Fastcampus

Computer Science Extension School

Python Basic_Day7

List Comprehension

존재하는 리스트를 활용하여 새로운 리스트를 생성하는 방법

비슷한 표현들

- Set Comprehension
- Dictionary Comprehension
- Parallel list Comprehension

List Comprehension

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        doubled_list.append(i * 2)
```

List Comprehension

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        doubled_list.append(i * 2)
```

```
doubled_list = []
```

List Comprehension

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        doubled_list.append(i * 2)
```

```
doubled_list = [i * 2]
```

List Comprehension

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        doubled_list.append(i * 2)
```

```
doubled_list = [i * 2 for i in old_list]
```

List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        if i % 2 == 0:
                doubled_list.append(i * 2)
```

List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        if i % 2 == 0:
                doubled_list.append(i * 2)
```

```
doubled_list = []
```

List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        if i % 2 == 0:
                doubled_list.append(i * 2)
```

```
doubled_list = [i * 2]
```

List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        if i % 2 == 0:
                doubled_list.append(i * 2)
```

```
doubled_list = [i * 2 for i in old_list]
```

List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        if i % 2 == 0:
                doubled_list.append(i * 2)
```

```
doubled_list = [i * 2 for i in old_list if i % 2 == 0]
```

Mini Project

- List comprehension 으로 FizzBuzz 한줄로 구현하기

```
["Fizz"*(not i%3) + "Buzz"*(not i%5) or i for i in range(1,100)]
```

File I/O

File I/O

```
f = open(filename, mode)
f.close()
```

mode

r - 읽기모드

w - 쓰기모드

a - 추가모드(파일의 마지막에 새로운 내용을 추가)

Create New File

```
f = open("Newfile.txt", 'w')
f.close()
```

Write text

```
f = open("Newfile.txt", 'a')
for i in range(1,11):
    text = "line %d. \n" % i
    f.write(text)
f.close()
```

Read text

```
f = open("Newfile.txt", 'r')
text = f.readline()
print(text)
f.close()
```

Read All text

```
f = open("Newfile.txt", 'r')
while True:
        text = f.readline()
        if not text: break
        print(text)
f.close()
```

Read All text using readlines

```
f = open("Newfile.txt", 'r')
texts = f.readlines()
for text in texts:
        print(texts)
f.close()
```

Add text

```
f = open("Newfile.txt", 'a')
for i in range(11, 20):
        text = "New line %d \n" % i
        f.write(text)
f.close()
```

Get rid of f.close()

```
with open("foo.txt", 'w') as f:
        f.write("foo is text dummy")
```

Error Handle

by using `try, except`

필요한 만큼만 적절히 사용하셔야 합니다 by PEP 8

Error Handle - Syntax

```
try:
        실행문
except:
        실행문
```

Error Handle - ValueError

```
try:
        some_input = int(input("type some number: "))
except ValueError:
        print("I said type some NUMBER!!!!")
```

Error Handle - ValueError

```
try:
        some_input = int(input("type some number: "))
except ValueError as e:
        print("I said type some NUMBER!!!!")
    print(e)
```

Error Handle - FileNotFoundError

```
try:
        f = open('error_example.txt', 'r')
except FileNotFoundError as e:
        print(e)
else:
        text = f.read()
        f.close()
```

Error Handle - Multiple Error

```
try:
        ...
except error type 1:
        ...
except error type 2:
        ...
```

Error Handle - Pass Error

```
try:
        f = open('error_example.txt', 'r')
except FileNotFoundError as e:
        pass
else:
        text = f.read()
    f.close()
```

lambda

lambda

- 익명함수(이름이 없는 함수)
- 간단한 수식을 함수로 지정해 한 두번 쓸 용도로 사용할 때
- 두 줄 이상 실행될 함수는 그냥 함수로 정의하는게 나음!

lambda

- python은 모든 것이 객체로 존재
- 간단한 연산 함수 조차 객체로 존재하여 리소스를 점유

lambda - traditional function

```
def get_next_integer(a):
        return a + 1
```

lambda - lambda function

```
lambda a: a+1
```

lambda - usual example

```
>>> (lambda a: a+1)(12)
13
```

lambda

- can't assign to lambda

```
(lambda a,b: a*b)(10,20)
```

- only expression, not statement

map, reduce, filter

map

- list의 각 element에 대해 특정한 함수를 적용

map - example

```
def get_squared(num_list):
    squared = []
    for num in num_list:
        squared.append(num**2)
    return squared
```

map - example

```
def squared_lambda(x):
    return x ** 2

list(map(squared_lambda, [1,2,3,4]))
```

map with lambda - example

```
list(map(lambda x: x**2, [1,2,3,4]))
```

map is rather than for

```
def print_with_sleep(x):
    time.sleep(1)
    return x ** 2
```

```
m = map(print_with_sleep, [1,2,3])
next(m)
next(m)
next(m)
..
```

```
for i in range(1,3+1):
    print_with_sleep(i)
```

timing is perfect!

```
m = map(print_with_sleep, [1,2,3])
for i in m:
    print(i)
```

```
m = map(print_with_sleep, [1,2,3])
list(m)
```

map is rather than for

- map은 제너레이터를 생성해 함수와 인자를 바인딩만 하고, 필요할 때 순회하며 값을 처리
- for는 한번에 처리하므로 for 수행 중 다른 일을 할 수 없음

filter

- 특정함수를 만족하는 요소만 남기는 필터

filter - example

```
def even_selector(x):
    if x % 2 == 0:
        return True
    else:
        return False

filter(even_selector, range(1,10+1))
```

filter with lambda - example

```
filter(lambda x: x%2==0, range(1,10+1))
```

Mini Project

```
recycle_bin = [1,2, "Fastcampus", [], 5,4, 5.6, "패스트캠퍼스"]
```

위 리스트의 요소 중 정수만 제곱하여 출력하기

map, filter, lambda를 모두 사용

> Hint: isinstance(1, int)