

Refactoring in Python

소프트웨어 끈대 강의

노기섭 교수

(kafa46@cju.ac.kr)

학습 방향

Study Direction

■ PART I.

- Refactoring Concept

■ PART II.

- Toy Project - Create a Simple Crawler
 - Web page에서 표 크롤링 하기
 - Toy Project: 포켓몬(Pocketmon) 종류별 특성 데이터 추출

■ PART III.

- Refactoring - Practice
 - Toy Project를 활용하여 Refactoring 실습

Toy Project - Create a Simple Crawler

<https://pokemondb.net/pokedex/all>

Refactoring - Practice

Refactoring Approach

■ Divide code in context block

1. Import block
2. Initializing variables (page, doc, and tr_elements) for scraping
3. Extract titles from each of columns in HTML table
4. Extracting contents for each column in HTML table

■ Do factoring one by one

■ Advanced Refactoring Technology

Import Block

■ Import block

- 프로그램 개발에 필요한 모듈/라이브러리 임포트
- 특별히 수정할 사항 없음

```
import requests
import lxml.html as lh
import pandas as pd
import pickle
```

Initializing Variables for Crawling

■ 웹 데이터 크롤링 위한 사전 준비

- 크롤링에 필요한 초기 셋업 후 변수로 생성
- 특별히 Refactoring 필요 없음

```
url='http://pokemondb.net/pokedex/all'  
page = requests.get(url)  
doc = lh.fromstring(page.content)  
tr_elements = doc.xpath('//tr')  
  
temp1 = [len(T) for T in tr_elements[:12]]  
print(f'Lenth of header in first 10 rows: {temp1}')
```


**Extract titles from each of columns
in HTML table**

Extract titles from each of columns in HTML table (1/7)

■ 웹 페이지에 있는 표에서 제목 부분 추출하기

- 전체 코드에서 첫번째 for loop에 해당하는 부분

```
col = []  
  
i = 0  
for t in tr_elements[0]:  
    i += 1  
    name = t.text_content()  
    print('{}: {}'.format(i, name))  
    col.append((name, []))
```

- 변수 i 반복문 밖에서 초기화 → 반복문 내부에서 증가
→ 'enumerate' 기능을 활용해 초기화 가능
→ 적용 결과: 다음 슬라이드 참고

Extracting contents for each column in HTML table (2/7)

■ Refactoring 1. Applying “enumerate”

```
col = []

i = 0
for t in tr_elements[0]:
    i += 1
    name = t.text_content()
    print('{}: {}'.format(i, name))
    col.append((name, []))
```














```
col = []

for i, t in enumerate(tr_elements[0]):
    name = t.text_content()
    print('{}: {}'.format(i, name))
    col.append((name, []))
```

Extracting contents for each column in HTML table (3/7)

■ enumerate

- 다양한 언어에서 지원 (이름, 문법은 약간 다를 수 있음)
 - 반복문의 인덱스 관리를 편리하게 도와 줍니다. → 개발자라면 안 쓸 이유가 없습니다.
 - 코드를 간결하게 해주며, 개발자의 인덱스 직접 관리에 따른 실수를 방지합니다.

	Rust	enumerate	trait.Iterator
	Python	enumerate	-
	Racket	enumerate	list-utils
	D	enumerate	range
	Ruby	with_index	Enumerable
	Kotlin	withIndex	collections
	Elixir	with_index	Enum
	Racket	indexed	data/collection
	Haskell	indexed	Data.List.Index
	Clojure	map-indexed*	core
	C#	Select	Enumerable
	Scala	zipWithIndex	various

이미지 출처: <https://www.youtube.com/watch?v=KTll1MugsSY>

Extracting contents for each column in HTML table (4/7)

■ Refactoring 2. Removing unnecessary statements

- 불필요한 인덱스, 불필요한 화면 출력
 - 반복문 내에서 사용하는 인덱스 i 경우 오직 print 문에서 1번만 사용
 - 아마도 디버깅 목적으로 개발자가 추가한 코드로 생각할 수 있음

최종 코드에서는 삭제하여
코드를 보다 명료/간결하게 유지

```
col = []

for i, t in enumerate(tr_elements[0]):
    name = t.text_content()
    print('{}: {}'.format(i, name))
    col.append((name, []))
```



```
col = []

for t in tr_elements[0]:
    name = t.text_content()
    col.append((name, []))
```

