데이터분석 동아리

Python 기초

경남대학교
데이터분석동아리
김동우

데이터분석 동아리

INDEX

가장 기본적인 조건문 if

If문은 조건문이라고도 하며 조건문은 참과 거짓을 판단하는 문장을 말한다.

In [26]:
```
if 조건 :
    조건이 맞는결과

else:
    조건에 안맞았을시 결과
```

조건문 if의 기본적인 형태

In [44]:
```
a = 2000


if a >1500:
    print('크다')

else:
    print('작다')
```

크다

비교 연산자

비교 연산자는 두 값을 비교할때 쓰는 연산자들을 의미한다

비교연산자	설명
x < y	x가 y보다 작다
x > y	x가 y보다 크다
x == y	x와 y가 같다
x != y	x와 y가 같지 않다
x >= y	x가 y보다 크거나 같다
x <= y	x가 y보다 작거나 같다

In [45]:
```
a = 2000


if a == 2000:
    print('같다')

else:
    print('다르다')
```

같다

In [46]:
```
a = 2000


if a != 1500:
    print('다르다')

else:
    print('같다')
```

다르다

비교 연산자

In [40]:
```
a = '데이터'

if a == '데이터':
    print('좋아')
else:
    print('싫어')
```

좋아

In [50]:
```
a = '데이터'

if a < '데이터':
    print('좋아')
else:
    print('싫어')
```

싫어

And,or,not 사용하기

연산자	설명
x or y	x와 y 둘중에 하나만 참이어도 참이다
x and y	x와 y 모두 참이어야 참이다
not x	x가 거짓이면 참이다

In [51]:
```
a = 2000
b = 1600


if a > 1500 and b <2000 :
    print('좋아')
else:
    print('시러요')
```

좋아

And,or,not 사용하기

In [52]:
```
a = 2000
b = 1600


if a > 1500 or b >2000 :
    print('좋아')
else:
    print('시러요')
```

좋아

In [53]:
```
a = 2000
b = 1600


if a < 1500 or b >2000 :
    print('좋아')
else:
    print('시러요')
```

시러요

In 사용하기

In [42]:
```
a = ['빅','데이터','분석','좋아요호홍']

if '도시재생'in a:
    print('좋아용')

else:
    print('싫어욧')
```

싫어욧

In [43]:
```
a = ['빅','데이터','분석','좋아요호홍']

if '도시재생'in a:
    print('좋아용')

elif '분석' in a:
    print("보통이에요")

else:
    print('싫어욧')
```

보통이에요

반복문 for 사용하기

For문은 다재다능하고 가장 많이 사용하는 것이 될것

```
In [ ]: for x in '선언할것':
            '1번작업'
            '2번작업'
```

```
In [2]: a = ['빅','데이터','분석','좋아요호홍']
        for i in a:
            print(i)
```

```
빅
데이터
분석
좋아요호홍
```

반복문 for 사용하기

For문은 다재다능하고 가장 많이 사용하는 것이 될것

In [3]:
```
a = [(1,2),(3,4),(5,6)]

for i,x in a:
    print(i + x)
```

```
3
7
11
```

In [4]:
```
a = [(1,2),(3,4),(5,6)]

for x in a:
    print( x)
```

```
(1, 2)
(3, 4)
(5, 6)
```

반복문 for 사용하기

For문은 다재다능하고 가장 많이 사용하는 것이 될것

```
# marks1.py
marks = [90, 25, 67, 45, 80]

number = 0
for mark in marks:
    number = number +1
    if mark >= 60:
        print("%d번 학생은 합격입니다." % number)
    else:
        print("%d번 학생은 불합격입니다." % number)
```

```
C:\doit>python marks1.py
1번 학생은 합격입니다.
2번 학생은 불합격입니다.
3번 학생은 합격입니다.
4번 학생은 불합격입니다.
5번 학생은 합격입니다.
```

반복문 for 에 자주 사용하는 함수 range 사용하기

```
In [5]: range(10)
Out[5]: range(0, 10)
```

```
In [6]: list(range(10))
Out[6]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [7]: for i in range(10):
            print(i)
0
1
2
3
4
5
6
7
8
9
```

