Project myre

신승우

Python Open Source Study

1 사전지식

1.1 정규표현식 (Regular Expression)

정규표현식이란? 정규표현식은 특정한 형식을 가진 문자열로써, 다른 문자열들 여러 개를 표현하는데 쓰인다. 조금 더 수학적으로 말하면, 정규표현식은 문자열의 집합을 나타내는 형식이라고 볼 수도 있다.

정규표현식을 쓰는 이유 문자열을 다루다 보면 특정 조건을 만족하는 문자열인지를 체크해야 할 필요가 있는 경우가 많다. 예를 들어서, 날짜를 나타내기 위해서는 연, 월, 일을 나타내야 하며, 이들 각각은 4자리, 2자리, 2자리의 숫자로 나타낼 수 있다[1]. 따라서 이러한 조건을 만족하는지를 체크하는 함수를 다음과 같이 작성할 수 있다.

Implementation 1: *check dates (checkdate.py)*

```
1 def check_date(text):
2     for char in text:
3         assert char in ['0', '1', '2', '3', '4', \
4             '5', '6', '7', '8', '9',]
5     assert len(char) == 8
```

위와 같은 체크 방법에는 크게 두 가지 문제가 있다.

- 매번 함수를 새로 작성해야 한다. 이는 번거롭기도 하고, 비슷한 패턴들에 대해서도 각각 함수를 다 작성해야 하는 것 자체가 불필요한 일일 수 있다. 예를 들어서, 날짜를 20190101과 같은 형태로 나타내는 것과, 2019-01-01과 같이 나타내는 것은 사실상 매우 비슷한 패턴이지만, 위 check_date와 같이 함수를 작성하면 이를 재활용하기가 어렵다.
- 또 위 체크 방법은 틀렸다. 예를 들어서, 저런 방식의 경우 20191040같은 것도 날짜로 보겠지만, 이는 잘못된 날짜이다. 따라서 각 문자가 숫자임을 체크하는 것으로는 충분하지 않다. 이는 많은 경우에 그러한데, 예를 들어서 주민등록번호라면 뒷자리의 첫 번째 숫자가 1 혹은 2여야 한다. 따라서 조금 더 세밀하게 문자열을 검사할 필요가 있다.

따라서 위 두 문제를 해결하기 위해서 다음과 같이 생각할 수 있다.

- 많이 쓰이는 가장 atomic한 패턴을 정의한 후, 이를 체크하는 함수들을 미리 짜놓는다. 예를 들어서, 앞에서 나온 날짜를 쓰는 방법 두 가지는 모두 어떤 문자열이 숫자인지를 체크하는 과정이 필요하다. 이를 함수로 미리 정의해놓고 재사용한다면 더 편하게 구현할 수 있을 것이다.
- 문자열을 체크하는 함수를 구현할 때 다양한 인자를 받을 수 있도록 한다. 예를 들어서, 위 날짜의 경우 비단 숫자인지만 체크하는 것이 아니라 어떤 숫자인지 역시 체크할 수 있도록 하면 편할 것 같다. 그렇게 하는 것으로 월은 1월부터 12월까지만 가능하게 하는 식으로 할 수 있도록 함수를 유연하게 구현하면 된다.

[1] 1월 같은 경우는 01과 같이, 0을 앞에 넣어 두 자리로 고정하는 것으로 생각하자.

먼저 첫 번째 해결책부터 적용해보자. 다음과 같이 생각할 수 있다.

Implementation 2: *check_numeric 함수 추가 (checkdate.py)*

```
def check_numeric(char):
    assert char in ['0', '1', '2', '3', '4', \
        '5', '6', '7', '8', '9',]

def check_date(text):
    for char in text:
        assert check_numeric(char)
```

check_numeric 함수를 이용하면 숫자인지를 편하게 체크할 수 있고 다른 형식의 패턴을 체크할 때도 광범위하게 쓰일 수 있으므로 재사용성을 늘릴 수 있을 것 같다. 그렇다면 이번에는 두 번째 문제를 해결해 보자.

