DAY 3

목차

- fluent python
  - enumerate
  - zip
  - lambda
  - map
  - List comprehension

enumerate

1. 문제를 먼저 풀어보자. 여기는 동네 유명한 빵집이다. 사람들에게 먼저 온 순서대로 번호표를 나누어주려고 한다. 번호표를 나누어주는 함수를 작성해보자.
   - 함수는 사람이름으로 되어 있는 리스트를 받아서 "대기번호 x번 : 사람이름" 를 화면에 출력하고 (번호표, 사람이름)을 원소로 이루어진 리스트를 반환한다.

enumerate

```
In [1]: people = ['펭수', '뽀로로', '뚝딱이', '텔레토비']
```

```
In [2]: def func1(line) :
            new_lines = []
            i = 1
            for x in line :
                print("대기번호 %d번 : %s" %(i, x))
                new_lines.append((i, x))
                i += 1
            return new_lines
```

→ 대기 번호를 트래킹하는 변수 i

→ 별도로 업데이트 해야함

```
In [3]: lines = func1(people)
```

```
대기번호 1번 : 펭수
대기번호 2번 : 뽀로로
대기번호 3번 : 뚝딱이
대기번호 4번 : 텔레토비
```

enumerate

- 반복가능한 객체의 인덱스와 원소에 함께 접근할 수 있는 함수
- tuple(인덱스, 원소)의 형태로 객체를 반환
- 보통 리스트와 함께 쓰임

```
In [23]: lst = ['a','b','c']
         for x in enumerate(lst) :
             print(x)

(0, 'a')
(1, 'b')
(2, 'c')
```

```
In [18]: st = 'abcd'

         for x in enumerate(st) :
             print(x)

(0, 'a')
(1, 'b')
(2, 'c')
(3, 'd')
```

```
In [16]: dic = {0 : 'p', 1 : 'b' , 2 : 'd'}

         for x in enumerate(dic):
             print(x)

(0, 0)
(1, 1)
(2, 2)
```

```
In [22]: se = {'a','b','c'}
         for x in enumerate(se) :
             print(x)

(0, 'c')
(1, 'b')
(2, 'a')
```

* 주의 * set은 순서가 없는 자료형

zip

- 반복가능한 객체들을(2개 이상) 병렬적으로 묶어주는 함수
- 각 원소들을 튜플의 형식으로 묶어줌

In [13]:
```
str_list = ['one', 'two', 'three', 'four']
num_list = [1, 2, 3, 4]

for i in zip(num_list, str_list) :
    print(i)
```

```
(1, 'one')
(2, 'two')
(3, 'three')
(4, 'four')
```

In [14]:
```
s1 = '123'
s2 = 'abc'
s3 = 'ㄱㄴㄷ'
list(zip(s1,s2,s3))
```

Out[14]:
```
[('1', 'a', 'ㄱ'), ('2', 'b', 'ㄴ'), ('3', 'c', 'ㄷ')]
```

lambda

- lambda(람다)는 식 형태로 되어있어서 lambda expression(람다 표현식)이라고도 불림
- 람다는 익명의 함수로서 함수를 간편하게 작성할 수 있게 해줌
- python3에서 사용이 권장 되지는 않지만 머신러닝이나 데이터 분석시 많이 사용됨

lambda 매개변수 **: 리턴 값**

lambda

```
In [85]: def plus_two(num) :
             return num + 2
```

```
In [86]: a = 2
         b = plus_two(a)
         print(b)
```

```
4
```

```
In [88]: lambda x : x + 2
Out[88]: <function __main__.<lambda>(x)>
```

→ 람다는 함수를 생성함

```
In [89]: func2 = lambda x : x + 2
```

```
In [90]: c = func2(2)
```

```
In [91]: c
Out[91]: 4
```

map

- 리스트, 튜플, 스트링 등 자료형 각각의 원소에 동일한 함수를 적용

map (함수, 자료형 **)**

```
items = [x for x in range(1, 6)]
print(items)
```

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

```
squared = []
for i in items :
    squared.append(i*i)
```

```
print(squared)
```

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

map

```
squared_map = list(map(lambda x : x**2, items))
```

```
print(squared_map)
```

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

List comprehension

- 파이썬만의 독특한 문법으로 간결한 코딩을 할 수 있음

1) for 문

0부터 9까지를 순서대로 가지고 있는 리스트를 만드세요.

In [24]:
```
list_1 = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [25]:
```
list_2 = []
for x in range(10) :
    list_2.append(x)
print(list_2)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [26]:
```
lc = [x for x in range(10)]
print(lc)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

List comprehension

- 파이썬만의 독특한 문법으로 간결한 코딩을 할 수 있음

1) for 문

0부터 9까지를 순서대로 가지고 있는 리스트를 만드세요.

```
In [26]: lc = [x for x in range(10)]
         print(lc)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

변수 = **[** 저장할 값 **for** 원소 **in** 반복 가능한 객체 **]**

List comprehension

2) for 문 + if 문

10부터 20 사이의 숫자들 중에서 짝수만을 담은 리스트를 만들어보자.

```
In [37]: list_3 = []
         for x in range(10, 21) :
             if x % 2 == 0 :
                 list_3.append(x)
         print(list_3)

         [10, 12, 14, 16, 18, 20]

In [38]: lc_2 = [x for x in range(10, 21) if x % 2 == 0]

In [39]: lc_2

Out[39]: [10, 12, 14, 16, 18, 20]
```

변수 = **[** 저장할 값 **for** 원소 **in** 반복 가능한 객체 **if 조건]**

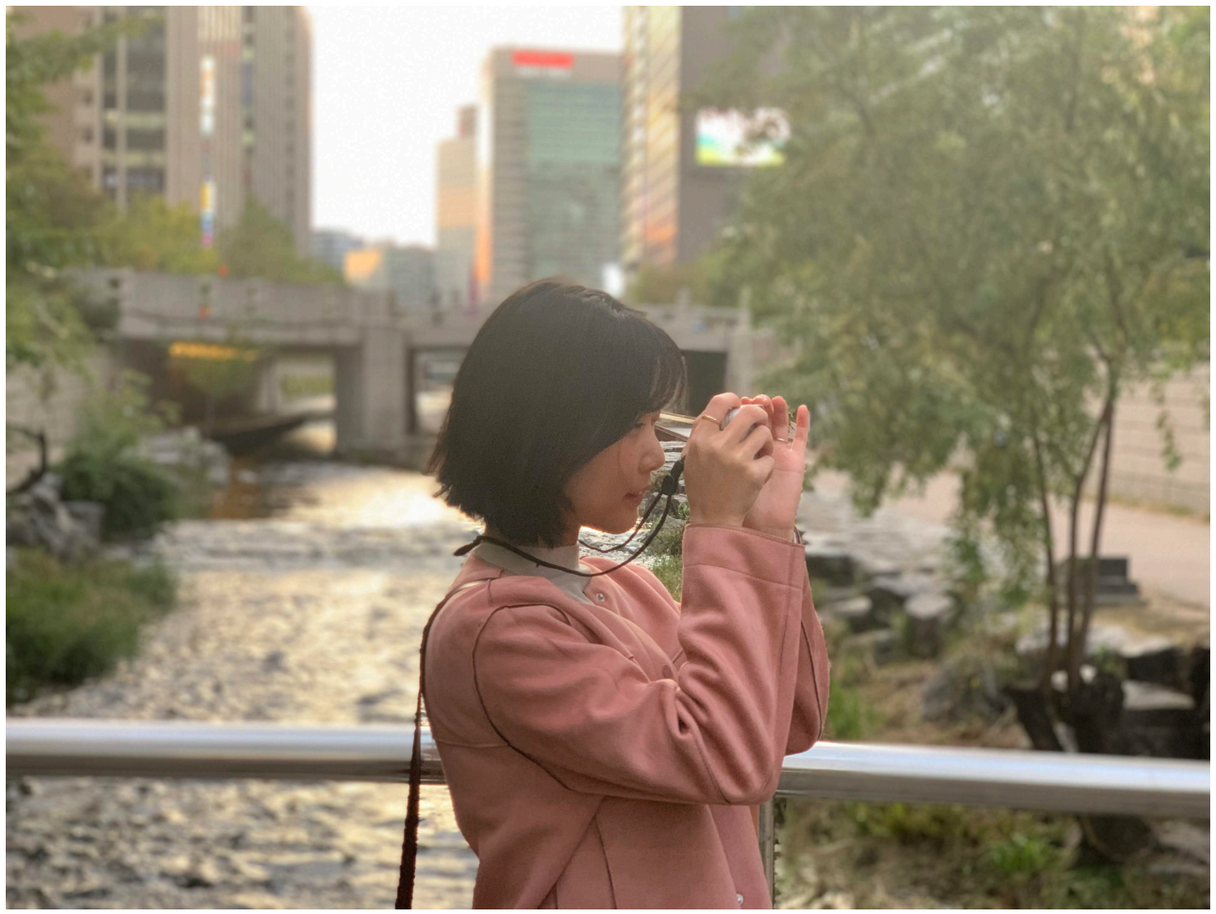
List comprehension

2) for 문 + if 문

1부터 10 까지의 숫자들 중 홀수이면 제곱수를, 짝수이면 세제곱수를 담은 리스트를 만들어보자.

```
In [43]: list_4 = []
         for x in range(1, 11) :
             if x % 2 == 1 :
                 list_4.append(x ** 2)
             else :
                 list_4.append(x ** 3)

In [44]: list_4

Out[44]: [1, 8, 9, 64, 25, 216, 49, 512, 81, 1000]

In [45]: lc_4 = [x ** 2 if x % 2 == 1 else x ** 3 for x in range(1 , 11)]

In [46]: lc_4

Out[46]: [1, 8, 9, 64, 25, 216, 49, 512, 81, 1000]
```

변수 = **[** 저장할 값 **if** 조건 **else** 저장할 값 **for** 원소 **in** 반복 가능한 객체**]**



정민지

mjmingd@gmail.com

github.com/mjmingd/python_basic