201921540 신수량

파이썬 스타일 코드1 - 연습해보기

일반문제

Css Selector 수정

Css Selector는 웹 페이지에서 특정 요소를 선택하기 위해 해당 요소까지 찾아갈 수 있도록 해주는 주소와 같은 것이다. 대부분의 웹브라우져에서 는 해당 요소에 대한 css selector 값을 쉽게 얻어올 수 있다 (F12 > Select an Element). 다음과 같은 selector가 있다고 한다.

```
In [2]: #today_main_news > div.hdline_news > ul > li:nth-child(1)
```

이러한 selector를 웹크롤링에서 사용하기 위해서는 :nth-child라는 부분을 제거하는 작업이 필요한데, 이를 자동화 해보자.

> **Q: 해당 Selector를 문자열로 표시하고, split과 join 함수를 활용하여 다음 예시와 같은 selector를 출력하시오.**

출력결과 예시

```
In [24]: selector = "#today_main_news > div.hdline_news > ul > li:nth-child
         (1)"
         ## CODE
         ## '#today_main_news  >  div.hdline_news  >  ul  >  li'
```

HINT

1. 특정 구분자(seperator)를 통해 구분된 리스트를 만든다.

2. 구분된 리스트에서 해당 부분을 선택하고 1과는 다른 특정 구분자로 나눠준다.

3. 2의 리스트에서 필요한 부분만 선택하여 기존 리스트에 할당한다.

4. 구분자를 기준으로 리스트를 문자열로 합쳐준다.

In [6]:
```
selector = "#today_main_news > div.hdline_news > ul > li:nth-child
(1)"
selector_list = selector.split(">")
# >로 구분
selector_list[-1] = selector_list[-1].split(":")[0]
# li를 저장

" > ".join(selector_list)
# > 기준으로 원래 있는거랑 합친다.
```

Out[6]:
```
'#today_main_news  >  div.hdline_news  >  ul  >  li'
```

2. list comprehension으로 만드는 구구단

PR5 문제 3번에서 만들었던 구구단 계산기를 list comprehension으로 구현해보고자 한다.

Q: list comprehension을 사용하여 구구단을 연산하는 함수 gugu_com을 작성하고 구구단 7단을 출력하시오.

출력결과 예시

In [9]:
```
# gugu_com(x=2)

# 2  x  1 = 2
# 2  x  2 = 4
# 2  x  3 = 6
# 2  x  4 = 8
# 2  x  5 = 10
# 2  x  6 = 12
# 2  x  7 = 14
# 2  x  8 = 16
# 2  x  9 = 18
```

In [16]:
```
def gugu_com(x):
    [print(f"{x} x {i} = {x*i}") for i in range(1, 10)]
# list comprehension으로 구구단 구현
gugu_com(7)
# x자리에 7대입
```

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
```

3. 두 주사위의 곱

두 주사위의 곱은 다음과 같은 결과를 가진다.

> Q : list comprehension을 사용하여, 힌트를 제외하고는 한줄의 코드로 해당 결과를 가지는 이차원 리스트를 만드시오.
>
> ### 출력결과 예시

In [41]:
```
## CODE

## [[1, 2, 3, 4, 5, 6],
##  [2, 4, 6, 8, 10, 12],
##  [3, 6, 9, 12, 15, 18],
##  [4, 8, 12, 16, 20, 24],
##  [5, 10, 15, 20, 25, 30],
##  [6, 12, 18, 24, 30, 36]]
```

HINT

1. 한개의 주사위는 다음과 같이 표현할 수 있습니다.

die = [i for i in range(1,7)]

In [43]:
```
die = [i for i in range(1,7)]
# FOR문을 오한 반복문

