Data Analytics 아카데미
정규표현식

목차

1. 학습목표
2. Regex로 회계감사에서 할 수 있는 것
3. Part I – RegEx 기초
4. Part II - RegEx 응용
5. 결론

1. 학습 목표

- **Regex를 활용하여 감사의 효과성과 효율성을 높인다.**
  - ✓ **문자열 검색**
  - ✓ **데이터 형식 변경**

Reference

- **https://www.w3schools.com/python/python_regex.asp**
- **https://docs.python.org/ko/3/howto/regex.html**

Housekeeping

- **Jupyter lab 3.0.14를 이용해 진행할 것임 (KICPA_ADA_Regex_Lecture_Note.ipynb파일 참고)**
- **Python 3.8.10, pandas 1.3.0 이용함.**
- **Python 설치 및 기초 활용 방법에 대해서는 다른 강의 참고할 것.**

2. Regex로 회계감사에서 할 수 있는 것

- 전표 적요 내 검색
  - 대표이사관련(회장, CEO, 사장, 대표…), 오류(수정, 정정, 에러)
- 데이터 클렌징
  - 날짜 형식 변환

3. Part I – RegEx 기초

3.1. 정규표현식(Regular Expression)이란?

- 검색대상 문자의 패턴을 표현하는 문자들의 조합
- 정규표현식(regular expression), 줄여서 정규식(regex)라고도 부른다.
- 다른 프로그래밍 언어에도 유사한 형태의 정규식이 구현되어 있다. 우리는 python의 정규식을 다룰 예정
- Python의 내재(built-in) 모듈로 별도 설치가 불필요
- 정규식 패턴은 bytecode로 compile된 후 matching engine에 의해 실행된다.
- 작고 빠르지만, 모든 경우의 패턴을 다 표현할 수 있는 것은 아니고, 너무 복잡해질 때도 있다. ➔ python 코드를 쓰는 것이 실행에는 느리지만, 이해하기 쉬울 수도 있다.
- 활용사례
  - 검색엔진
  - Cyber security – 악성코드 탐지
  - Data cleansing – 형변환, noise 제거

수업 환경 구성하기

- **수업 환경 파일이 KICPA_ADA.yml 로 제공되었음.**
- **Anaconda에서 yml 파일 설치하여 환경 구성하기**

```
conda env create –f KICPA_ADA.yml
```

- **만들어진 환경 보기**

```
conda env list
```

- **만들어진 환경으로 전환하기**

```
conda activate py3.8
```

- **Jupyter lab 실행하기**

```
jupyter lab 또는 jupyter-lab
```

3.2 RegEx 라이브러리 불러오기

➢ RegEx 라이브러리 불러오기

```
import re
```

- RegEx 라이브러리를 사용하기 위해서는 사용자가 개발하고 있는 소스 코드에 불러들여야 한다.
- 다른 라이브러리와 달리 이름이 짧기 때문에 별명(alias)를 지정하지 않는다.

3.3 RegEx Functions/methods

➢ **문자열 패턴에 사용할 수 있는 주요 함수/메소드**

함수명	형식	개요
match()	.match(pattern, string)	string이 pattern으로 시작하면 발견된 substring의 정보 포함된 객체 생성
search()	.search(pattern, string)	pattern이 string에 있는지 확인. 다수 발견 시 처음 발견된 것의 위치 정보 포함된 객체 생성
findall()	.findall(pattern, string)	모든 match 결과 반환. 다수 발견 시 발견된 모든 것의 [리스트] 반환
split()	.split(pattern, string)	pattern을 구분자(delimeter)로 보고 나눠진 substring의 [리스트] 반환
sub()	.sub(pattern, repl, string)	대체(repl)된 문자열 반환

RegEx의 객체 (object)

- **Pattern object**

```
import re
p = re.compile('까까오')
```

- **Match object**

```
import re
m = p.match(적요str) 또는 p.search(적요str)
```

함수(?) vs. 메소드(?)

- re에서는 함수를 바로 호출할 수도, compile하여 pattern 객체를 생성한 후 메소드를 부를 수도 있다.
- re.search(pattern, string) vs. p.search(string) 중 어느 것을 써야 할까?
- 함수 호출 시에도 pattern object를 생성하여 캐시(cache) 에 저장하므로 본질은 동일하다.
- 루프를 돌릴 때는 pattern 객체 사용이 조금 낫고, 그 이외에는 별 차이 없다.

