

## 파이썬 한글 인코딩(UTF8, Unicode, Euc-Kr) 탐구

2017.06.30 23:39 :: 프로그래밍/도구제작 & 개발

안녕하세요. **Message**입니다.

한동안 큐브리드와 파이썬에서 인코딩 문제로 골머리를 썩었습니다.

특히 아래 오류는 이제 보기만 해도 지긋지긋하네요.

**UnicodeDecodeError : Ascii codec can't decode byte 0xea in position 0: ordinal not in range(128)**

인코딩이라는 주제가 처음엔 이해가 되는가 싶어도 검색 시간이 늘어날수록 혼란이 늘어나더군요.

문제를 해결했다고 해도, 일주일만(사실 하루이틀..) 지나면 까먹고선 똑같은 검색어로 구글링하고 있습니다.

지난주에 봤던 블로그가 어디더라 하면서 즐겨찾기를 하지 않은 스스로를 원망합니다 ㅜ

짧게라도 필요했던 부분을 정리해서 포스팅하겠습니다. 인코딩의 늪에 빠져 스트레스 받고 계시는 분들에게 도움이 되길 바랍니다.

### 1. 그래서 "인코딩" 이라는게 무엇이고

사실 IT 분야에서 Ascii, Unicode, UTF8, Base64, URL 등 인코딩과 관련된 단어는 많이 보고 들을 수 밖에 없기 때문에 책에서 스치듯 읽은 내용이나 주위들은 지식만으로 문자 인코딩을 알고있다고 생각했습니다.

하지만 인코딩에 대한 정확한 이해가 없다보니 읽어들인 Hex값이 뭘 의미하는지,

어떤 방식으로 인코딩/디코딩을 해야 내가 원하는 문자가 화면에 딱 나오는지 알수가 없어 막막하더군요.

이번 기회에 제가 주관적으로 이해한 인코딩에 대해 적어보겠습니다.

#### 1) 정의 살펴보기

일단 가장 객관적인 인코딩의 정의를 보고 넘어갑시다.

**"특정한 문자 집합 안의 문자들을 컴퓨터 시스템에서 사용할 목적으로 일정한 범위 안의 정수들로 변환하는 방법"**

분명 나는 IT를 전공했고, 한국인임에도 불구하고 이게 무슨말인지 이해가 안갑니다.

일단 지금 당장은 한글 UTF8이 3바이트로 저장된대거나 유니코드의 한글 범위가 어디서 어디까지다..라는

상세한 내용은 접어두고 인코딩의 원리를 이해하는데 초점을 맞추어 하나씩 실습해보겠습니다.

#### 2) 직접 테스트하기

변수에 "안녕"이라는 한글 문자열을 UTF8 인코딩으로 저장하면 어떤 바이너리 형태로 저장될까요?

Python IDLE을 이용하여 확인해봅시다. "Wx" 문자가 붙으면서 16진수 형태로 저장되네요.

어떤 원리로 인코딩이란게 되어 저런값이 최종적으로 나왔는지는 아직 모릅니다.

```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = u"안녕".encode("UTF-8")
>>> a
'\xe4\x95\x86\xe7\x85\x95'
>>>
```

그럼 반대로 UTF8로 인코딩된 Hex값을 직접 입력해서 출력해봅시다. "안녕"이 찍히네요...(헐)

```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print "\xec\x95\x88\xeb\x85\x95"
안녕
>>> print "\xec"
i
>>>
```

신기해서 "0xEC"만 입력 했더니 "ㅇ" 는 출력이 안됩니다. (ㅋㅋ) 사실 이부분이 중요하다고 느낀건,

1byte를 출력하면 외계어가 출력되지만 3바이트를 모두 모아서 출력하면 한글이 출력된다는 사실입니다.

```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = chr(0xec)
>>> b = chr(0x95)
>>> c = chr(0x88)
>>> print a+b+c
안
>>> |
```

여기서 파이썬이 사용자 입력한 "안녕" 문자열을 "0xEC9588 0xEB8595"으로 저장하기 위해 매칭한 표가 바로 인코딩의 정의에서 읽어오 이해가 되지 않았던 "문자 집합" 입니다.

아래 테이블은 한글 문자에 해당하는 UTF8, Unicode 값과 매치 해놓은 표입니다.

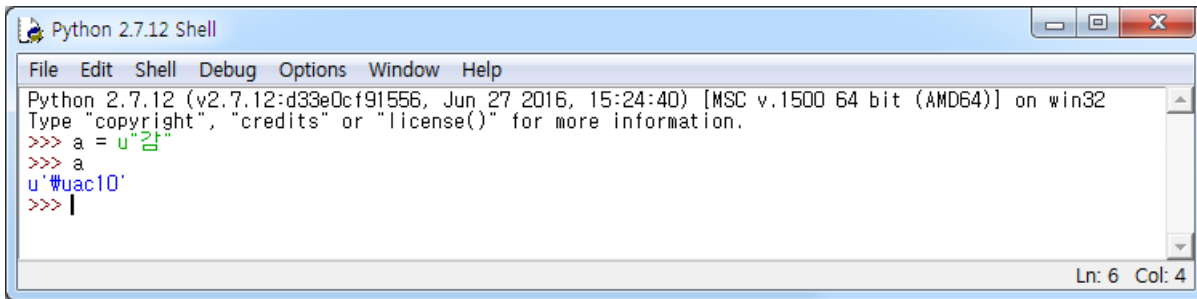
Unicode code point	character	UTF-8 (hex.)	name
U+AC00	가	0xea 0xb0 0x80	<Hangul Syllable, First>
U+AC01	각	0xea 0xb0 0x81	
U+AC02	갇	0xea 0xb0 0x82	
U+AC03	갓	0xea 0xb0 0x83	
U+AC04	간	0xea 0xb0 0x84	
U+AC05	감	0xea 0xb0 0x85	
U+AC06	갈	0xea 0xb0 0x86	
U+AC07	갇	0xea 0xb0 0x87	
U+AC08	갈	0xea 0xb0 0x88	
U+AC09	갇	0xea 0xb0 0x89	
U+AC0A	갈	0xea 0xb0 0x8a	
U+AC0B	갈	0xea 0xb0 0x8b	
U+AC0C	갇	0xea 0xb0 0x8c	
U+AC0D	갈	0xea 0xb0 0x8d	
U+AC0E	갈	0xea 0xb0 0x8e	
U+AC0F	갈	0xea 0xb0 0x8f	
U+AC10	감	0xea 0xb0 0x90	
U+AC11	갇	0xea 0xb0 0x91	
U+AC12	갇	0xea 0xb0 0x92	
U+AC13	갇	0xea 0xb0 0x93	
U+AC14	갇	0xea 0xb0 0x94	
U+AC15	가	0xea 0xb0 0x95	

그림출처 : <http://www.utf8-chartable.de/unicode-utf8-table.pl>

사용자가 "감"이라는 문자를 유니코드로 저장하려 합니다. 만약 우리가 파이썬이라면 어떻게 인코딩 해주어야 할까요?

그럼 위의 테이블에서 "감"에 매칭되는 유니코드에 해당되는 Hex 값인 "0xAC1"으로 저장해주면 될겁니다.

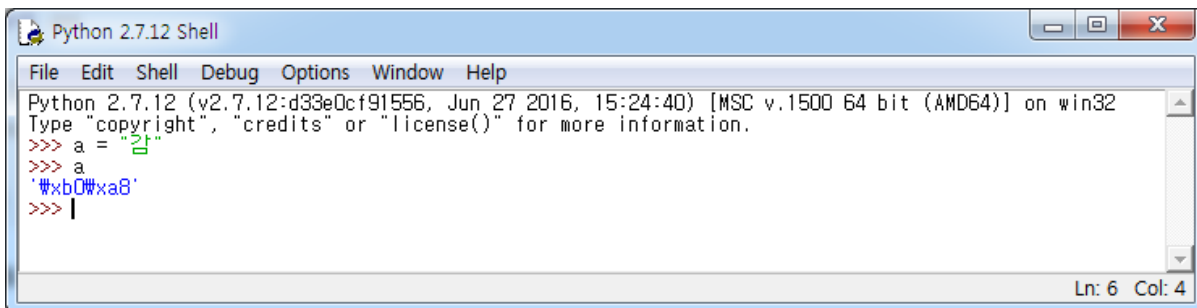
항상 의심의(?) 눈초리로 직접 확인해봅시다. 앞에 "u"라는 문자가 추가로 붙기는 하지만 값은 맞습니다.



```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = u"감"
>>> a
u'\uac10'
```

만약 encode 함수를 사용하지 않고 변수에 "안녕" 문자열을 할당하게 되면

위에서 보았던 UTF8이나 Unicode가 아닌 또다른 문자 집합으로 인코딩하여 저장합니다. (지정해주지 않았으니까요..)

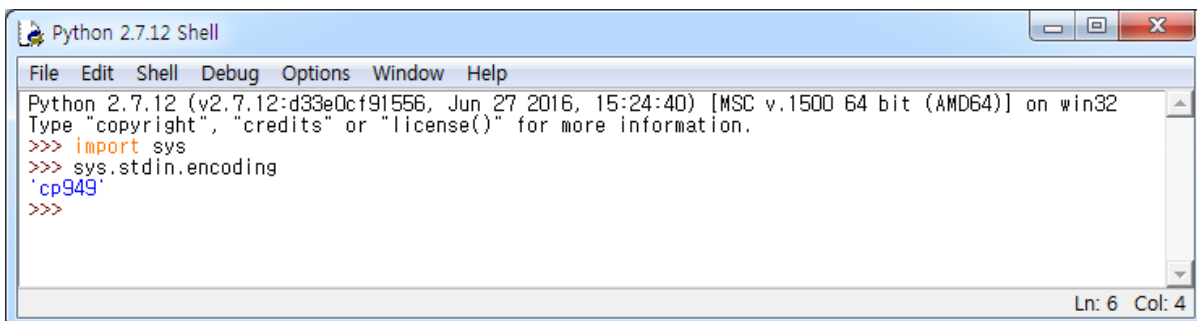


```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = "감"
>>> a
'\xb0\xa8'
```

아마 한글을 표현할 수 있는 또다른 인코딩 형태인듯 합니다.

처음에는 한글에 많이 쓰이는 EUC-KR인줄 알았으나, 실제로 아래와 같이 확인해보면 'CP949' 임을 알 수 있습니다.

(이래서 인코딩은 확인 또 확인...!!\_\_2017.07.17 수정)



```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import sys
>>> sys.stdin.encoding
'cp949'
```

코드페이지 949는 마이크로소프트(MS)에서 도입한 문자 집합이며, Ks\_c\_5601-1987으로도 불립니다. (위키백과)

사실 CP949는 EUC-KR의 확장이기도 하고, 하위 호환이 되기 때문에 착각할 여지가 있습니다.

아래 EUC-KR의 코드표에서 '감' 문자를 찾아봐도 "0xB0A8"으로 위의 IDLE에서의 결과값과 동일합니다.

b0a0	가	각	간	간	갈	갈	갈	감	갑	값	갓	갓	강	갓	갓
b0b0	갈	값	갈	개	객	객	객	객	갸	갸	갸	갸	갸	가	각
b0c0	갸	강	개	객	객	객	객	객	갸	갸	갸	갸	갸	갸	갸
b0d0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b0e0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b0f0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b1a0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b1b0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b1c0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b1d0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b1e0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b1f0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b2a0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b2b0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b2c0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b2d0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b2e0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸
b2f0	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸	갸

출처 : <http://www.mkexdev.net/Community/Content.aspx?parentCategoryID=4&categoryID=14&ID=125>

한글이 저장되는 방식이 네가지(Unicode, UTF8, EUC-Kr, CP949)나 나왔습니다. (사실 UTF-8은 유니코드의 저장 방식중 하나지만요)

만약 프로그래밍을 하다가 웹이 되었든 DB가 되었든 읽어들이 값이 "Wxb0Wxa8" 이었다면

이것이 EUC-KR(CP949)로 인코딩 되었다는 것을 알아야 우리가 원하는 UTF8이나 Unicode 인코딩 형식으로 바꿀 수 있습니다.

만약 이러한 개념이 없다면 무슨 인코딩 방식인지도 모른채 옴한 문자 집합 함수를 남발하게 될 것이며...

그럼에도 불구하고 자꾸 한글은 깨지고 UnicodeDecodeError를 보는 횟수는 늘어나고, 시간은 하염없이 흘러갈 확률이 높습니다.. (또르르..)

## 2. Unicode, UTF8에 대하여

이쯤에서 다시한번 인코딩의 정의를 짚어보고 넘어갑시다.

**"특정한 문자 집합 안의 문자들을 컴퓨터 시스템에서 사용할 목적으로 일정한 범위 안의 정수들로 변환하는 방법"**

그러니까...이걸 조금 쉽게 말하면

사용자가 문자를 입력하고 문자집합은 이거야! 라고 정해주면 시스템이 문자집합 테이블에 매칭되는 정수(or Hex)로 저장해주는 거군요?

(만약 문자집합을 지정해주지 않으면 시스템이나 프로젝트의 기본 로케일을 따르겠고요, 위에서는 CP949였죠)

이제 인코딩에 대해 쌀 한톨만큼은 알게된 것 같습니다. 적어도 인코딩의 흐름은 안것 같군요.

그렇다면 한글 입/출력시 가장 골머리 썩는 Unicode, UTF8, EUC-KR에 대해 좀 더 상세하게 알아보겠습니다.

### 1) Unicode

유니코드를 헛갈리지 않기 위해서는, 유니코드가 어떤 바이트 인코딩 방식이 아니라는 점을 먼저 알아야 합니다.

유니코드는 문자코드가 각국의 윈도우마다 겹치는 영역이 존재하기 때문에 이러한 현상이 발생하지 않기 위해

전세계 모든 언어에 겹치지 않는 코드를 할당(=매핑)한 코드입니다. 유니코드의 문자 집합은 위에서 이미 보았습니다.

그중에 우리가 자주 쓰는 '한글'이 할당받은 유니코드 범위는 **"AC00 ~ D7AF"** 인거죠.

유니코드 형식으로 저장하면 전세계 언어를 깨짐 없이 제공할 수 있겠군요.

## 2) UTF8

UTF8은 유니코드 기반의 가변 길이 문자 인코딩 방식입니다. 즉, 유니코드를 변환해서 UTF8로 만들었다는 뜻입니다.

그렇게 만든 이유는 유니코드를 8비트 단위로 만들기 위한 목적이며,

값이 큰 유니코드는 8비트 안에 전부 다 안들어가다 보니(?) 여러개의 바이트로 표현합니다.

(자연스럽게 UTF16은 16비트 기반으로 저장하기 위한 인코딩)

사실 예전부터 "UTF8 한글은 3byte" 라고 외우기만 했지 이런 의미가 있는지는 몰랐습니다.

아래 표를 보면 UTF8이 유니코드 문자 범위에 따라 1 ~ 4 바이트를 사용하는것을 알 수 있으며,

ASCII의 범위는 0 ~ 127 이라서 8비트로 표현이 가능하기 때문에 UTF8에서도 1byte로 표현되며,

"AC00 ~ D7AF" 범위인 한글은 1byte, 2byte로 표현 안되다 보니 3byte가 되었습니다.

이때 바이트 수가 여러개임을 판단 하기 위해 아래표 처럼 UTF8 형식이 지정되어 있습니다.

파란색 숫자는 바이트 수를 의미하며, **x**로 표현된 곳은 UTF8로 인코딩되기 위해 유니코드 값이 쪼개져서 들어갈 위치입니다.

여기서 중요한 사실은, **UTF8 인코딩 방식은 유니코드를 쪼개서 만드는 방식이기 때문에 무조건 유니코드를 한번 거쳐야 합니다.**

즉, 유니코드가 아닌 문자열을 UTF8로 만드는 함수에 넣었을 경우 우리가 자주 보았던 인코딩 에러가 발생할겁니다. (★★★)

Unicode		UTF8		
U+0000~U+007F	0 - 127	0xxxxxxx	(1byte)	ASCII
U+0080~U+07FF	128 - 2047	110xxxxx 10xxxxxx	(2byte)	C280 - DFD9
U+0800~U+FFFF	2,048 - 65,535	1110xxxx 10xxxxxx 10xxxxxx	(3byte)	E0A080 - FEFBFB
U+10000~U+10FFFF	65,536 - 1,114,111	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	(4byte)	F0908080 - F090BD9F

※ Tip : UTF8 첫번째 바이트에서 1의 개수는 총 바이트 수를 의미 :: 0 → 1byte, 110 → 2byte, 1110 → 3byte, 11110 → 4byte

"안" 문자의 경우에 유니코드값은 "U+C548" 이며, UTF8 인코딩 값은 "0xEC9588" 입니다.

그렇다면 Unicode를 UTF8로 변환하기 위해 한글 3byte의 x 값에 유니코드 값을 순서대로 대입합니다.

그리고 실제로 변환이 오류없이 잘되는지 IDLE로 확인해봅니다.