[[j*i for i in die] for j in die]
# 이중반복문
```

Out[43]:
```
[[1, 2, 3, 4, 5, 6],
 [2, 4, 6, 8, 10, 12],
 [3, 6, 9, 12, 15, 18],
 [4, 8, 12, 16, 20, 24],
 [5, 10, 15, 20, 25, 30],
 [6, 12, 18, 24, 30, 36]]
```

4. 두 주사위의 합

간단한 테이블 형태의 데이터를 2차원 리스트로 표현해보자. 2개의 주사위를 굴리면 다음 표와 같이 36가지의 결과가 나온다.

> **Q: 이것을 6 x 6 크기의 2차원 리스트로 생성하고, 인덱싱을 통해 2 + 6의 값을 2가지 방법으로 나타내시오. (2차원 리스트 생성시 방법의 제한은 없습니다.)**

In [45]:
```
dice_sum = [[2, 3, 4, 5, 6, 7],
            [3, 4, 5, 6, 7, 8],
            [4, 5, 6, 7, 8, 9],
            [5, 6, 7, 8, 9, 10],
            [6, 7, 8, 9, 10, 11],
            [7, 8, 9, 10, 11, 12]]

print(dice_sum[1][5])
# 맨마지막줄 7, 8,9,10,11,12에서의 8 이 출력
print(dice_sum[5][1])
# 2번째줄 3,4,5,6,7,8에서 8이 출력
```

```
8
8
```

In [46]:
```
die = [i for i in range(1,7)]

dice_sum = [[j+i for i in die] for j in die]

print(dice_sum[1][5])
print(dice_sum[5][1])
#위와 동일
```

```
8
8
```

도전문제

표절 검사 프로그램

강의노트 07 자료구조 collections 설명 참고

아주대학교 글로벌 경영학과의 한 교수님은 과제의 표절 검사를 쉽게 하기 위해 Python을 통한 간단한 표절 검사 프로그램을 작성해보고자 한다.

현재 구상 중인 프로그램은 복잡한 알고리즘을 필요로하지 않고, 간단하게 단어 빈도를 기반으로 하여, 그 유사도를 측정하고자한다.

In [47]:
```
from collections import defaultdict, Counter

text = """Python is a very simple programming language so even if yo
u are new to programming, you can learn python without facing any is
sues."""

text2 = """C is a very difficult programming language so even if you
are good at programming, you can learn c with facing any issues."""

text3 = """R Programming is good at statistical analysis. you can le
arn easily"""
```

문제1

> Q: defaultdict를 활용하여 text를 입력받으면 단어별 빈도를 측정하여 반환하는 함수 word_counter를 만드시오.

HINT

1. collections 모듈의 defaultdict는 단순한 dict와 다르게, 인덱싱에서 key 값이 없으면 오류가 아닌 0을 기본 값으로 가지게 한다.

In [52]:
```
# word_dict = dict()
# word_dict["key"]

## KeyError

# word_dict = defaultdict(lambda: 0)
# word_dict["key"]
## 0

# word_dict["key"] += 1
# word_dict["key"]
## 1
```

유사도 측정을 위해 문장을 단어별로 분할해야하며, 편의를 위해 모두 소문자로 바꿔준다.

- split
- lower

In [53]:
```
def word_counter(text):
    word_count = defaultdict(lambda: 0)
    for word in text.lower().split():
        word_count[word] += 1

    return word_count
```

In [54]:
```
word_counter(text)
```

Out[54]:
```
defaultdict(<function __main__.word_counter.<locals>.<lambda>()>,
            {'python': 2,
             'is': 1,
             'a': 1,
             'very': 1,
             'simple': 1,
             'programming': 1,
             'language': 1,
             'so': 1,
             'even': 1,
             'if': 1,
             'you': 2,
             'are': 1,
             'new': 1,
             'to': 1,
             'programming,': 1,
             'can': 1,
             'learn': 1,
             'without': 1,
             'facing': 1,
             'any': 1,
             'issues.': 1})
```

문제2

> Q: 도전문제 1의 word_counter 활용하여 text와 text2의 유사도와 text와 text3의 유사도를 구하시오.

HINT collections 모듈의 Counter는 dict의 형태이지만 Counter들 간의 덧셈, 뺄셈 연산이 가능하며 defaultdict를 Counter로 변환할 수 있다. Counter({"a": 1, "b": 2, "c": 3}) - Counter({"a": 1, "b": 1, "c": 1})

Counter({'b': 1, 'c': 2}) dictionary 형태의 모든 자료구조는 .values() 를 통해 value 값만 추출할 수 있다.
sum(Counter({"a": 1, "b": 2, "c": 3}).values()) # 전체 단어수 합

6

1. Counter(A)가 Counter(B)와 얼마나 유사한지는 다음과 같은 공식을 따른다고 한다.(시그마는 해당 Counter dict 안의 value 값을 모두 합하라는 의미)

In [56]:
```
def text_similarity(text_count_1, text_count_2):
    text1_count = Counter(text_count_1)
    text2_count = Counter(text_count_2)

