원소에 해당하는 인덱스 번호는 0이며 순서대로 1씩 증가
- ❖ 리스트의 마지막 원소를 기준으로 인덱스 번호 표현 가능(마지막 원소가 -1이며, 역순으로 -1씩 증가)

```
In [8]: # 예제 1-8 리스트 자료 내 원소 1개 선택하기: 인덱스 번호 활용
        k    =   ['a','b','c','d','e']
        #index    0   1   2   3   4
        #index   -5  -4  -3  -2  -1

        print( k[1] )

        b
```

파이썬 자료구조

◆ 리스트

❖ 슬라이싱(Slicing)

- ➢ 리스트에서 여러 원소를 구간으로 선택할 경우 활용
- ➢ 형식: 리스트변수이름[시작 인덱스 번호 : 마지막 인덱스 번호]
- ➢ 주의: 콜론(:) 뒷부분에 적히는 인덱스 번호는 포함하지 않고 바로 앞의 인덱스 번호까지만 포함
- ➢ 인덱스 번호는 왼쪽부터 셀 수 있고, 오른쪽부터도 셀 수 있음

```
In [9]: # 예제 1-9 리스트 자료 구간별 글자 선택하기: 슬라이싱
        # list [인덱스시작번호: 끝 다음 번호]
        k     = ['a', 'b','c','d','e']
        #index      0   1   2   3   4
        #index     -5  -4   -3  -2  -1
        print( k[ 1 : 3 ] )
        print( k[ -4 : -2 ] )

        ['b', 'c']
        ['b', 'c']
```

리스트 k의 인덱스 번호 3번 직전(인덱스 번호 2)까지만 선택해서 화면 출력

리스트 k의 인덱스 번호 -4(오른쪽 끝에서부터 4번째 자리의 원소)부터 인덱스 번호 -2 직전(인덱스 번호 -3)까지만 선택해서 화면 출력

파이썬 자료구조

◆ 리스트

❖ 병합(Merge)

- ➢ 두 개 이상의 리스트를 합쳐서 하나의 리스트로 만들 수 있음
- ➢ 방법 1
  - ✓ 형식: 리스트A + 리스트B
  - ✓ 리스트A의 원소 이후에 리스트B의 원소가 차례대로 나열된 리스트 생성
  - ✓ 리스트 순서에 따라 결과가 달라짐

```
In [10]:
# 예제 1-10 리스트 합치기: 리스트1 + 리스트2
# A + B
l1 = ['a']
l2 = ['b','c','e']
print(l1 + l2)
print(l2 + l1)

['a', 'b', 'c', 'e']
['b', 'c', 'e', 'a']
```

- ➢ 방법 2
  - ✓ 형식: 리스트A.append(리스트B)
  - ✓ 리스트A의 우측 끝 원소 자리에 리스트B 전체를 마지막 원소로 추가
  - ✓ 리스트A 내에 리스트B가 하나의 원소로 들어가게 되며, 리스트A의 오른쪽 끝 원소인 리스트A[-1]은 리스트B가 됨

```
In [11]:
# 예제 1-11 리스트 추가하기(하나의 원소로 붙이기): 리스트1.append(리스트2)
l1 = ['a']
l2 = ['b','c','e']
l1.append(l2)
print(l1)

['a', ['b', 'c', 'e']]
```

print(l1[-1])
=> [‘b’, ‘c’, ‘e’]

파이썬 자료구조

◆ 반복문

❖ 동일한 코드를 여러 번 반복해서 실행

```
In [12]:

# 예제 1-12 반복문 활용하기
fruits = ['바나나','사과','딸기','배','감']
for fruit in fruits:
    print(fruit)
```

```
바나나
사과
딸기
배
감
```

fruits 리스트에서 원소를 순서대로 한나씩 꺼내어 fruit 변수에 저장

반복문을 사용할 때는 들여쓰기(탭 또는 공백 4칸으로 띄어스기)를 사용해야 하며, 들여쓰기가 사용되지 않은 곳부터는 반복이 일어나지 않음

파이썬 자료구조

◆ 문자열

❖ 문자열 인덱스, 슬라이싱

- ➢ 문자열 중 일부 문자만 선택
- ➢ 리스트에서 사용했던 인덱스 번호와 슬라이싱을 문자열에서도 동일하게 사용
- ➢ 공백이나 문장 부호 등도 하나의 글자로 인식

```
In [17]:
```

```
# 예제 1-17 문자열 일부 선택하기: 인덱스, 슬라이싱
k       =   "가나다라마바"
# index      0 1 2 3 4 5
# index     -6-5-4-3-2-1
print(k[2])
print(k[-1])
print(k[2:5])
print(k[3:])
```

```
다
바
다라마
라마바
```