반복문 for 에 자주 사용하는 함수 range 사용하기

```
>>> add = 0
>>> for i in range(1, 11):
...     add = add + i
...
>>> print(add)
55
```

```
#marks3.py
marks = [90, 25, 67, 45, 80]
for number in range(len(marks)):
    if marks[number] < 60:
        continue
    print("%d번 학생 축하합니다. 합격입니다." % (number+1))
```

문제1
Gapminder 를 불러온 다음 continent변수가 Aisa면 Yes라는 값을 가지고 아니면 No라는 값을 가지는 파생변수를 만들어라

```
In [38]: df
```

Out [38]:

	country	continent	year	lifeExp	pop	gdpPercap	Aisa
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	Yes
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	Yes
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	Yes
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	Yes
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	Yes
...	...	...	...	...	...	...	...
1699	Zimbabwe	Africa	1987	62.351	9216418	706.157306	No
1700	Zimbabwe	Africa	1992	60.377	10704340	693.420786	No
1701	Zimbabwe	Africa	1997	46.809	11404948	792.449960	No
1702	Zimbabwe	Africa	2002	39.989	11926563	672.038623	No
1703	Zimbabwe	Africa	2007	43.487	12311143	469.709298	No

1704 rows × 7 columns

In [37]:
```
from tqdm import tqdm
df['Aisa'] = 0

for i in tqdm(range(len(df))):
    if df['continent'][i] == 'Asia':
        df['Aisa'][i] = 'Yes'
    else:
        df['Aisa'][i] = 'No'
```

문제2
Gapminder 를 불러온 다음 lifeExp가 55이상이고 pop이 10500000 이상이면 '좋음'을 45이상 9500000 이상이면 '보통'을 둘 다 아니면 '나쁨'이라는 파생변수를 만들어라

```
In [43]: df
```

Out [43]:

	country	continent	year	lifeExp	pop	gdpPercap	Aisa	좋보싫
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	Yes	나쁨
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	Yes	나쁨
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	Yes	나쁨
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	Yes	나쁨
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	Yes	나쁨
...	...	...	...	...	...	...	...	...
1699	Zimbabwe	Africa	1987	62.351	9216418	706.157306	No	나쁨
1700	Zimbabwe	Africa	1992	60.377	10704340	693.420786	No	좋음
1701	Zimbabwe	Africa	1997	46.809	11404948	792.449960	No	보통
1702	Zimbabwe	Africa	2002	39.989	11926563	672.038623	No	나쁨
1703	Zimbabwe	Africa	2007	43.487	12311143	469.709298	No	나쁨

In [42]:

```
df['좋보싫'] = 0
for i in tqdm(range(len(df))):
    if (df['lifeExp'][i] >= 55) & (df['pop'][i] >= 10567083):
        df['좋보싫'][i] = '좋음'
    elif (df['lifeExp'][i] >= 45) & (df['pop'][i] >= 9500000):
        df['좋보싫'][i] = '보통'
    else:
        df['좋보싫'][i] = '나쁨'
```

함수 def란 무엇인가

함수란 입력값을 가지고 어떤 일을 수행한 다음에 그 결과를 내어 주는 작업을 말한다

```
def 함수명(매개변수):
    <수행할 문장1>
    <수행할 문장2>
    ...
```

함수의 기본 형태

```
In [46]: def add(a,b):
             result = a+b
             return result

In [47]: add(35,40)
Out[47]: 75
```

함수의 기본 형태 예시

함수 def 사용법

```
In [57]: def add():
             return 'Hello Data Analyst'

In [59]: add(13)
```

```
TypeError                                 Traceback (most recer
<ipython-input-59-6fcf8aefa56c> in <module>
----> 1 add(13)

TypeError: add() takes 0 positional arguments but 1 was given
```

```
In [60]: add()
Out[60]: 'Hello Data Analyst'

In [61]: print(add())
Hello Data Analyst
```

입력값이 없는 함수

```
In [48]: def add(a,b):
             result = a+b
             print(result)

In [50]: a = add(35,40)
75

In [51]: print(a)
None
```

결과값이 없는 함수

함수 def 사용법

```
In [62]: def add_many(*arge):
             result = 0
             for i in arge:
                 result = result +i
             return result

In [66]: add_many(1,2,3)
Out[66]: 6

In [67]: add_many(1,2,3,4,6,7,8,5,8)
Out[67]: 44
```