3.4 메타문자 (Metacharacters)

➢ **메타문자는 특별한 의미를 가진 문자들이다.**

메타문자	개요	예시
[ ]	[ ]안에 포함된 문자들의 집합	다음 장에서 살펴보자
\	special sequence 의 신호	다음 장에서 살펴보자
.	모든 글자 (행변환 제외)	.
^	시작	^뉴욕
$	끝	사장님$
*	없거나 여러 번 있거나	사*장님*
+	한번 이상 있는 경우	.장님+
?	없거나 한 번이거나(0 or 1)	
{ }	{ } 안에 표시한 숫자 만큼 있는 경우	대표.{2}
|	또는	사장님|회장님|대표이사
( )	( )안 글자의 그룹	(대표.{2}|.장님)+

➢ **아래의 적요에 위의 예시를 적용한다면 어떻게 될까?**

적요1 = ''뉴욕 출장 숙박비 100만원, 식대 100만원 - 까까오뱅크 대표이사, 까까오페이 회장님, 까까오커머스 사장님"

3.5 집합(Set)

➢ [ ] 안에 포함된 문자들의 집합

	개요	예시
[가나다]	[ ]안에 포함된 문자들의 집합	[까오] – 까 또는 오
[가-다]	- 앞의 문자부터 가나다 순으로 – 뒤의 문자까지	[까-오] – 까 부터 오까지
[^가]	^다음의 문자를 제외한 모든 문자	[^까오]
[012]	숫자 0, 1,2의 집합	[10]
[0-9]	0부터 9까지	[4-9]
[메타문자]	[ ] 안의 메타문자는 의미가 없고, 그 문자 자체를 의미	[까+] vs. 까+

➢ **아래의 적요에 위의 예시를 적용한다면 어떻게 될까?**

적요1 = "뉴욕 출장 숙박비 100만원, 식대 100만원 - 까까오뱅크 대표이사, 까까오페이 회장님, 까까오커머스 사장님"

3.6 Special Sequence

➢ \ 이후에 아래 문자들이 이어지는 것으로 특별한 의미가 있다.

	개요
\d	숫자(digit) 포함
\D	숫자 불포함. \d의 반대
\s	공백(space) 포함
\S	공백 불포함. \S의 반대
\w	숫자와 글자(word) 포함
\W	Word 불포함.

➢ 아래의 적요에 위의 예시를 적용한다면 어떻게 될까?

적요1 = ''뉴욕 출장 숙박비 100만원, 식대 100만원 - 까까오뱅크 대표이사, 까까오페이 회장님, 까까오커머스 사장님”

4. Part II - RegEx 응용

Logistics

- 지금부터 Regex를 이용하여 전표 적요를 검색하고, 날짜 형식 바꾸는 방법을 실습할 것이다.
- 수업에 이용할 데이터는 총계정원장_Regex.txt 이다.
- 총계정원장_Regex.txt – 구분자(#|#), 컬럼헤드가 첫 번째 행에 위치

4.1 총계정원장 불러 들이기

- 테이블 형태 데이터 가공에 특화되어 있는 Pandas library를 이용할 것이다.
- 자료 요청 시 포함되어야 할 컬럼리스트 및 데이터 타입(형식), 구분자, 컬럼헤더 포함 여부등이 명확히 커뮤니케이션 되어야 한다.
- 컬럼 헤더가 첫 번째 행(row index로는 0)에 포함되어 있음을 명시할 수 있으나, 안하면 자동 유추한다.

```
import pandas as pd

df_GL = pd.read_csv("dataset/총계정원장_regex.txt", sep='#₩|#', engine='python')

df_GL
```

4.2 전표 적요 검색하기

- 다양한 임원 명칭이 전표 적요에 포함되었는지 검색한다.
- 적요에서 '대표, CEO, 회장, 사장, VIP 등이 포함되었는지 검색하고 발견되면 전표번호를 도출한다.

```
import re
pattern = '대표|CEO|회장|사장|VIP'

for i in range(len(df_GL)):
    result = re.findall(pattern, str(df_GL.loc[i,'적요']))
    if result:
        print(df_GL.loc[i,'전표번호'], result)
```

4.2 전표 적요 검색하기

- 오류 수정 분개를 검색한다.
- 적요에서 오류, 정정, 에러, 수정, error 등이 포함되었는지 검색하고 발견되면 전표번호를 도출한다.

```
import re
pattern = '오류|정정|에러|수정|error'

for i in range(len(df_GL)):
    result = re.findall(pattern, str(df_GL.loc[i,'적요']))
    if result:
        print(df_GL.loc[i,'전표번호'], result)
```

4.3 데이터 클렌징

- 날짜 형식이 다르게 표시된 것을 찾아 통일시켜 준다.
- 거래일 컬럼에서 m/dd/yyyy로 된 날짜를 yyyy-mm-dd로 바꿔준다.
- 거래일 컬럼을 str 형식에서 날짜형식으로 변환시켜준다.

```
import re
p = re.compile('₩s?₩d{1,2}/₩s?₩d{2}/₩s?₩d{4}')

r = []
for 거래일 in df_GL['거래일']:
    #print(거래일)
    s = p.findall(거래일)
    #print(s)
    if s:
        r.append(s)
```

5. 결론

정규식은 search engine이다!

What we can do!

- 대용량 텍스트문서에서 원하는 정보를 빨리 찾을 수 있다.
- 데이터 형식 변환을 용이하게 할 수 있다.

What we need to be cautious!!

- Regex 구성자의 기대 수준을 벗어날 수 없다.
- 회사 제시 데이터를 변환하는 과정은 challenge 를 받을 수 있으니, 적절한 문서화가 필요하다.
- SQL에도 regex가 있으니, 정말 큰 데이터라면 SQL로 접근하자.