Extracting contents for each column in HTML table (5/7)

■ Refactoring 3. Applying “Comprehension”

- List comprehension: In-place 적용으로 빠르게 list 생성 가능
- “ITM Anti Pattern” 달성 가능

```
col = []  
  
for t in enumerate(tr_elements[0]):  
    name = t.text_content()  
    col.append((name, []))
```



```
col = []  
  
for t in enumerate(tr_elements[0]):  
    col.append((t.text_content(), []))
```



```
col = [(t.text_content(), []) for t in tr_elements[0]]
```

Extracting contents for each column in HTML table (6/7)

■ ITM pattern

Initialize
Then
Modify

```
col = []  
i = 0  
for t in tr_elements[0]:  
    i += 1  
    name = t.text_content()  
    print('{}: {}'.format(i, name))  
    col.append((name, []))
```

```
col = []  
  
for t in enumerate(tr_elements[0]):  
    name = t.text_content()  
    col.append((name, []))
```

■ Anti-pattern ??

- 다음 슬라이드에서 좀 더 자세히 설명

Extracting contents for each column in HTML table (7/7)

■ Anti-pattern (자료 출처: [Wikipedia: Anti-pattern](#))

An **anti-pattern** in [software engineering](#), [project management](#), and [business processes](#) is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.

- 의역해 봤습니다.
 - 소프트웨어공학, 프로젝트관리, 비즈니스 프로세스에서 사용하는 용어
 - 비효과적 문제를 반복적으로 발생시키거나,
 - 매우 비생산적인 위험이 있는 문제에 대한 일반적인 대응

■ Anti-pattern

- 초기에는 문제해결에 적절해 보이지만, 결과적으로 안 좋음(가독성/재활용/생산성 저하)
- ITM anti-pattern
 - ITM을 사용하는 anti-pattern

Extracting contents for each column in HTML table

Extracting contents for each column in HTML table (1/6)

■ Refactoring 4. Removing unnecessary statements

```
for j in range(1, len(tr_elements)):
    T = tr_elements[j]
    if len(T) != 10:
        break
```

```
i = 0
```

```
for t in T.iterchildren():
    data = t.text_content()
    if i > 0:
        try:
            data = int(data)
        except:
            pass
    col[i][1].append(data)
    i += 1
```

불필요한 명령
(특별한 사유가 없다면 삭제)

Extracting contents for each column in HTML table (2/6)

■ Refactoring 5. Using List Slicing

```
for j in range(1, len(tr_elements)):  
    T = tr_elements[j]  
    i = 0  
    for t in T.iterchildren():  
        data = t.text_content()  
        if i > 0:  
            try:  
                data = int(data)  
            except:  
                pass  
        col[i][1].append(data)  
        i += 1
```

변수 T의 목적? Very confusing!!!
tr_elements의 각 요소의
내부 탐색 용도

range 함수 설정, 변수 j 해석이 매우 난해
→ Slicing을 활용하여 Refactoring 수행

다음 슬라이드 참고

Extracting contents for each column in HTML table (3/6)

■ Refactoring 6. Applying “enumerate” <-- Avoid ITM anti-pattern (same as before)

```
for T in tr_elements[1:]:
```

slicing 적용결과 → 명료성, 간결성 증가

```
i = 0
```

```
for t in T.iterchildren():  
    data = t.text_content()  
    if i > 0:
```

ITM pattern 다시 등장!
enumerate 적용

```
try:  
    for T in tr_elements[1:]:  
        for i, t in enumerate(T.iterchildren()):  
            data = t.text_content()  
            if i > 0:  
                try:  
                    data = int(data)  
                except:  
                    pass  
            col[i][1].append(data)
```

Extracting contents for each column in HTML table (4/6)

■ Refactoring 7. Applying Removing “No meaning ‘if’ condition”

- 어떤 환경에서도 전혀 실행되지 않는 조건문 → 삭제하는 것이 바람직

```
for T in tr_elements[1:]:  
    for i, t in enumerate(T.iterchildren()):  
        data = t.text_content()  
        if i > 0:  
            try:
```

어떤 경우도 항상 실행 $\pi\pi$
→ 의미 없음

```
for T in tr_elements[1:]:  
    for i, t in enumerate(T.iterchildren()):  
        data = t.text_content()  
        try:  
            data = int(data)  
        except:  
            pass  
        col[i][1].append(data)
```

Extracting contents for each column in HTML table (5/6)

■ Refactoring 8. Applying Conditional Expression instead of try - except

- try - except 구문 사용 → 가독성(코드 이해) 저하
- 에러 처리 보다는 데이터 형식을 체크 → 조건문 사용
 - 다른 언어의 경우 삼항연산자 $a = b ? c : d$ 를 많이 활용, Python은 if - else로 처리

```
for T in tr_elements[1:]:  
    for i, t in enumerate(T.iterchildren()):  
        data = t.text_content()
```

에러처리?
조건확인?

```
try:
```

```
except:  
    for T in tr_elements[1:]:  
        for i, t in enumerate(T.iterchildren()):  
            data = t.text_content()  
            data = int(data) if data.isnumeric() else data  
            col[i][1].append(data)
```

Extracting contents for each column in HTML table (6/6)

■ Refactoring 9. Reducing the number of redundant variables

```
for T in tr_elements[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        data = int(data) if data.isnumeric() else data
        col[i][1].append(data)
```

반복
변수



```
for T in tr_elements[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        col[i][1].append(int(data) if data.isnumeric() else data)
```

Refactoring 중간 점검

리팩토링 중간 점검

```
21 #####
22 # 포켓몬 표에서 셀 값 추출하기
23 #####
24
25 # 포켓몬 표 주소
26 url='http://pokemondb.net/pokedex/all'
27
28 # 해당 주소에서 URL 정보 가져오기
29 page = requests.get(url)
30
31 # doc 변수에 URL 정보 중 콘텐츠 정보를 doc 변수에 저장
32 doc = lh.fromstring(page.content)
33
34 # HTML의 <tr>.</tr> 사이에 존재하는 데이터 추출 (파싱, parsing)
35 tr_elements = doc.xpath('//tr')
36
37 # 처음 12개 데이터 길이 확인
38 templ = [len(T) for T in tr_elements[:12]]
39 print('Lenth of header in first 10 rows: {templ}')
40 # 각 행(row)은 10개의 데이터로 데이터 보유
41 # 아래와 같은 형태의 출력 확인
42 # 출력 형태 -> [10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
43
44
45 #####
46 # 포켓몬 표에서 컬럼 정보 (table header) 추출하기 #
47 #####
48
49 # 표 제목 정보를 담기 위한 리스트 생성
50 # 각각의 행에서, 첫번째 요소의 이름을 header 이름으로 저장하고
51 # 나중에 각 header 정보를 담기 위한 빈 리스트를 추가
52 col = [] # -> 각 header 세로축 정보를 담기 위한 빈 리스트
53 i = 0 # -> coloum 인덱스
54 for t in tr_elements[0]:
55     i += 1
56     name = t.text_content()
57     print('(): {}'.format(i, name)) # 각 column header 이름 출력 (디버깅 목적)
58     col.append((name, []))
59
60 > ***# 표 제목이 정상적으로 출력되었는지 확인...
61
62 #####
63 # Pandas DataFrame 생성 #
64 #####
65
66 # 첫번째 행은 헤더 정보이므로 별도로 저장하고,
67 # 두번째 행부터 표 데이터가 실제로 저장됩니다.
68 for j in range(1, len(tr_elements)): # -> 첫번째 행 이후 모든 행을 방문
69
70     T = tr_elements[j] # -> 변수 T는 j번째 행을 의미
71
72     # 만약 행의 길이가 10이 아니라면,
73     # 해당 데이터는 우리가 수집한 데이터가 아님
74     if len(T) != 10:
75         break
76
77     i = 0 # -> i는 각 행의 컬럼(column, 표 header) 인덱스
78
79     # 각 행에서 각 열에 해당하는 요소를 방문
80     for t in T.iterchildren():
81         data = t.text_content()
82
83         # 행이 비어 있는지 확인
84         if i > 0:
85             try:
86                 data = int(data) # -> 글자 형태의 숫자 정보가 있다면 정수로 변환
87             except:
88                 pass
89
90         col[i][1].append(data) # -> 추출한 데이터를 i번째 컬럼의 리스트에 추가
91         i += 1 # -> 다음 컬럼으로 이동하기 위해 인덱스 i 값을 1만큼 증가
```

1st Stage of Refactoring

- Reduce lines: 83 → 11 lines!
- More Readable!
- More Maintainable!

```
22 url='http://pokemondb.net/pokedex/all' # 포켓몬 표 주소
23
24 page = requests.get(url) # URL 정보 가져오기
25 doc = lh.fromstring(page.content) # 콘텐츠 정보 저장
26 tr_elements = doc.xpath('//tr') # HTML <tr> 정보
27 col = [(t.text_content(), []) for t in tr_elements[0]] # Column 제목
28
29 # Pandas DataFrame 생성
30 for T in tr_elements[1:]:
31     for i, t in enumerate(T.iterchildren()):
32         data = t.text_content()
33         col[i][1].append(int(data) if data.isnumeric() else data)
```

계속 이어서 Refactoring!

**Extracting contents for each column
in HTML table**

Extracting contents for each column in HTML table (1/7)

■ Refactoring 10. Reducing the complexity of data structure

```
- [(t.text_content(), []) for t in tr_elements[0]]
```

- 각 열 제목과 제목에 해당하는 데이터를 별도의 리스트로 관리하자!
- 리스트 원소로 리스트를 갖는 상황 → 불필요하게 복잡 → 가독성 저하

제목, 내용 리스트를
별도로 분리

```
url='http://pokemondb.net/pokedex/all' # 포켓몬 표 주소

page = requests.get(url)                # URL 정보 가져오기
doc = lh.fromstring(page.content)        # 콘텐츠 정보 저장
tr_elements = doc.xpath('//tr')          # HTML <tr> 정보
col = [(t.text_content(), []) for t in tr_elements[0]] # Column 제목

# Pandas DataFrame 생성
for T in tr_elements[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        col[i][1].append(int(data) if data.isnumeric() else data)

# 크롤링한 정보를 Pandas DataFrame으로 생성
temp_dic = {title:column for title, column in col}
df = pd.DataFrame(temp_dic)
```

결과확인
→ 다음 슬라이드

Extracting contents for each column in HTML table (2/7)

■ Refactoring 10. Reducing the complexity of data structure

→ 적용 결과 확인

```
url='http://pokemondb.net/pokedex/all' # 포켓몬 표 주소

page = requests.get(url)                # URL 정보 가져오기
doc = lh.fromstring(page.content)        # 콘텐츠 정보 저장
tr_elements = doc.xpath('//tr')         # HTML <tr> 정보
titles = [(t.text_content()) for t in tr_elements[0]] # Column 제목

# Pandas DataFrame 생성
cols = [] * len(titles) # 저자의 코딩 실수로 보입니다.
cols = [[] for _ in range(titles)] # 정상 작동
for T in tr_elements[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        col[i].append(int(data) if data.isnumeric() else data)

# 크롤링한 정보를 Pandas DataFrame으로 생성
temp_dic = {title:column for title, column in col}
df = pd.DataFrame(temp_dic)
```

Extracting contents for each column in HTML table (3/7)

■ Refactoring 11. Applying lambda function

- 기능적 반복은 가급적 함수로 작성 → 간단한 입출력인 경우 람다(lambda) 함수 활용

```
url='http://pokemondb.net/pokedex/all' # 포켓몬 표 주소

page = requests.get(url)           # URL 정보 가져오기
doc = lh.fromstring(page.content)   # 콘텐츠 정보 저장
tr_elements = doc.xpath('//tr')    # HTML <tr> 정보
titles = [(t.text_content(), []) for t in tr_elements[0]] # Column 제목

# Pandas DataFrame 생성
cols = [] * len(titles) # 자자의 오타인 듯
cols = [[]] * len(titles) # 메모리 참조 오류 주의
cols = [[] for _ in range(titles)] # 정상 작동

for T in tr_elements[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        cols[i].append(int(data) if data.isnumeric() else data)

# 크롤링한 정보를 Pandas DataFrame으로 생성
temp_dic = {title:column for title, column in col}
df = pd.DataFrame(temp_dic)
```

반복기능
→ 함수로 작성 가능

Extracting contents for each column in HTML table (4/7)