파이썬 자료구조

◆ 문자열

❖ 문자열 처리 함수

- ➢ 문자열 시작과 끝부분의 공백을 제거하거나 글자 변경, 나누기 등이 필요
- ➢ 웹 크롤링을 통해 데이터를 수집했거나 텍스트 분석을 진행할 경우에는 데이터 분석이 용이하도록 문자열 전처리 필요
- ➢ 문자열 처리 함수 활용

```
In [18]:
# 예제 1-18 문자열 정리하기: strip(), replace(a,b), split()
t1 = "   hp010-0000-0000   "
print(t1)
t2 = t1.strip()
print(t2)
t3 = t2.replace('hp', ' ')
print(t3)
t4 = t3.split('-')
print(t4)
```

```
   hp010-0000-0000
hp010-0000-0000
 010-0000-0000
[' 010', '0000', '0000']
```

t1 문자열의 시작과 끝부분에 있는 공백 문자(띄어쓰기, 줄 바꿈, 들여쓰기 등)을 모두 없앤 뒤 변수 t2에 저장

t2에 있는 'hp'라는 문자를 모두 ' '로 변경하여 변수 t3에 저장. ''를 사용하게 되면 'hp'를 제거하라는 의미와 동일

'-' 기준으로 문자열을 분할, 분할된 문자열은 리스트로 생성


Chapter. 2
Pandas 기초

Pandas 기초

◆ pandas란?

- ❖ 엑셀(Excel)의 스프레드시트(spreadsheet) 같이 행(row)과 컬럼(column)으로 구성된 테이블 형태의 데이터를 쉽게 다룰 수 있는 파이썬 라이브러리
- ❖ 데이터프레임(Dataframe): 데이터를 표의 형태로 처리할 수 있는 자료구조
- ❖ import pandas
  - ➢ 파이썬 라이브러리를 사용하기 위해서는 import문으로 라이브러리를 불러와야 함
  - ➢ 아나콘다는 데이터 분석에 활용되는 많은 라이브러리가 파이썬과 함께 설치함으로 pandas 역시 바로 사용 가능

```
In [1]:
# 예제 2-1 pandas 라이브러리 불러오기
import pandas as pd
```

- pandas 내부의 함수들을 파이썬에서 사용할 수 있음
- pandas.함수명( ) 형태로 사용
- as pd를 붙여서 pd라는 별명(alias)을 지정하면 pd.함수명( )으로 사용 가능

Pandas 기초

◆ 데이터 불러오기(read_excel)

❖ 예제 데이터(sample_1.xlsx)

	A	B	C	D
1	2019년 11월 입국객 수			
2	국적코드	성별	입국객수	전년동기
3	A01	남성	106320	85815
4	A01	여성	191436	125241
5	A31	남성	319	299
6	A31	여성	42	54
7	A18	남성	158912	124486
8	A18	여성	232943	163466
9	총 합계	689972		
10	전년동기	499361		

1행: 해당 데이터에 대한 제목으로 데이터 분석에는 불필요한 정보

2행: 4개의 컬럼명. 예제 데이터 분석에서는 국적코드, 성별, 입국객수 컬럼 사용 예정

3-8행, A-C열: 예제 데이터 분석에서 분석할 데이터

9-10행: 총합계와 전년동기라는 요약 정보로써 데이터 분석에는 불필요한 정보

❖ 예제 데이터 다운로드: github.com/shinbyungjoo/Python-Lecture

Pandas 기초

◆ 데이터 불러오기(read_excel)

