한글 인코딩 구조와 그에 따른 파이썬 정규식

Xero
박준혁 (한국디지털미디어고등학교 1 학년)
2011-11-05
wnsgurzxc@nate.com

목차

0. 잡담 - 3
1. 한글 인코딩 구조 - 4
2. EUC-KR 와 그에 따른 정규식 - 5
3. CP949 와 그에 따른 정규식 - 6
4. UTF-8 와 그에 따른 정규식 - 7
5. 후기 - 10

0. 잡담

필자는 기숙형 학교에 다니고 있다. 기숙형 학교인 만큼 학교에서 세 끼를 해결한다. 그러므로 많은 학생들이 식단을 궁금해 한다. 그러나 급식소까지 가서 식단을 보기에는 무리라고 생각, 홈페이지의 식단을 파싱하여 프로그램을 만들어 식단 보기의 불편함을 해소하기로 했다. 만들던 도중, 학교 홈페이지가 utf-8 인코딩을 사용하는 것을 발견, 보통의 정규식과는 달라서 애를 먹었다.
검색해도 나오지 않기에 직접 한글 인코딩 구조를 파악하고 정규식을 만들었다.
이 문서는 내가 알게 된 정보들의 총 정리로써 인코딩 구조들과 그에 따른 한글 정규식 정보들을 담고 있다.

1. 한글 인코딩 구조

한글 인코딩 구조에는 다음과 같은 종류가 있다.

- 한글 완성형 인코딩
- 한글 상용 조합형 인코딩
- 한글 조합형 인코딩

1. 한글 완성형 인코딩

한글 완성형 인코딩 또는 완성형은 한글을 그 구조와 관계 없이 코드를 배당하여 표현하는 문자 인코딩들을 총칭하는 말이며, 흔히 한글 조합형 인코딩과 비교된다.

2. 한글 상용 조합형 인코딩

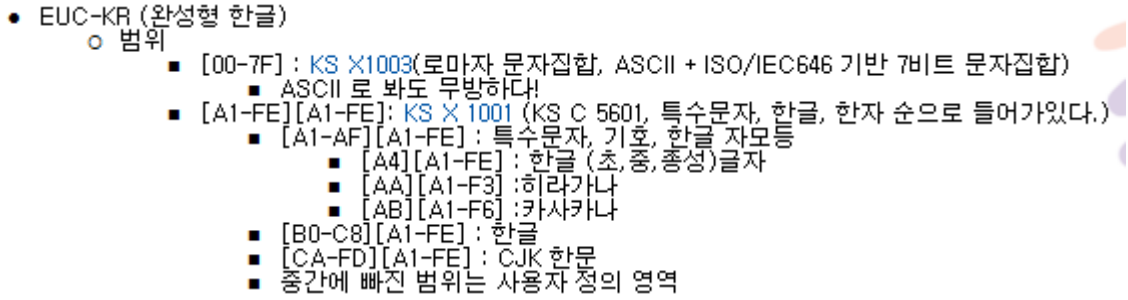
한글 조합형 인코딩 또는 조합형은 한글을 그 낱자에 따라 기계적으로 조합하여 표현하는 문자 인코딩들을 총칭하는 말이다. 1980 년대 초반부터 1990 년대 중반까지 널리 사용되었다.

3. 한글 조합형 인코딩

한글 조합형 인코딩 또는 조합형은 한글을 그 낱자에 따라 기계적으로 조합하여 표현하는 문자 인코딩들을 총칭하는 말이다. 1980 년대 초반부터 1990 년대 중반까지 널리 사용되었다.

2. EUC-KR 와 그에 따른 정규식

EUC-KR 은 KS X 1001 와 KS X 1003 을 사용하는 8 비트 문자 인코딩으로, EUC 의 일종이며 대표적인 한글 완성형 인코딩이기 때문에 보통 완성형이라고 불린다.
EUC-KR 의 범위는 다음과 같다.