    word_total = sum(text1_count.values())
    word_diff = sum((text1_count - text2_count).values())

    return (1 - word_diff / word_total) * 100
```

In [57]:
```
text_similarity(word_counter(text), word_counter(text2))
#위에 word_counter과 주석으로 단 text등의 데이터를 넣어서 유사도 측정
```

Out[57]: 73.91304347826086

In [59]:
```
text_similarity(word_counter(text), word_counter(text3))
#위에 word_counter과 주석으로 단 text등의 데이터를 넣어서 유사도 측정
```

Out[59]: 21.739130434782606

파이썬 스타일 코드2 - 연습해보기

--

실습코드

1. 람다함수

- 람다(lambda) 함수는 함수의 이름 없이, 함수처럼 사용할 수 있는 익명의 함수를 말한다. 선형대수나 미적분 등의 과목을 수강하다 보면, 한 번쯤 람다 대수라는 표현을 들어 보았을 것이다. 람다 함수의 '람다'는 바로 이 람다 대수에서 유래하였다. 일반적으로 람다 함수는 이름을 지정하지 않아도 사용할 수 있다.

1.1. 기존 함수

In [61]:
```
def f(x,y):
    return x + y

print(f(1,4))
# 1더하기4
```

5

1.2. lambda 함수 할당

In [62]:
```
f=lambda x,y: x + y
print(f(1,4))
# 1더하기 4
```

5

1.3. 익명의 lambda 함수

In [63]:
```
print((lambda x, y:x + y)(1, 4))
# 동일하게 1더하기 4
```

5

2. 맵리듀스

2.1. map 함수

연속 데이터를 저장하는 시퀀스 자료형에서 요소마다 같은 기능을 적용할 때 사용한다. 일반적으로 리스트나 튜플처럼 요소가 있는 시퀀스 자료형에 사용된다. 다음의 사용 예제를 보자.

In [66]:
```
ex = [1,2,3,4,5]
f = lambda x:x**2
print(list(map(f, ex)))
# 제곱수 구하기 리스트로 출력
```

```
[1, 4, 9, 16, 25]
```

- 코드 설명
- 위 코드에서는 먼저 ex라는 이름의 리스트를 만들고, 입력된 값을 제곱하는 람다함수 f를 생성하였다. 그리고 'map(함수이름, 리스트 데이터)'의 구조에서 map(f,ex) 코드를 실행한다. 이는 해당 코드로 함수 f를 ex의 각 요소에 매핑하라는 뜻이다.
- 파이썬 2.x와 3.x의 차이는 제너레이터의 사용인데 3.x 부터는 map()함수의 기본 반환이 제너레이터이므로 list() 함수를 사용해야 리스트로 반환된다.
- 제너레이터(generator)는 시퀀스 자료형의 데이터를 처리할 때, 실행 시점의 값을 생성하여 효율적으로 메모리를 관리할 수 있다는 장점이 있다.
- 만일 list를 붙이지 않는다면, 다음 코드처럼 코딩할 수도 있다. 여기서 함수는 반드시 람다함수일 필요는 없고, 일반 함수를 만들어 사용해도 문제 없다.

In [67]:
```
ex=[1,2,3,4,5]
f=lambda x:x**2
for value in map(f,ex):
    print(value)
#제곱수 구하기 map사용
```