엑셀 파일을 불러오는 함수

- 엑셀 파일의 경로(Path)
- './'는 상대경로를 의미하는데, 파이썬 코드가 있는 폴더의 위치를 의미

- 컬럼명이 있는 위치
- 파이썬에서는 시작 숫자가 1이 아닌 0이기 때문에 엑셀에서 2행을 의미

마지막 행으로부터 두 행을 생략

A컬럼부터 C컬럼까지 가져옴

- 처음부터 3번째 행까지 보여주는 함수
- 불러온 데이터의 행이 많을 경우 파이썬에서 원하는 형태로 잘 불러왔는지 확인하기 위한 용도로 주로 사용

```
In [2]:
# 예제 2-2 sample_1.xlsx 엑셀데이터 불러오기
sample_1 = pd.read_excel('./files/sample_1.xlsx',
                         header=1,
                         skipfooter=2,
                         usecols='A:C')
sample_1.head(3)
```

Out [2]:

	국적코드	성별	입국객수
0	A01	남성	106320
1	A01	여성	191436
2	A31	남성	319

데이터를 잘 불러왔는지 확인하기 위해 앞부분 뿐만 아니라 마지막 부분도 확인하기 위한 함수

```
In [3]:
# 예제 2-3 tail() 함수 활용
sample_1.tail(3)
```

Out [3]:

	국적코드	성별	입국객수
3	A31	여성	42
4	A18	남성	158912
5	A18	여성	232943

	A	B	C	D
1	2019년 11월 입국객 수			
2	국적코드	성별	입국객수	전년동기
3	A01	남성	106320	85815
4	A01	여성	191436	125241
5	A31	남성	319	299
6	A31	여성	42	54
7	A18	남성	158912	124486
8	A18	여성	232943	163466
9	총 합계		689972	
10	전년동기		499361	

Pandas 기초

◆ 데이터 불러오기

❖ 데이터 정보 확인: info() 함수는 데이터에 대한 요약 정보 제공

pandas의 DataFrame(데이터프레임)으로 데이터 구성

```
In [4]:

# 예제 2-4 sample_1 데이터 정보 살펴보기
sample_1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   국적코드    6 non-null      object
 1   성별      6 non-null      object
 2   입국객수    6 non-null      int64
dtypes: int64(1), object(2)
memory usage: 272.0+ bytes
```

0-5까지, 총 6개의 행으로 구성

국적코드 컬럼은 빈 칸(non-null)이 없이 6개의 행으로 구성된 문자열 데이터

입국객수 컬럼은 빈 칸이 없이 6개의 행으로 구성된 정수형 데이터

총 3개의 컬럼으로 구성

성별 컬럼은 빈 칸이 없이 6개의 행으로 구성된 문자열 데이터

데이터는 정수형 데이터 1개, 문자열 데이터 2개로 구성

데이터가 총 224바이트 메모리 사용

	A	B	C	D
1	2019년 11월 입국객 수			
2	국적코드	성별	입국객수	전년동기
3	A01	남성	106320	85815
4	A01	여성	191436	125241
5	A31	남성	319	299
6	A31	여성	42	54
7	A18	남성	158912	124486
8	A18	여성	232943	163466
9	총 합계	689972		
10	전년동기	499361		

Pandas 기초

◆ 데이터 불러오기

❖ 데이터 기초통계량 확인: describe() 함수는 숫자형 데이터에 대한 여러가지 통계량 출력

```
In [5]:
# 예제 2-5 sample1_ 데이터 기초통계량 확인
sample_1.describe()
```

Out [5]:

	입국객수
count	6.000000
mean	114995.333333
std	98105.752006
min	42.000000
25%	26819.250000
50%	132616.000000
75%	183305.000000
max	232943.000000

데이터 개수

평균값

표준편차

최소값

1사분위수

2사분위수(중위수)

3사분위수

최대값

	A	B	C	D
1	2019년 11월 입국객 수			
2	국적코드	성별	입국객수	전년동기
3	A01	남성	106320	85815
4	A01	여성	191436	125241
5	A31	남성	319	299
6	A31	여성	42	54
7	A18	남성	158912	124486
8	A18	여성	232943	163466
9	총 합계	689972		
10	전년동기	499361		