ASCII 범위는 [00-7F] 이고, 한글의 범위는 [A1-FE][A1-FE] 이다.
그러므로 다음과 같이 정규식을 작성하면 된다.
sAnswer=re.findall('[₩xa1-₩xfe][₩xa1-₩xfe],sProb)

다음과 같이 euc-kr 으로 인코딩 셋을 지정하고 위의 정규식으로 한글을 추출하는 코드를 짜 보았다.

```
# -*- coding: euc-kr -*-
import re
sProb='한글'
sAnswer=re.findall('[\xa1-\xfe][\xa1-\xfe]',sProb)
print sAnswer, ''.join(sAnswer)
```

실행시키면 다음과 같이 성공적으로 한글이 추출된다.

```
>>>
['\xc7\xd1', '\xb1\xdb'] 한글
```

정리하자면 EUC-KR 에서의 한글 패턴은 다음과 같다.
[₩xa1-₩xfe][₩xa1-₩xfe]

3. CP949 와 그에 따른 정규식

코드 페이지 949(CP949)는 마이크로소프트사가 도입한 코드 페이지이다. 본래는 KS C 5601 의 완성형 한글을 표현한 코드 페이지였으나, 윈도 95 부터는 확장 완성형 혹은 통합형 한글 코드(Unified Hangul Code)이라는 명칭으로 확장되어 현대의 모든 한글을 수용하게 되었다. 마이크로소프트에서는 이 인코딩을 기반 문자 집합

이름인 "ks_c_5601_1987"로 사용하고 있다. 다만 이 코드 페이지는 IANA 에 등록되어 있지 않으므로 인터넷 상에서 정보를 주고받는 데 대한 표준은 아니다.

- CP949 (확장 완성형 한글)
  - 범위
    - [00-7F] : KS X1003(로마자 문자집합, ASCII + ISO/IEC646 기반 7비트 문자집합)
      - ASCII 로 봐도 무방하다!
    - [A1-FE][A1-FE]: KS X 1001 (KS C 5601, 특수문자, 한글, 한자 순으로 들어가있다.)
      - [A1-AF][A1-FE] : 특수문자, 기호 등
        - [A4][A1-FE] : 한글 (초,중,종성)글자
        - [AA][A1-F3] :히라가나
        - [AB][A1-F6] :카사카나
      - [B0-C8][A1-FE] : 한글
      - [CA-FD][A1-FE] : CJK 한문
      - 중간에 빠진 범위는 사용자 정의 영역 등
    - [81-A0][41-5A,61-7A,81-FE] : 확장 한글
    - [A1-C5][41-5A,61-7A,81-A0] : 확장 한글
    - [C6][41-52] : 확장 한글

CP949 또한 EUC-KR 과 같이 ASCII 의 범위가 [00-7F]이고 한글의 범위가 [A1-FE][A1-FE] 인 것을 확인할 수 있다.
그러므로 CP949 또한 EUC-KR 과 같이 정규식을 작성하였다.
sAnswer=re.findall('[₩xa1-₩xfe][₩xa1-₩xfe],sProb)

다음과 같이 cp949 로 인코딩 셋을 지정하고 한글을 추출하였다.

```
# -*- coding: cp949 -*-
import re
sProb='한글'
sAnswer=re.findall('[\xa1-\xfe][\xa1-\xfe]',sProb)
print sAnswer, ''.join(sAnswer)
```

다음과 같이 정상적으로 추출된 것을 확인할 수 있다.

```
>>>
['\xc7\xd1', '\xb1\xdb'] 한글
```

CP949 에서의 한글 패턴은 다음과 같다.
[₩xa1-₩xfe][₩xa1-₩xfe]

4. UTF-8 와 그에 따른 정규식

UTF-8 은 유니코드를 위한 가변 길이 문자 인코딩 방식 중 하나로, 켄 톰프슨과 롭파이크가 만들었다. 본래는 FSS-UTF(File System Safe UCS/Unicode Transformation)라는 이름으로 제안되었다.
이 문서에서는 한글 인코딩 구조에 따른 정규식을 주로 다루기 때문에 인코딩 구조에 대해서는 상세히 설명하지 않겠다.
UTF-8 에 대한 자세한 정보는 다음을 참고하면 좋다.
http://ko.wikipedia.org/wiki/UTF-8