```
1
4
9
16
25
```

- 리스트 컴프리헨션과 비교 최근에는 람다함수나 map() 함수를 프로그램 개발에 사용하는 것을 권장하지 않는다. 굳이 두 함수를 쓰지 않더라도 리스트 컴프리헨션 기법으로 얼마든지 같은 효과를 낼 수 있기 때문이다. 만약 위 코드를 리스트 컴프리헨션으로 변경하고자 한다면, 다음처럼 코딩하면 된다.

In [69]:
```
ex = [1, 2, 3, 4, 5]
[x**2 for x in ex]
# 먼저 만든 리스트에 반복문 사용해 제곱수 구하기
```

Out[69]:
```
[1, 4, 9, 16, 25]
```

- 한개 이상의 시퀀스 자료형 데이터의 처리 map()함수는 2개 이상의 시퀀스 자료형 데이터를 처리하는 데도 문제가 없다.

In [73]:
```
ex=[1,2,3,4,5]
f=lambda x,y:x+y
list(map(f,ex,ex))
# 1+1 2+2 3+3 처럼 동일 수 더하기
```

Out[73]:
```
[2, 4, 6, 8, 10]
```

In [76]:
```
[x+y for x,y in zip(ex,ex)] # 리스트 컴프리헨션 용법
```

Out[76]:
```
[2, 4, 6, 8, 10]
```

2.2. reduce 함수

- map() 함수와 다르지만 형제처럼 사용하는 함수로 리스트와 같은 시퀀스 자료형에 차례대로 함수를 적용하여 모든 값을 통합하는 함수이다.
- lambda 함수와 함께 쓰여 좀 복잡해 보여 예전에는 많이 쓰였으나 최근 버전에서는 사용을 권장하지 않는다. 그러나 많은 코드들이 여전히 사용하고 있어 이해차원에서 배울 필요가 있다.

In [79]:
```
from functools import reduce
print(reduce(lambda x,y:x+y, [1,2,3,4,5]))
#패키지에서 모듈 불러와서 출력
```

```
15
```

- 비교 코드

In [81]:
```
x=0
for y in [1,2,3,4,5]:
    x += y
print(x)
# 패키지 사용하지 않고 1부터 5까지 쭉 더해 출력
```

```
15
```

3. 별표의 활용

3.1. 가변 인수로 활용

- 가변 인수

In [83]:
```
def asterisk_test(a, *args):
    print(a,args)
    print(type(args))

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

# 객체 이용 tuple
```

```
1 (2, 3, 4, 5, 6)
<class 'tuple'>
```

- 키워드 가변 인수

In [85]:
```
def asterisk_test(a,**kargs):
    print(a,kargs)
    print(type(kargs))
asterisk_test(1,b=2,c=3,d=4,e=5,f=6)
# 딕셔너리 사용
```

```
1 {'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6}
<class 'dict'>
```

3.2. 별표의 언패킹 기능

- 함수에서의 사용

In [87]:
```
def asterisk_test(a,args):
    print(a,*args)
    print(type(args))
asterisk_test(1,(2,3,4,5,6))
#위와 동일한데 별표 언패킹
```

```
1 2 3 4 5 6
<class 'tuple'>
```

In [88]:
```
def asterisk_test(a,args):
    print(a,args)
    print(type(args))
asterisk_test(1,(2,3,4,5,6))
#위와 일맥상통
```

```
1 (2, 3, 4, 5, 6)
<class 'tuple'>
```

- 일반 자료형에서의 사용

In [91]:
```
a,b,c=([1,2], [3,4], [5,6])
print(a,b,c)
data=([1,2], [3,4], [5,6])
print(*data)
# 2차원 리스트
```

```
[1, 2] [3, 4] [5, 6]
[1, 2] [3, 4] [5, 6]
```

- zip 함수와의 응용

In [92]:
```
for data in zip(*[[1,2],[3,4],[5,6]]):
    print(data)
    print(type(data))

# 반복문을 통해 튜플 앞 1,3,5 / 2,4,6
```

```
(1, 3, 5)
<class 'tuple'>
(2, 4, 6)
<class 'tuple'>
```

- 키워드 가변 인수 응용

In [93]:
```
def asterisk_test(a,b,c,d):
    print(a,b,c,d)
data={"b":1, "c":2, "d":3}
asterisk_test(10, **data)
```

```
10 1 2 3
```

4. 선형대수학

4.1. 파이썬 스타일 코드로 표현한 벡터

In [95]:
```
vector_a=[1,2,10]  # 리스트로 표현한 경우
vector_b=(1,2,10)   # 튜플로 표현한 경우
vector_c={'x':1, 'y':2, 'z':10}  # 딕셔너리로 표현한 경우
```

- 벡터의 연산: 벡터합

In [97]:
```
u=[2,2]
v=[2,3]
z=[3,5]
result=[]

for i in range(len(u)):
    result.append(u[i]+v[i]+z[i])
print(result)
```

```
[7, 10]
```

In [98]:
```
u=[2,2]
v=[2,3]
z=[3,5]
result=[sum(t) for t in zip(u,v,z)]
print(result)
```

