lambda 사용 예제

정의: lambda: 1회용 간단한 기능의 함수를 만드는 것 (익명함수)

- 사용법:
λ 매 개 변 수 1, 매 개 변 수 2: 매 개 변 수 를 이 용 한 표 현 식

In [1]:
```
sum = lambda a, b : a + b # a + b의 결과를 리턴 (sum 변수가 받음)
```

In [78]:
```
print(sum(3, 7))
```

```
10
```

key 매개변수

- 여러 항목을 가지는 리스트를 정렬할 때 정렬 기준을 설정

- sorted(iterable, key=None, reverse=False)

In [18]:
```
students = [('홍길동', 3.9, 20160303), # 이름, 학점, 학번 순서
                       ('김철수', 3.0, 20160302),
                       ('최자영', 4.3, 20160301)]

# students 항목의 인덱스 [2]: 학번을 기준으로 오름 차순 정렬
sorted_students = sorted(students, key=lambda s : s[2])
print(sorted_students)
```

```
[('최자영', 4.3, 20160301), ('김철수', 3.0, 20160302), ('홍길동', 3.9, 20160303)]
```

In [19]:
```
# students 항목의 인덱스 [1]: 학점을 기준으로 내림 차순 정렬
sorted_scores = sorted(students, key=lambda s: s[1], reverse=True)
print(sorted_scores)
```

```
[('최자영', 4.3, 20160301), ('홍길동', 3.9, 20160303), ('김철수', 3.0, 20160302)]
```

DataFrame에 lambda 적용

- apply()함수와 같이 적용

In [20]:
```
import pandas as pd
import numpy as np

df = pd.DataFrame([[1, 2], [3, 4], [5, 6]], columns=['A', 'B'])
df
```

Out[20]:

	A	B
0	1	2

	A	B
1	3	4
2	5	6

DataFrame의 컬럼에 연산 일괄 적용:

- apply() 및 일반 함수 사용: apply(함수명)

In [21]:
```
def plus_one(x):
        x = x + 1
        return x
```

In [22]:
```
df['A'] = df['A'].apply(plus_one)
df['B'] = df['B'].apply(plus_one)
df
```

Out[22]:

	A	B
0	2	3
1	4	5
2	6	7

DataFrame.apply(함수)

- DataFrame 전체에 함수를 일괄 적용

In [23]:
```
# DataFrame 전체에 plus_one() 함수 일괄 적용
df = df.apply(plus_one)
df
```

Out[23]:

	A	B
0	3	4
1	5	6
2	7	8

DataFrame의 컬럼에 연산 일괄 적용

- apply() 및 lambda 함수 사용: apply(lambda)

- λ 입 력 변 수 : 리 턴 값 형식

- 복잡한 연산의 경우, 별도의 함수를 작성

In [24]:
```
df['A'] = df['A'].apply(lambda x : x + 1)
df
```

Out[24]:

	A	B
0	4	4
1	6	6
2	8	8

In [25]:
```
df = df.apply(lambda x : x + 1)
df
```

Out[25]:

	A	B
0	5	5
1	7	7
2	9	9

기존 DataFrame에 새로운 컬럼 ['C'] 추가하기

- 특정 컬럼들에 apply(lambda 함수) 적용

In [87]:
```
#df['C'] = pd.Series([10, 20, 30])
df['C'] = [10, 20, 30] # 새로운 컬럼 추가
df
```

Out[87]:

	A	B	C
0	50	5	10
1	70	7	20
2	90	9	30

In [88]:
```
df[['A', 'C']] = df[['A', 'C']].apply(lambda x : x * 10)
df
```

Out[88]:

	A	B	C
0	500	5	100
1	700	7	200
2	900	9	300

공공데이터 강의 내용에 적용한 예제

In [89]:
```
##표준행정구역이름으로수정: 경기->경기도,경남->경상남도,...
addr_aliases = {'경기':'경기도', '경남':'경상남도', '경북':'경상북도', '충북':'충청북도', ''
                                '서울시':'서울특별시', '부산특별시':'부산광역시', '대전시':'대전
                                '부산시':'부산광역시',  '충남':'충청남도', '전남':'전라남도', '
# dict.get(key[,default])
```

```
# - key에 해당하는 값이 없으면 default값을 리턴
#addr['시도']= addr['시도'].apply(lambda v: addr_aliases.get(v, v))
```

In [90]:
```
addr_df = pd.DataFrame(['경기', '대전광역시', '경남', '경북', '충북', '충남', '전북', '전남', '
                                                columns=['시도'])
```

In [91]:
```
addr_df
```

Out[91]:

	시도
0	경기
1	대전광역시
2	경남
3	경북
4	충북
5	충남
6	전북
7	전남
8	경상북도

dict.get(key, default)

- key에 해당하는 값이 없으면 default값을 리턴

In [82]:
```
# addr_df['시도']에서 가져오는 값은 단 한개 밖에 없음
# addr_df['시도']의 값이 '경기'이면,
# key='경기'를 사용하여, addr_aliases.get('경기', '경기')를 검색함
# 만약 addr_aliases 딕셔너리에 '경기'란 key가 없으면, None을 리턴하지 않고 두 번째 파라미터인 '
addr_df['시도']= addr_df['시도'].apply(lambda v: addr_aliases.get(v, v))
addr_df
```

Out[82]:

	시도
0	경기도
1	대전광역시
2	경상남도
3	경상북도
4	충청북도
5	충청남도
6	전라북도
7	전라남도
8	경상북도

lambda v : addr_aliases.get(v, v) 를 일반함수(get_dict_value(key))로 구현

In [95]:
```
def get_dict_value(key):
        if not addr_aliases.get(key):
                #print('key:{}에 해당되는 값이 없어서 {}를 반환함'.format(key, key))
                return key
        else:
                value = addr_aliases.get(key)
                #print('key:{}, value:{}'.format(key, value))
                return value

print(get_dict_value('대구'))
```

대구

DataFrame에 apply(함수)적용 예제

apply(get_dict_value) 적용

- addr_df1['시도']에서 한 라인씩 데이터가 get_dict_value()함수에 전달됨

In [96]:
```
addr_df1 = pd.DataFrame(['경기', '대전광역시', '경남', '경북', '충북', '충남', '전북', '전남',
                                        columns=['시도'])
addr_df1['시도']= addr_df1['시도'].apply(get_dict_value)
addr_df1
```

Out[96]:

	시도
0	경기도
1	대전광역시
2	경상남도
3	경상북도
4	충청북도
5	충청남도
6	전라북도
7	전라남도
8	경상북도

In [83]:
```
print(addr_aliases.get('경기'))
print(addr_aliases.get('대전')) # None을 리턴
print(addr_aliases.get('부산', '부산광역시'))
```

경기도
None
부산광역시

간단한 딕셔너리에 lambda 적용

- map(function, iterable)함수: 리스트의 요소를 지정된 lambda 함수로 처리함

In [67]:
```
d = {'a': 1, 'b': 2}
# map의 첫 번째 파라미터 function은 lambda 함수로 대체
values = map(lambda key: d[key], d.keys())
print(list(values))
```

```
[1, 2]
```

In [16]: