연세대학교
YONSEI UNIVERSITY

Python Extension
(Comprehension, Zip, Enumerate)

Yonsei University Digital Analytics
Bo Kyung Seo
2019.02.26

Contents

1. Comprehension

- Iterable을 간결하게 표현하기 위한 방법
  List_name = [expression **for** item **in** list **if** conditional]

```python
# 12 original
def solve(n):
    ans = []
    i = 1

    while i <= n:
        if n%i == 0:
            ans.append(i)
        i = i+1
    s = sum(ans)
    return s

n = int(input("입력: "))
print("약수의 합: {0}".format(solve(n)))
```

[1]

```python
# 12 using list comprehension
def solve(n):
    ans = []
    [ans.append(i) for i in range(1,n+1) if n%i == 0]
    s = sum(ans)

    return s

n = int(input("입력: "))
print("약수의 합: {0}".format(solve(n)))
```

Output

```
입력: 20
약수의 합: 42
```

1. Comprehension

```python
# runtime 측정을 위해 time 모듈을 import
import time

# 구구단 2중 for문
start = time.time()
for i in range(1,10):
    print('구구단 ', i, '단은: ', end="")
    for j in range(1, 10):
        print(i * j, end=' ')
    print('₩n')
end = time.time()
print('runtime: ', end-start, '₩n₩n')
```

```python
# 구구단 comprehension 사용
start2 = time.time()
for i in range(1, 10):
    print('구구단 ', i, '단은: ', end="")
    v = [ print(i * j, end=' ') for j in range(1, 10)]
    print('₩n')
end2 = time.time()

print('runtime: ', end2-start2)
```

Output

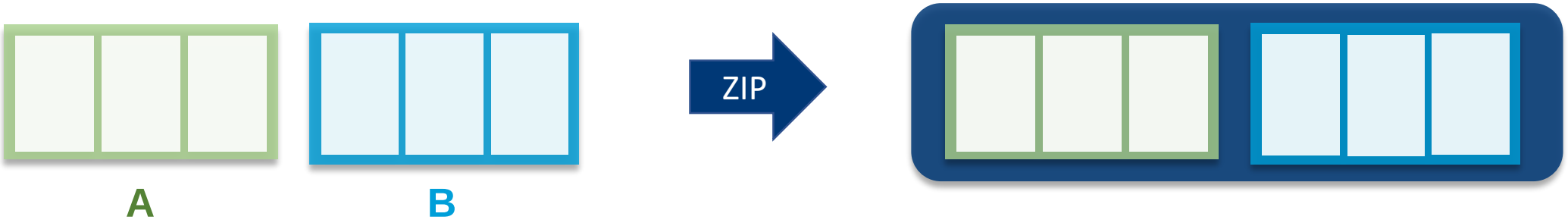
```
구구단  1 단은: 1 2 3 4 5 6 7 8 9
구구단  2 단은: 2 4 6 8 10 12 14 16 18
구구단  3 단은: 3 6 9 12 15 18 21 24 27
구구단  4 단은: 4 8 12 16 20 24 28 32 36
구구단  5 단은: 5 10 15 20 25 30 35 40 45
구구단  6 단은: 6 12 18 24 30 36 42 48 54
구구단  7 단은: 7 14 21 28 35 42 49 56 63
구구단  8 단은: 8 16 24 32 40 48 56 64 72
구구단  9 단은: 9 18 27 36 45 54 63 72 81
```

❶ runtime: 0.003979921340942383

❷ runtime: 0.0019936561584472656

2. Zip

- Zip(*iterable): 동일한 개수로 이루어진 자료형들을 묶음
  (* iterable은 반복 가능한 자료형 여러 개가 입력으로 가능하다는 의미) [2]



- 길이가 다른 iterator를 사용하면 결과를 잘라냄
- 내장 모듈 **itertools**의 **zip_longest** 함수를 사용하면 튜플을 None으로 채워가며 모든 iterator을 소모할 때까지 생성 [3]

2. Zip

- E.g.) Zip(*iterable): 동일한 개수로 이루어진 자료형들을 묶음

1)

```
1  list(zip('abc', 'def'))
```

Output

```
[('a', 'd'), ('b', 'e'), ('c', 'f')]
```

2)

```
1  A = ['남자', '아빠', '남성']
2  B = ['여자', '엄마', '여성']
3  result = {A:B for A,B in zip(A,B)}
4  print(result)
5  print(type(result))
```

Output

```
{'남자': '여자', '아빠': '엄마', '남성': '여성'}
<class 'dict'>
```

2. Zip

- E.g.) 길이가 다른 iterator를 사용하면 짧은 길이에 맞춤

```python
list1 = [10, 20, 30, 40]
list2 = [11, 21, 31]

result = zip(list1, list2)
print(result)
print(type(result))
for e1, e2 in result:
    print(e1, e2)
```

Output

```
<zip object at 0x0000198280F2708>
<class 'zip'>
10 11
20 21
30 31
```

```python
list1 = [10, 20, 30, 40]
list2 = [11, 21, 31]

result = list(zip(list1, list2))
print(result)
print(type(result))
```

Output

```
[(10, 11), (20, 21), (30, 31)]
<class 'list'>
```

2. Zip

- E.g.) 내장 모듈 **itertools**의 **zip_longest** 함수를 사용하면 튜플을 None으로 채워가며 모든 iterator을 소모할 때까지 생성 [3]

```
1  import itertools
2
3  list1 = [10, 20, 30, 40]
4  list2 = [11, 21, 31]
5
6  result = itertools.zip_longest(list1, list2)
7  print(result)
8  print(type(result))
9
10 result = list(result)
11 print(result)
```

Output

```
<itertools.zip_longest object at 0x0000014DFD02CAE8>
<class 'itertools.zip_longest'>
[(10, 11), (20, 21), (30, 31), (40, None)]
```

3. Enumerate

- 순서가 있는 자료형(list, tuple, string)을 입력으로 받아 **인덱스 값**을 포함하여 결과를 출력
- Enumerate를 for문과 함께 사용하면 자료형의 인덱스와 그 값을 쉽게 알 수 있음
- for문 처럼 반복되는 구간에서 객체가 현재 어느 위치에 있는지 알려주는 인덱스 값이 필요할 때 유용 [2]

3. Enumerate

- E.g.)

```
1 flavor_list = ['vanilla', 'chocolate', 'pecan', 'strawberry']
2 for i in range(len(flavor_list)):
3     flavor = flavor_list[i]
4     print('%d: %s' % (i + 1, flavor))
```

Output

```
1: vanilla
2: chocolate
3: pecan
4: strawberry
```

```
1 for i, flavor in enumerate(flavor_list):
2     print('%d: %s' % (i+1, flavor))
```

```
1 for i, flavor in enumerate(flavor_list, start=1):
2     print('%d: %s' % (i, flavor))
```

[4]

Output

```
1: vanilla
2: chocolate
3: pecan
4: strawberry
```

3. Enumerate

- Enumerate vs Zip:
  상황에 맞게 적절하게 내장함수들을 사용하는 것이 중요

```python
names = ['Cecilia', 'Lise', 'Marie']
letters = [len(n) for n in names]
```

```python
longest_name = None
max_letters = 0

for i in range(len(names)):
    count = letters[i]
    if count > max_letters:
        longest_name = names[i]
        max_letters = count

print(longest_name)
```

```python
longest_name = None
max_letters = 0

for i, name in enumerate(names):
    count = letters[i]
    if count > max_letters:
        longest_name = name
        max_letters = count

print(longest_name)
```

```python
longest_name = None
max_letters = 0

for name, count in zip(names, letters):
    if count > max_letters:
        longest_name = name
        max_letters = count

print(longest_name)
```

[4]

Output

```
Cecilia
```

4. Examples

❖ Comprehension & Enumerate **(Medium)**

- Problem: 귀여운 라이언
  라이언 인형과 어피치 인형이 N개 일렬로 놓여 있다. 라이언 인형은 1, 어피치 인형은 2로 표현한다. 라이언 인형이 K개 이상 있는 가장 작은 연속된 인형들의 집합의 크기를 구하여라.(== K번 째 라이언 인형의 인덱스를 구하시오.) 그런 집합이 없다면 -1을 출력한다. [5]

Output

```
N과 K 입력 (e.g) 10 3): 10 3
인형의 정보 입력 (e.g) 1 2 2 2 1 2 1 2 2 1):1 2 2 2 1 2 1 2 2 1
6
```

4. Examples

❖ Comprehension & Enumerate (Medium)

```
1  N, K = map(int, input("N과 K 입력 (e.g) 10 3): ", ).split())
2  arr = input("인형의 정보 입력 (e.g) 1 2 2 2 1 2 1 2 2 1):", ).split()
3
4  if arr.count('1') >= K:
5      ryan = [i for i, x in enumerate(arr) if x == '1']
6      print(ryan[K-1])
7  else:
8      print(-1)
```

Output

```
N과 K 입력 (e.g) 10 3): 10 3
인형의 정보 입력 (e.g) 1 2 2 2 1 2 1 2 2 1):1 2 2 2 1 2 1 2 2 1
6
```

4. Examples

❖ Comprehension & Zip & Enumerate **(Hard)**

- Problem: 음료 주문 프로젝트

  menu_print: 인덱스, 메뉴와 금액을 출력 함수

  menu_select: 메뉴 선택 함수, 메뉴 인덱스로 주문, 반복하여 음료 주문, 0 입력 시 주문 완료 (return: total price)

  menu_pay(total price): 선택한 메뉴 결제 함수, 리스트 bill에 저장된 금액을 인덱스와 함께 출력, 인덱스로 지불 금액 선택, 거스름 돈 출력 [6]

  *** Comprehension, Zip, Enumerate를 모두 한 번 이상 사용하여 코딩**

4. Examples

❖ Comprehension & Zip & Enumerate **(Hard)**

```
1  menu = ['','Americano', 'Latte', 'Espresso', 'Mocha', '식혜', '수정과']
2  price = [0, 1500, 2000, 1700, 2500, 2000, 1900]
3  bill = [0, 50000, 10000, 5000, 1000]
```

```
1  menu_print()
2  total = menu_select()
3  menu_pay(total)
```

[6]

Output

```
1. Americano  1500
2. Latte  2000
3. Espresso  1700
4. Mocha  2500
5. 식혜  2000
6. 수정과  1900
음료를 선택하세요 : 1
Americano 1500 원  합계  1500 원

계속 주문은 음료 번호를, 지불은 0을 누르세요 : 4
Mocha 2500 원  합계  4000 원

계속 주문은 음료 번호를, 지불은 0을 누르세요 : 0
주문이 완료되었습니다.
1 . 50000 원   2 . 10000 원   3 . 5000 원   4 . 1000 원
지불 금액을 입력하세요 : 2
총 지불액 : 10000 원
거스름  6000 원
```

4. Examples

❖ Comprehension & Zip & Enumerate **(Hard)**

```
1  menu = ['','Americano', 'Latte', 'Espresso', 'Mocha', '식혜', '수정과']
2  price = [0, 1500, 2000, 1700, 2500, 2000, 1900]
3  bill = [0, 50000, 10000, 5000, 1000]
4
5  menu_price = list(zip(menu, price))
```

메뉴 보이기

```
1  def menu_print():
2  #     i = 1
3  #     while i< len(menu):
4  #         print(i, menu[i], price[i])
5  #         #print("%d. %-10s %5d"% (i, menu[i], price[i]))
6  #         i = i+1
7      for idx, val in enumerate(menu_price):
8          if idx != 0:
9              m = val[0]
10             p = val[1]
11             print('%d. %s %5d' %(idx, m, p))
```

[6]

4. Examples

❖ Comprehension & Zip & Enumerate **(Hard)**

음료 선택

```
1  def menu_select():
2      n = int(input("음료를 선택하세요 : "))
3      price_sum = price[n]
4      print(menu[n], price[n],'원 ', '합계 ', price_sum, '원')
5
6      # 음료 추가
7
8      while True:
9          print()
10         n = int(input("계속 주문은 음료 번호를, 지불은 0을 누르세요 : "))
11         if n > 0 and n < len(menu):
12             price_sum = price_sum + price[n]
13             print(menu[n], price[n],'원 ', '합계 ', price_sum, '원')
14         else :
15             if n == 0 :
16                 print("주문이 완료되었습니다.")
17                 break
18             else :
19                 print("없는 메뉴입니다.")
20     return price_sum
```

[6]

4. Examples

❖ Comprehension & Zip & Enumerate **(Hard)**

지불

```
1  def menu_pay(total_price):
2      # 지불 방법 출력
3      [print( i,'.',bill[i],'원',end='    ') for i in range(1, len(bill))]
4  #     for i in range (1, len(bill)):
5  #         print( i,'.',bill[i],'원',end='    ')
6      print()
7
8      # 지불
9      pay = 0
10     while pay < total_price:
11         n = int(input("지불 금액을 입력하세요 : "))
12         if n>0 and n<len(bill):
13             pay = pay + bill[n]
14             print('총 지불액 :', pay,'원')
15         else :
16             print('다시 선택하세요.')
17
18     print('거스름 ', pay-total_price, '원')
```

[6]

4. Examples

❖ Zip & Comprehension **(Medium)**

- Problem: 단어 단위 N-gram 만들기
  표준 입력으로 문자열을 입력하여, 2-gram을 튜플로 출력하시오 (리스트 표현식 사용). [7]
  * N-gram: 문자열에서 연속된 N개 단어들을 추출

Output

```
문자열을 입력하세요: this is python script
[('this', 'is'), ('is', 'python'), ('python', 'script')]
```

4. Examples

❖ Zip & Comprehension (Medium)

```
1  text = input('문자열을 입력하세요: ', )
2
3  words = text.split()
4  two_gram = list(zip(words, words[1:]))
5
6  [i for i in two_gram]
```

[7]

Output

```
문자열을 입력하세요: this is python script
[('this', 'is'), ('is', 'python'), ('python', 'script')]
```

5. References

[1] 박선주. 필수 알고리즘 with 파이썬. 영진닷컴, 2018.
[2] 박응용. Do it! 점프 투 파이썬. 이지스퍼블리싱, 2016년
[3] Ramalho, Luciano. “Fluent Python,” O’Reilly Media, 2015.
[4] Slatkin, Brett. “Effective Python,” Addison-Wesley Professional, 2015.
[5] “백준 15565번”. 백준 온라인 저지. 2019년 2월 20일 접속, https://www.acmicpc.net/problem/15565
[6] 박영준. 파이썬 연습. WikiDocs, 2018.
[7] 남재윤. 파이썬 코딩 도장. 길벗, 2018.

Q&A

Thank You