Implementation 3: *check_numeric 함수 추가 (checkdate.py)*

```
def check_numeric(char, start, end):
    assert char in [str(e) for e in range(start, end+1)]

def check_date(text):
    assert check_numeric(text[0], 0, 9)
    assert check_numeric(text[1], 0, 9)
    assert check_numeric(text[2], 0, 9)
    assert check_numeric(text[3], 0, 9)
    if check_numeric(text[4], 0, 0):
        assert check_numeric(text[5], 0, 9)
    elif check_numeric(text[4], 1, 1):
        assert check_numeric(text[5], 0, 1)
    assert check_numeric(text[6], 0, 3)
    if check_numeric(text[6], 0, 2):
        assert check_numeric(text[7], 0, 9)
    elif check_numeric(text[4], 3, 3):
        assert check_numeric(text[7], 0, 9)
```

이와 같이 하면 대부분의 숫자 관련 문자열 점검에서 check_numeric 함수를 사용할 수 있을 것이라고 생각된다. 하지만, 이 역시 상당히 길고 복잡하다고 볼 수 있다. 왜냐하면 사실상 check_numeric 함수에서 쓰이는 것은 start와 end 두 숫자만 쓰이기 때문에, 이를 잘 wrapping하는 형식을 만들면 숫자 두 개만 가지고 check_numeric 함수를 대체할 수 있을 것이라고 기대할 수 있다. 예를 들어서, [n-m] 같은 형식의 문자열을 파싱하여 위와 같은 함수를 자동으로 생성할 수 있다면 매우 편리할 것이다.

이를 위해서 많이 쓰이는 패턴들을, 매번 함수를 작성하는 것이 아니라 특정 형식을 가진 문자열로써 편하게 나타내어 이 문자열을 파싱하여 자동으로 내가 의도한 패턴을 체크하는 함수를 생성하는 것이 정규표현식이라고 할 수 있다.

2 구현 진행

10.28 구현사항 아래의 함수 2개를 구현해본다.

- check(pattern : str) → boolean : 주어진 문자열 pattern이 정규표현식이 맞으면 True, 틀리면 False를 반환.
- exact_match(pattern: str, text: str) → boolean : 주어진 정규표현식 pattern과 문자열 text에 대해서, 문자열 text가 pattern이 나타내는 문자열의 집합에 포함되었다면 True. 아니라면 False. 만약 pattern이 정규표현식이 아니라면 ValueError를 raise한다.

check check 함수는 주어진 문자열이 정규표현식이 맞는지를 체크하여, 맞으면 True, 틀리면 False를 반환하는 함수이다. [2]

exact_match exact_match 함수는 주어진 패턴과 텍스트가 일치하는지를 검사한다. 패턴은 정규표현식 문법을 따르는 문자열이고, text는 일반 문자열이다. 이 때, 패턴이 따라야 하는 스펙은 아래와 같다. 아래 스펙에서 expr은 정규표현식 스펙을 만족하는 문자열을 의미한다. [3]

패턴	설명	예시	비고
알파벳	(a)	a, b, ...	우선은 한글을 다루지 않기로 함.
숫자	(a)	0,1,..	
공백	(a)		
특수문자들	(e)	(e)	
\+ 특수문자들	(a)		(d)
expr1 expr2	(b)	(c)	(d)
expr1 | expr2	(f)	(g)	

(a) 패턴이 나타내는 문자열의 집합은 패턴 그 자체이다. 예를 들어서, 패턴이 '1'이라면 나타내는 문자열도 '1' 하나이다.

(b) 식 expr1, expr2를 만족하는 문자열이 연결되어있음을 나타냄.

(c) ab의 경우, a도 정규표현식이며 b도 정규표현식이므로 ab를 이어놓은 것 역시 정규표현식이다.

(d) 보기 편하기 위해서 가운데에 공백이나 + 기호가 있으나, 실제로 공백이 있는 것은 아니다.

(e) 정규표현식에서는 다양한 특수문자들을 제공한다. 우선 여기서는 .만을 다루기로 한다.

- . : wildcard. 1개의 임의의 문자에 대응된다. 예를 들어서, a. 은 a로 시작하고, 뒤에 아무 글자나 하나 붙어 있는 문자열들을 전부 나타낸다. 예를 들어서, ab는 a 뒤에 b라는 글자가 하나 있으므로 이는 a.과 match된다. 다른 예로, abc는 a뒤에 b,c 두 글자가 있으므로 match 되지 않는다.

(f) expr1과 match되는 문자열이거나 expr2과 match되는 문자열.

(g) a|b의 경우 a혹은 b를 나타냄. 따라서 a, b 둘 다 match됨.

[2] 파이썬에 내장된 함수는 아니지만, 정규표현식을 구현하기 위해서는 이러한 체크 함수가 있어야 하기에 추가하였음

[3] 이 역시 파이썬에 있는 함수는 아니지만, 본격적인 구현 전에 구현해놓으면 앞으로 계속 쓸 수 있기 때문에 구현하면 좋을 것으로 생각됨.