```
[7, 10]
```

- 별표를 사용한 함수화

In [100]:
```
def vector_addition(*args):
    return [sum(t) for t in zip(*args)]   # unpacking 통해 zip(u,v,z) 효과를 낼 수 있음.

vector_addition(u,v,z)
```

Out[100]:
```
[7, 10]
```

- 간단한 두벡터의 합

In [101]:
```
a = [1, 1]
b = [2, 2]

[x + y for x, y in zip(a, b)]
```

Out[101]:
```
[3, 3]
```

- 벡터의 연산: 스칼라곱

In [103]:
```
u=[1,2,3]
v=[4,4,4]

alpha=2

result=[alpha*sum(t) for t in zip(u,v)]
result
```

Out[103]:
```
[10, 12, 14]
```

4.2. 파이썬 스타일코드로 표현한 행렬

딕셔너리로 표현하는 경우 좌표정보나 이름정보를 넣을 수 있으나 복잡함

In [104]:
```
matrix_a=[[3,6], [4,5]] #리스트로 표현한 경우
matrix_b=[(3,6), (4,5)] #튜플로 표현한 경우
matrix_c={(0,0):3, (0,1):6, (1,0):4, (1,1):5}  #디셔너리로 표현한경우
```

- 행렬의 연산: 행렬의 elemnet-wise 합

In [106]:
```
matrix_a=[[3,6], [4,5]]
matrix_b=[[5,8], [6,7]]

result=[[sum(row) for row in zip(*t)] for t in zip(matrix_a, matrix_
b)]
print(result)
```

```
[[8, 14], [10, 12]]
```

일반문제

주민등록번호로 성별 찾기 with map

PR6에서 split을 활용하여 주민등록번호 뒷자리의 맨 첫 번째 숫자를 추출하여 성별을 알아내는 과정을 구현하였다. 이번에는 여러개의 요소를 가지는 다음과 같은 리스트에서 성별을 찾는 과정을 맵리듀스를 이용해 간단하게 구현해보자.

> pins = ["891120-1234567", "931120-2335567", "911120-1234234", "951120-1234567"]
>
> Q: lambda와 map을 사용하여 위의 리스트에서 출력결과 예시와 같이 성별을 나타내는 값을 추출하시오.

출력결과 예시

CODE

['1', '2', '1', '1']

HINT

1. lambda 함수로 주민등록번호 문자열에서 성별을 추출하는 과정을 구현한다.
2. map 함수에 해당 lambda함수와 주민등록번호 리스트를 입력한다.
3. 실습코드 2.1.에서 map 과 lambda를 어떻게 함께 사용하는지와 결과를 list로 출력하는 과정을 참고하세요.

In [109]:
```
pins = ["891120-1234567", "931120-2335567", "911120-1234234", "951120-1234567"]


list(map(lambda x: x.split("-")[1][0], pins))
# -를 기준으로 나누고 맨 앞자리 숫자가 성별 나타냄
```

Out[109]:
```
['1', '2', '1', '1']
```

도전문제

벡터의 내적

크기가 같은 두 벡터의 내적은 벡터의 각 성분별 곱한 값을 더해준 값이다.

v1*w1+…+vn*wn

> Q: 크기가 같은 두 벡터 (list 형태)를 받으면 이를 내적한 값을 도출하는 함수 dot을 구현하고, 이를 활용하여 a=[1, 2], b=[3,4]를 내적한 값을 구하시오.

HINT

1. 실습코드 4.1.에서는 벡터의 합과 곱에 대한 연산만을 다루고 있습니다. 이중 간단한 벡터의 합에서 리스트 컴프리헨션을 사용한 방법에서 연산을 바꾸면 각 벡터별 곱을 간단히 구할 수 있습니다.

In [111]:
```
a = [1, 2]
b = [3, 4]

dot = lambda a,b : sum([x*y for x, y in zip(a, b)])
dot(a,b)

# 3+8 = 11
```

Out[111]:
```
11
```