Pandas 기초

◆ 데이터 선택 – 컬럼 기준

❖ 데이터 확인

```
In [6]:
# 예제 2-6 sample_1 데이터 확인하기
sample_1
```

Out [6]:

	국적코드	성별	입국객수
0	A01	남성	106320
1	A01	여성	191436
2	A31	남성	319
3	A31	여성	42
4	A18	남성	158912
5	A18	여성	232943

❖ 한 개의 컬럼 선택

```
In [7]:
# 예제 2-7 한 개 컬럼 선택하기
sample_1['입국객수']
```

```
Out [7]:
0    106320
1    191436
2       319
3        42
4    158912
5    232943
Name: 입국객수, dtype: int64
```

	A	B	C	D
1	2019년 11월 입국객 수			
2	국적코드	성별	입국객수	전년동기
3	A01	남성	106320	85815
4	A01	여성	191436	125241
5	A31	남성	319	299
6	A31	여성	42	54
7	A18	남성	158912	124486
8	A18	여성	232943	163466
9	총 합계	689972		
10	전년동기	499361		

Pandas 기초

◆ 데이터 선택 – 컬럼 기준

❖ 여러 개의 컬럼 선택

```
In [8]:
# 예제 2-8 여러 컬럼 선택하기
sample_1[['국적코드', '입국객수']]
```

Out [8]:

	국적코드	입국객수
0	A01	106320
1	A01	191436
2	A31	319
3	A31	42
4	A18	158912
5	A18	232943

- 대괄호로 묶은 형태는 리스트를 의미함
- 여러 개의 컬럼을 선택하기 위해서는 여러 개의 컬럼을 리스트로 묶어서 입력

[['A01', 106320],
['A01', 191436],
['A31', 319],
['A31', 42],
['A18', 158912],
['A18', 232943]]

❖ 신규 컬럼 생성

```
In [9]:
# 예제 2-9 컬럼 생성하기
sample_1['기준년월'] = '2019-11'
sample_1
```

Out [9]:

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

데이터에 존재하지 않던 컬럼에 값을 부여하면 새로운 컬럼이 생성됨

	A	B	C	D
1	2019년 11월 입국객 수			
2	국적코드	성별	입국객수	전년동기
3	A01	남성	106320	85815
4	A01	여성	191436	125241
5	A31	남성	319	299
6	A31	여성	42	54
7	A18	남성	158912	124486
8	A18	여성	232943	163466
9	총 합계		689972	
10	전년동기		499361	

Pandas 기초

◆ 데이터 선택 – 행 기준(Filtering)

❖ 성별이 남성인 데이터 필터링

In [10]:

```
# 예제 2-10 필터링하기 1
condition = (sample_1['성별'] == '남성')
condition
```

Out[10]:

```
0     True
1    False
2     True
3    False
4     True
5    False
Name: 성별, dtype: bool
```

In [6]:

```
# 예제 2-6 sample_1 데이터 확인하기
sample_1
```

Out[6]:

	국적코드	성별	입국객수
0	A01	남성	106320
1	A01	여성	191436
2	A31	남성	319
3	A31	여성	42
4	A18	남성	158912
5	A18	여성	232943

❖ 결과

In [11]:

```
# 예제 2-11 필터링하기 2
sample_1[condition]
```

Out[11]:

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
2	A31	남성	319	2019-11
4	A18	남성	158912	2019-11

Pandas 기초

◆ 데이터 선택 – 행 기준(Filtering)

❖ 입국객수가 150,000명 이상인 데이터 필터링

```
In [12]:
# 예제 2-12 필터링하기 3
condition = (sample_1['입국객수'] >= 150000)
sample_1[condition]
```

Out[12]:

	국적코드	성별	입국객수	기준년월
1	A01	여성	191436	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

❖ 한 컬럼에 대해 여러 조건으로 데이터 필터링

```
In [15]:
# 예제 2-17 한 컬럼에 여러 조건 필터링하기 1
conditions = (sample_1['국적코드'] == 'A01') |
             (sample_1['국적코드'] == 'A18')
sample_1[conditions]
```

Out[15]:

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

❖ 성별이 남성이면서 입국객수가 150,000명 이상인 데이터 필터링

```
In [14]:
# 예제 2-14 두 개 컬럼에 필터링하기 1
conditions = (sample_1['성별'] == '남성') & (sample_1['입국객수'] >= 150000)
sample_1[conditions]
```