입력값을 여러 개 받는 함수

```
In [69]: def elel(elel,*args):
             if elel == '빼빼로':
                 result = 0
                 for i in args:
                     result = result +i
             elif elel == '포키':
                 result = 1
                 for i in args:
                     result = result *i
             return result

In [73]: elel('포키',1,2,3,4)
Out[73]: 24

In [74]: elel('빼빼로',1,2,3,4)
Out[74]: 10
```

여러조건의 함수

함수 def 결과값의 특징

```
In [75]: def add_and_mul(a,b):
             return a+b,a*b

In [76]: add_and_mul(5,6)
Out[76]: (11, 30)
```

```
In [77]: def add_and_mul(a,b):
             return a+b
             return a*b

In [78]: add_and_mul(5,6)
Out[78]: 11
```

함수의 결과값은 언제나 하나이다

Labbda 사용하기

Lambda는 일반적인 함수를 좀 짧게 쓴다고 생각하면 된다

```
In [79]: def add(a,b):
             return a+b

In [80]: add(2,3)
Out[80]: 5
```

기본적인 def 함수

```
In [82]: add = lambda x,y: x+y

In [83]: add(2,3)
Out[83]: 5
```

를 lambda 식으로 표현

Labbda 사용하기

Lambda는 일반적인 함수를 좀 짧게 쓴다고 생각하면 된다

```
In [96]: def add(a):
             if a == '빼빼로':
                 return '좋아 '
             elif a == '포키':
                 return '조금좋아 '
             else:
                 return '나가 '

In [98]: add('빼빼로')
Out[98]: '좋아 '

In [99]: add('포키')
Out[99]: '조금좋아 '

In [100]: add('가래떡')
Out[100]: '나가 '
```

기본적인 def 함수

```
In [103]: add = lambda a:'좋아' if a == '빼빼로' else ('조금 좋아' if a == '포키' else '나가')

In [104]: add('빼빼로')
Out[104]: '좋아'

In [105]: add('포키')
Out[105]: '조금 좋아'

In [106]: add('가래떡')
Out[106]: '나가'
```

를 lambda 식으로 표현

Apply와 map 사용하기

apply	반드시 Series 타입에 사용
map	반드시 2차원이상의 Series에 사용(데이터프레임)

```
In [139]: df['year']

Out[139]: 0       1952
          1       1957
          2       1962
          3       1967
          4       1972
                  ...
          1699    1987
          1700    1992
          1701    1997
          1702    2002
          1703    2007
          Name: year, Length: 1704, dtype: int64
```

Series

In [140]: df

Out[140]:

	country	continent	year	lifeExp	pop	gdpPercap	year_int
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	1952
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	1957
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	1962
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	1967
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	1972
...	...	...	...	...	...	...	...
1699	Zimbabwe	Africa	1987	62.351	9216418	706.157306	1987
1700	Zimbabwe	Africa	1992	60.377	10704340	693.420786	1992
1701	Zimbabwe	Africa	1997	46.809	11404948	792.449960	1997
1702	Zimbabwe	Africa	2002	39.989	11926563	672.038623	2002
1703	Zimbabwe	Africa	2007	43.487	12311143	469.709298	2007

1704 rows × 7 columns

DataFrame

Apply와 map 사용하기

```
In [144]: df['라이프이엑스피'] = df['lifeExp'].map(lambda x: '크다' if x>35 else '작다')
```

```
In [145]: df
```

Out[145]:

	country	continent	year	lifeExp	pop	gdpPercap	라이프이엑스피
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	작다
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	작다
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	작다
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	작다
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	크다
...	...	...	...	...	...	...	...
1699	Zimbabwe	Africa	1987	62.351	9216418	706.157306	크다
1700	Zimbabwe	Africa	1992	60.377	10704340	693.420786	크다
1701	Zimbabwe	Africa	1997	46.809	11404948	792.449960	크다
1702	Zimbabwe	Africa	2002	39.989	11926563	672.038623	크다
1703	Zimbabwe	Africa	2007	43.487	12311143	469.709298	크다