다음은 처음에 UTF-8 에 시도했던 정규식이다.
EUC-KR 과 같은 한글 범위로 정규식을 시도하였다.

```
# -*- coding: utf-8 -*-
import re
sProb='한글'
sAnswer=re.findall('[\xa1-\xfe][\xa1-\xfe]',sProb)
print sAnswer, ''.join(sAnswer)
```

그러자 다음과 같이 이상한 문자만이 추출되었다.

```
>>>
['\xea\xb8'] 湲
```

해결방법을 찾다가 UTF-8 코드표를 보았다.
다음은 UTF-8 코드표 주소이다.
http://titus.uni-frankfurt.de/unicode/unitestx.htm

우선 utf-8 로 인코딩 셋을 지정하고 'ㄱ' 이 어떻게 인코딩 되어 있는지 보기 위해 출력하였다.

```
# -*- coding: utf-8 -*-
import re
sProb='ㄱ'
print sProb, repr(sProb)
```

다음과 같이 ₩xe3₩x84₩xb1 이라는 값이 나왔다.

```
>>>
ㄱ '\xe3\x84\xb1'
```

위의 16 진수 값을 10 진수로 바꾸니 다음과 같은 값이 나왔다.
227,132,177

UTF-8 코드표를 참조하면 다음과 같이 227,132,177 값에 'ㄱ' 이 있는 것을 볼 수 있다.

		0	1
UTF8: 227, 132, 128;	UNICODE: 310	□	□
UTF8: 227, 132, 144;	UNICODE: 311	닁	닀
UTF8: 227, 132, 160;	UNICODE: 312	么	又
UTF8: 227, 132, 176;	UNICODE: 313	□	ㄱ

UTF-8 코드표에서 자음과 모음이 있는 곳의 값은 다음과 같다.

UTF8: 227, 132, 176;
UTF8: 227, 133, 128;
UTF8: 227, 133, 144;
UTF8: 227, 133, 160;
UTF8: 227, 133, 176;
UTF8: 227, 134, 128;

첫 값은 227, 둘째 값은 132~134, 셋째 값은 128~191 (176+15) 사이이다.
그러므로 다음과 같은 범위라는 것을 알 수 있다.
₩xe3[₩x84-₩x86][₩x80-₩xbf]

가~힣 까지의 글자들의 범위를 찾으면 다음과 같은 범위를 구할 수 있다.
[₩xea-₩xed][₩x80-₩xbf][₩x80-₩xbf]

위의 두 범위를 | 로 or 처리하여 다음의 정규식을 만들었다.
sAnswer=re.findall('₩xe3[₩x84-₩x86][₩x80-₩xbf]|[₩xea-₩xed][₩x80-₩xbf][₩x80-₩xbf]',sProb)

UTF-8 로 인코딩 셋을 지정하고 같이 코딩하였다.

```
# -*- coding: utf-8 -*-
import re
sProb='한글'
sAnswer=re.findall('\xe3[\x84-\x86][\x80-\xbf]|[\xea-\xed][\x80-\xbf][\x80-\xbf]',sProb)
print sAnswer, ''.join(sAnswer)
```

그러면 다음과 같이 한글이 정상적으로 추출된다.

```
>>>
['\xed\x95\x9c', '\xea\xb8\x80'] 한글
```

UTF-8 에서의 한글 범위는 다음과 같다.
₩xe3[₩x84-₩x86][₩x80-₩xbf]|[₩xea-₩xed][₩x80-₩xbf][₩x80-₩xbf]

5. 후기

이번 기회로 인해 한글 인코딩 구조에 대해 자세히 알게 되었다.
UTF-8 인코딩 문제 때문에 엄청 많은 시간을 소비하여 공부하였다.
많은 시간을 들여 파고든 만큼 더더욱 기억에 남을 것이다.
혹시나 나처럼 한글 인코딩 구조 때문에 시간 허비하는 사람들이 있을까 해서 문서를 쓴다.
이 문서를 봄으로써 한글 인코딩 구조를 파악하는 시간을 줄이고 좀 더 쉽게 이해했으면 한다.