Out[14]:

	국적코드	성별	입국객수	기준년월
4	A18	남성	158912	2019-11

❖ 한 컬럼에 대해 여러 조건으로 데이터 필터링

```
In [16]:
# 예제 2-18 한 컬럼에 여러 조건 필터링하기 2
conditions = (sample_1['국적코드'].isin(['A01', 'A18']))
sample_1[conditions]
```

Out[16]:

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

Pandas 기초

◆ 데이터 통합 – 옆으로 통합(Merge)

In [17]:

```
# 예제 2-21 데이터 확인하기
sample_1
```

Out[17]:

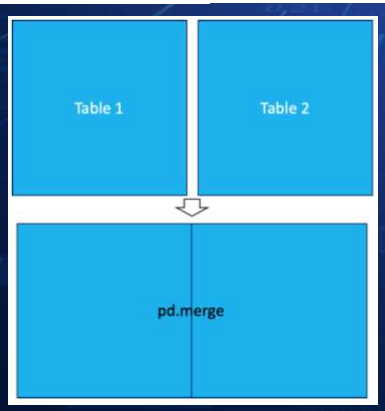
	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

In [18]:

```
# 예제 2-22 국적코드표 엑셀파일 불러오기
code_master = pd.read_excel('./files/sample_codemaster.xlsx')
code_master
```

Out[18]:

	국적코드	국적명
0	A01	일본
1	A02	대만
2	A03	홍콩
3	A18	중국
4	A19	이란
5	A22	우즈베키스탄
6	A23	카자흐스탄
7	A99	아시아 기타



Pandas 기초

◆ 데이터 통합 – 옆으로 통합(Merge)

기준 테이블과 정보를 결합하고 싶은 테이블 간에 공통된 컬럼에 대해 같은 값을 찾아서 결합해 하나의 테이블로 합치는 역할을 하는 함수

```
In [19]:
# 예제 2-23 데이터 옆으로 통합하기(left 조건)
sample_1_code = pd.merge(left=sample_1,
                         right=code_master,
                         how='left',
                         left_on='국적코드',
                         right_on='국적코드')
sample_1_code
```

왼쪽 테이블은 sample_1로 지정

오른쪽 테이블은 code_master로 지정

왼쪽 테이블을 기준으로 두 테이블 결합

왼쪽 테이블의 기준 컬럼

오른쪽 테이블의 기준 컬럼

Out [19]:

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국

In [17]:

```
# 예제 2-21 데이터 확인하기
sample_1
```

Out [17]:

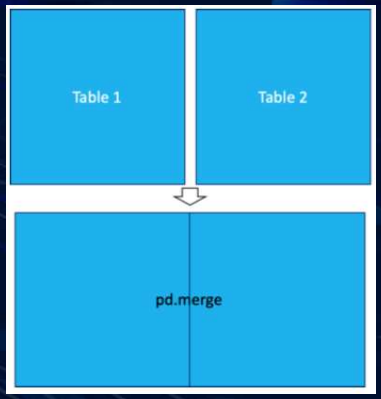
	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

In [18]:

```
# 예제 2-22 국적코드표 엑셀파일 불러오기
code_master = pd.read_excel('./files/sample_codemaster.xlsx')
code_master
```

Out [18]:

	국적코드	국적명
0	A01	일본
1	A02	대만
2	A03	홍콩
3	A18	중국
4	A19	이란
5	A22	우즈베키스탄
6	A23	카자흐스탄
7	A99	아시아 기타



Pandas 기초

◆ 데이터 통합 – 옆으로 통합(Merge)

```
In [20]:
# 예제 2-24 데이터 옆으로 통합하기(inner 조건)
sample_1_code_inner = pd.merge(left=sample_1,
                               right=code_master,
                               how='inner',
                               left_on='국적코드',
                               right_on='국적코드')
sample_1_code_inner
```

두 테이블의 기준 컬럼의 값이 서로 일치하는 경우에만 데이터 통합

Out [20]:

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A18	남성	158912	2019-11	중국
3	A18	여성	232943	2019-11	중국