1704 rows × 7 columns

```
In [86]: df['라이프이엑스피_1'] = df['lifeExp'].apply(lambda x: '크다' if x>35 else '작다')
```

```
In [87]: df
```

Out[87]:

	country	continent	year	lifeExp	pop	gdpPercap	라이프이엑스피	라이프이엑스피_1
0	Afghanistan	Asia	0 1952\n1 1957\n2 1962\n3 ...	28.801	8425333	779.445314	작다	작다
1	Afghanistan	Asia	0 1952\n1 1957\n2 1962\n3 ...	30.332	9240934	820.853030	작다	작다
2	Afghanistan	Asia	0 1952\n1 1957\n2 1962\n3 ...	31.997	10267083	853.100710	작다	작다
3	Afghanistan	Asia	0 1952\n1 1957\n2 1962\n3 ...	34.020	11537966	836.197138	작다	작다
4	Afghanistan	Asia	0 1952\n1 1957\n2 1962\n3 ...	36.088	13079460	739.981106	크다	크다
...	...	...	...	...	...	...	...	...
1699	Zimbabwe	Africa	0 1952\n1 1957\n2 1962\n3 ...	62.351	9216418	706.157306	크다	크다
1700	Zimbabwe	Africa	0 1952\n1 1957\n2 1962\n3 ...	60.377	10704340	693.420786	크다	크다
1701	Zimbabwe	Africa	0 1952\n1 1957\n2 1962\n3 ...	46.809	11404948	792.449960	크다	크다
1702	Zimbabwe	Africa	0 1952\n1 1957\n2 1962\n3 ...	39.989	11926563	672.038623	크다	크다
1703	Zimbabwe	Africa	0 1952\n1 1957\n2 1962\n3 ...	43.487	12311143	469.709298	크다	크다

1704 rows × 8 columns

Apply와 map 사용하기

```
In [89]: df.map(lambda x: print(x))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-89-750bf71d023c> in <module>
----> 1 df.map(lambda x: print(x))

~\anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   5463             if self._info_axis._can_hold_identifiers_and_holds_name(name):
   5464                 return self[name]
-> 5465             return object.__getattribute__(self, name)
   5466
   5467     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute 'map'
```

```
In [88]: df.apply(lambda x: print(x))
```

```
0       Afghanistan
1       Afghanistan
2       Afghanistan
3       Afghanistan
4       Afghanistan
           ...
1699       Zimbabwe
1700       Zimbabwe
1701       Zimbabwe
1702       Zimbabwe
1703       Zimbabwe
Name: country, Length: 1704, dtype: object
0         Asia
1         Asia
2         Asia
3         Asia
4         Asia
         ...
1699    Africa
1700    Africa
```

```
In [ ]:
```

Apply와 map 사용하기

문제.1

Gapminder 데이터프레임에서 country컬럼의 앞 글자만 따서 '이니셜'이란 새로운 컬럼을 만들어서 거기에 넣어라

```
In [98]: df
```

Out[98]:

	country	continent	year	lifeExp	pop	gdpPercap	이니셜
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	A
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	A
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	A
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	A
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	A
...	...	...	...	...	...	...	...
1699	Zimbabwe	Africa	1987	62.351	9216418	706.157306	Z
1700	Zimbabwe	Africa	1992	60.377	10704340	693.420786	Z
1701	Zimbabwe	Africa	1997	46.809	11404948	792.449960	Z
1702	Zimbabwe	Africa	2002	39.989	11926563	672.038623	Z
1703	Zimbabwe	Africa	2007	43.487	12311143	469.709298	Z

1704 rows × 7 columns

Apply와 map 사용하기

문제.1

Gapminder 데이터프레임에서 country컬럼의 앞 글자만 따서 '이니셜'이란 새로운 컬럼을 만들어서 거기에 넣어라

```
In [98]: df
```

Out[98]:

	country	continent	year	lifeExp	pop	gdpPercap	이니셜
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	A
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	A

```
In [97]: df['이니셜'] = df['country'].apply(lambda x: x[0])
```