■ Refactoring 11. Applying lambda function

→ 적용 결과 확인

```
url='http://pokemondb.net/pokedex/all' # 포켓몬 표 주소

page = requests.get(url)                # URL 정보 가져오기
doc = lh.fromstring(page.content)        # 콘텐츠 정보 저장
tr_elements = doc.xpath('//tr')         # HTML <tr> 정보
titles = [(t.text_content(), []) for t in tr_elements[0]] # Column 제목

# 데이터 뽑아오는 함수
get_data = lambda data: int(data) if data.isnumeric() else data

# Pandas DataFrame 생성
cols = [[] for _ in range(titles)]
for T in tr_elements[1:]:
    for i, t in enumerate(T.iterchildren()):
        cols[i].append(get_data(t.text_content()))

# 크롤링한 정보를 Pandas DataFrame으로 생성
temp_dic = {title:column for title, column in cols}
df = pd.DataFrame(temp_dic)
```

Extracting contents for each column in HTML table (5/7)

■ Refactoring 12. Applying list comprehension with zip function

```
url='http://pokemondb.net/pokedex/all' # 포켓몬 표 주소

page = requests.get(url)                # URL 정보 가져오기
doc = lh.fromstring(page.content)        # 콘텐츠 정보 저장
tr_elements = doc.xpath('//tr')          # HTML <tr> 정보
titles = [(t.text_content(), []) for t in tr_elements[0]] # Column 제목

# 데이터 뽑아오는 함수
get_data = lambda data: int(data) if data.isnumeric() else data

# Pandas DataFrame 생성
cols = [[] for _ in titles] # 정상 작동
for T in tr_elements[1:]:
    for i, t in enumerate(T.iterchildren()):
        cols[i].append(get_data(t.text_content()))

# 크롤링한 정보를 Pandas DataFrame으로 생성
temp_dic = {title:column for title, column in cols}
df = pd.DataFrame(temp_dic)
```

list comprehension
+ zip 활용하여 처리 가능

Extracting contents for each column in HTML table (6/7)

■ Refactoring 12. Applying list comprehension with zip function

→ 실행 결과 확인

```
url='http://pokemondb.net/pokedex/all' # 포켓몬 표 주소

page = requests.get(url)                # URL 정보 가져오기
doc = lh.fromstring(page.content)        # 콘텐츠 정보 저장
tr_elements = doc.xpath('//tr')         # HTML <tr> 정보
titles = [(t.text_content(), []) for t in tr_elements[0]] # Column 제목

# 데이터 뽑아오는 함수
get_data = lambda data: int(data) if data.isnumeric() else data
cols = list(zip(*[get_data(t.text_content()) for t in T.iterchildren()]
                for T in tr_elements[1:])))

# 크롤링한 정보를 Pandas DataFrame으로 생성
temp_dic = {title:column for title, column in cols} # 저자 코드
temp_dic = dict(zip(titles, cols)) # 좀 더 간결한 코드
df = pd.DataFrame(temp_dic)
```


Extracting contents for each column in HTML table (7/7)

■ Refactoring 13. More concise expression in Dictionary generation.

```
url='http://pokemondb.net/pokedex/all' # 포켓몬 표 주소

page = requests.get(url)                # URL 정보 가져오기
doc = lh.fromstring(page.content)        # 콘텐츠 정보 저장
tr_elements = doc.xpath('//tr')         # HTML <tr> 정보
titles = [(t.text_content(), []) for t in tr_elements[0]] # Column 제목

# 데이터 뽑아오는 함수
get_data = lambda data: int(data) if data.isnumeric() else data
cols = zip(*[[get_data(t.text_content()) for t in T.iterchildren()]
              for T in tr_elements[1:]])

# 크롤링한 정보를 Pandas DataFrame으로 생성
temp_dic = {title:column for title, column in zip(titles, cols)} # 저자 코드
temp_dic = dict(zip(titles, cols)) # 좀 더 간결한 코드
df = pd.DataFrame(temp_dic)
```

Recommendations

Recommendations on Refactoring

■ Know your algorithm

- 여러분이 만든 프로그램의 알고리즘(작동 원리)을 이해하세요.

■ Know your collections

- 여러분의 처리하고 있는 데이터 모음을 이해하세요. (예, 숫자 모음, 문자열 모음, ...)
- 그리고 적절히 사용할 수 있는 Container Data Type을 공부해 두세요.
 - 예) 리스트, 튜플, 사전, 큐, ...

■ Know your libraries

- 여러분이 사용할 수 있는 라이브러리에 대해 알고 있어야 합니다.
- 적절하게 사용할 수 있는 라이브러리를 알고 있다면,
 - 여러분의 야근과 삽질을 많이 줄일 수 있습니다.



이미지 출처:
<http://egloos.zum.com/epic/v/1940802>