In [17]:

```
# 예제 2-21 데이터 확인하기
sample_1
```

Out [17]:

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

In [18]:

```
# 예제 2-22 국적코드표 엑셀파일 불러오기
code_master = pd.read_excel('./files/sample_codemaster.xlsx')
code_master
```

Out [18]:

	국적코드	국적명
0	A01	일본
1	A02	대만
2	A03	홍콩
3	A18	중국
4	A19	이란
5	A22	우즈베키스탄
6	A23	카자흐스탄
7	A99	아시아 기타

Pandas 기초

◆ 데이터 통합 – 아래로 통합(Append)

In [24]:

```
# 예제 2-25 sample_1_code 확인하기
sample_1_code
```

Out [24]:

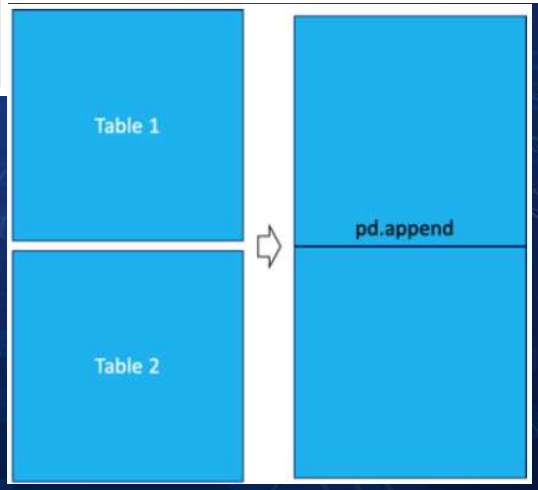
	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국

In [23]:

```
# 예제 2-26 sample_2_code 확인하기
sample_2_code
```

Out [23]:

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	92556	2019-12	일본
1	A01	여성	163737	2019-12	일본
2	A18	남성	155540	2019-12	중국
3	A18	여성	249023	2019-12	중국



Pandas 기초

◆ 데이터 통합 – 아래로 통합(Append)

- append() 함수를 사용해 sample_1_code의 칼럼을 기준으로 sample_2_code를 아래로 통합
- append() 함수를 사용하여 데이터 통합을 하기 위해서는 반드시 컬럼 순서가 동일해야 함

일반적으로 ignore_index = True 인자를 지정함

```
In [25]:
# 예제 2-27 데이터 아래로 통합하기 1
sample = sample_1_code.append(sample_2_code, ignore_index=True)
sample
```

Out [25]:

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국
6	A01	남성	92556	2019-12	일본
7	A01	여성	163737	2019-12	일본
8	A18	남성	155540	2019-12	중국
9	A18	여성	249023	2019-12	중국

```
In [24]:
# 예제 2-25 sample_1_code 확인하기
sample_1_code
```

Out [24]:

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국

```
In [23]:
# 예제 2-26 sample_2_code 확인하기
sample_2_code
```

Out [23]:

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	92556	2019-12	일본
1	A01	여성	163737	2019-12	일본
2	A18	남성	155540	2019-12	중국
3	A18	여성	249023	2019-12	중국

Pandas 기초

◆ 데이터 저장(to_excel)

```
In [26]:
# 예제 2-31 sample 데이터 엑셀파일로 저장하기 2
sample.to_excel('./files/sample_index_false.xlsx', index=False)
```

	A	B	C	D	E
1	국적코드	성별	입국객수	기준년월	국적명
2	A01	남성	106320	2019-11	일본
3	A01	여성	191436	2019-11	일본
4	A31	남성	319	2019-11	
5	A31	여성	42	2019-11	
6	A18	남성	158912	2019-11	중국
7	A18	여성	232943	2019-11	중국
8	A01	남성	92556	2019-12	일본
9	A01	여성	163737	2019-12	일본
10	A18	남성	155540	2019-12	중국
11	A18	여성	249023	2019-12	중국

Thank You


경남대학교
경상국립대학교
경성대학교
동명대학교
DSU
Dongseo University
동서대학교
동아대학교
동의대학교
부경대학교
부산대학교
신라대학교
울산대학교
인제대학교
창원대학교
한국해양대학교