1700	Zimbabwe	Africa	1992	60.377	10704340	693.420786	Z
1701	Zimbabwe	Africa	1997	46.809	11404948	792.449960	Z
1702	Zimbabwe	Africa	2002	39.989	11926563	672.038623	Z
1703	Zimbabwe	Africa	2007	43.487	12311143	469.709298	Z

1704 rows × 7 columns

Apply와 map 사용하기

문제.2

Gapminder 데이터 프레임에서 lifeExp가 60을 넘으면 그 값을 지우고 'Death'를 집어 넣어라

```
In [102]: df
```

Out[102]:

	country	continent	year	lifeExp	pop	gdpPercap	이니셜
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	A
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	A
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	A
3	Afghanistan	Asia	1967	34.02	11537966	836.197138	A
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	A
...	...	...	...	...	...	...	...
1699	Zimbabwe	Africa	1987	Death	9216418	706.157306	Z
1700	Zimbabwe	Africa	1992	Death	10704340	693.420786	Z
1701	Zimbabwe	Africa	1997	46.809	11404948	792.449960	Z
1702	Zimbabwe	Africa	2002	39.989	11926563	672.038623	Z
1703	Zimbabwe	Africa	2007	43.487	12311143	469.709298	Z

1704 rows × 7 columns

Apply와 map 사용하기

문제.2

Gapminder 데이터 프레임에서 lifeExp가 60을 넘으면 그 값을 지우고 'Death'를 집어 넣어라

```
In [102]: df
```

Out[102]:

	country	continent	year	lifeExp	pop	gdpPercap	이니셜
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	A

```
In [101]: df['lifeExp'] = df['lifeExp'].apply(lambda x: 'Death' if x >60 else x)
```

...	...	...	...	...	...	...	...
1699	Zimbabwe	Africa	1987	Death	9216418	706.157306	Z
1700	Zimbabwe	Africa	1992	Death	10704340	693.420786	Z
1701	Zimbabwe	Africa	1997	46.809	11404948	792.449960	Z
1702	Zimbabwe	Africa	2002	39.989	11926563	672.038623	Z
1703	Zimbabwe	Africa	2007	43.487	12311143	469.709298	Z

1704 rows × 7 columns

Apply와 map 사용하기

문제.2

Gapminder 데이터 프레임에서 continent컬럼의 문자 길이가 4면 뒤에 _good을 붙히고 아니면 _bad를 붙혀라

```
In [108]: df
```

Out[108]:

	country	continent	year	lifeExp	pop	gdpPercap	이니셜
0	Afghanistan	Asia_good	1952	28.801	8425333	779.445314	A
1	Afghanistan	Asia_good	1957	30.332	9240934	820.853030	A
2	Afghanistan	Asia_good	1962	31.997	10267083	853.100710	A
3	Afghanistan	Asia_good	1967	34.02	11537966	836.197138	A
4	Afghanistan	Asia_good	1972	36.088	13079460	739.981106	A
...	...	...	...	...	...	...	...
1699	Zimbabwe	Africa_bad	1987	Death	9216418	706.157306	Z
1700	Zimbabwe	Africa_bad	1992	Death	10704340	693.420786	Z
1701	Zimbabwe	Africa_bad	1997	46.809	11404948	792.449960	Z
1702	Zimbabwe	Africa_bad	2002	39.989	11926563	672.038623	Z
1703	Zimbabwe	Africa_bad	2007	43.487	12311143	469.709298	Z

1704 rows × 7 columns

Apply와 map 사용하기

문제.2

Gapminder 데이터 프레임에서 continent컬럼의 문자 길이가 4면 뒤에 _good을 붙히고 아니면 _bad를 붙혀라

```
In [108]: df
```

Out[108]:

	country	continent	year	lifeExp	pop	gdpPercap	이니셜
0	Afghanistan	Asia_good	1952	28.801	8425333	779.445314	A

```
In [107]: df['continent'] = df['continent'].apply(lambda x: x + '_good' if len(x) == 4 else x+'_bad')
```

1701	Zimbabwe	Africa_bad	1997	46.809	11404948	792.449960	Z
1702	Zimbabwe	Africa_bad	2002	39.989	11926563	672.038623	Z
1703	Zimbabwe	Africa_bad	2007	43.487	12311143	469.709298	Z

1704 rows × 7 columns

감사합니다