**1101100 10010101 10001000 = 0xEC9588 = "안"**

(C = 1100 / 5 = 0101 / 4 = 0100 / 8 = 1000)

```

Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = u"안"
>>> a
u'안'
>>> b = a.encode("UTF-8")
>>> b
'\xec\x95\x88'
>>>
  
```

### 3. 인코딩 변환 에러 케이스별 분석

#### 1) encode 함수

이쯤에서 우리가 저지르기 쉬운 실수를 살펴보면, 웹/DB/사용자입력 등에서 얻어온 문자열을 UTF8로 저장하기 위해 `encode("UTF-8")` 함수를 사용했는데 오류가 발생하는 케이스입니다.

이젠 보기만 해도 짜증이 치밀어 오르는 `UnicodeDecodeError` 입니다.

(파이썬 3.x 버전에서는 유니코드로 통일되어 아래 코드가 오류가 아닐겁니다)

```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = "안녕".encode("UTF-8")

Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    a = "안녕".encode("UTF-8")
UnicodeDecodeError: 'ascii' codec can't decode byte 0xbe in position 0: ordinal not in range(128)
>>> |
```

하지만 인코딩에 대해 어느정도 공부했으니 보이는게 있지요.

에러문율을 살펴보면, `Ascii codec`이 **"0xbe"**를 디코딩 하려고 했는데 `Ascii` 범위의 밖이라서 오류가 났다는 겁니다.

여기서 알 수 있는 사실은 2가지 있습니다.

=====

- ① `encode("UTF-8")` 함수를 사용했다고 해서 입력값을 **"1110xxxx"**의 x자리에 값을 우겨 넣어 인코딩을 해주지 않는다.
- ② `encode` 함수를 실행시키면 기본적으로 `Ascii`로 디코딩을 수행한 뒤 인코딩을 한다.

=====

따라서 사용자가 입력한 값을 디코딩해서 **"안녕"**을 처음 입력했을때 처럼 순수한(?) 값으로 되돌려 놓은 다음(뒤에서 다시 설명)

우리가 원하는 UTF 인코딩 과정에 돌입하게 됩니다. 그렇다면 위의 오류는 어떻게 해야 해결될까요?

에러를 다시 보면 **"0xbe"**를 디코딩 못했고, 우리가 아는 "안" 문자는 각 인코딩별로 값이 아래와 같습니다.

=====

- ① `Unicode` = **"0xC548"**
- ② `UTF8` = **"0xEC9588"**
- ③ `EUC-KR` = **"0xbec8"**

=====

그렇다면 위의 값을 보고 추측하건데, 내가 입력한 **"안"** 문자가 `EUC-KR(or CP949)`로 입력(후 인코딩) 되었고,

그것을 파이썬에 기본 설정되어 있는 아스키 코덱으로 디코딩 하려다 뺏어버렸다고 합리적인(?) 추측을 해볼 수 있습니다.

bea0	췐	췑	췒	췓	췔	췕	췖	췗	췘	췙	췚	췛	췜	췝	췞	췟	췠	췡	췢	췣	췤	췥	췦	췧	취	췩	췪	췫
beb0	췐	췑	췒	췓	췔	췕	췖	췗	췘	췙	췚	췛	췜	췝	췞	췟	췠	췡	췢	췣	췤	췥	췦	췧	취	췩	췪	췫
bec0	췐	췑	췒	췓	췔	췕	췖	췗	췘	췙	췚	췛	췜	췝	췞	췟	췠	췡	췢	췣	췤	췥	췦	췧	취	췩	췪	췫
bed0	앁	앗	앗	앙	알	알	앐	앑	앒	앓	암	압	앖	앗	았	앙	앚	앛	앜	앝	앞	앟	애	액	앢	앣	앤	앥
bee0	약	얀	알	알	암	압	앗	양	알	알	앐	앑	앒	앓	암	압	앖	앗	았	앙	앚	앛	앜	앝	앞	앟	애	액

그렇다면 우리가 직접 디코딩 함수를 통해 EUC-KR로 디코딩 해준 뒤 UTF8로 인코딩 해주면 어떻게 될까요?

왜 오류가 났는지 파악이 되어 쉽게 해결이 되었습니다.

하지만 다른 상황에서 UnicodeDecodeError가 발생했을 시 EUC-KR 디코딩이 만사해결책은 아닐겁니다.

```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = "안녕".encode("UTF-8")
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    a = "안녕".encode("UTF-8")
UnicodeDecodeError: 'ascii' codec can't decode byte 0xbe in position 0: ordinal not in range(128)
>>> a = "안녕".decode("EUC-KR").encode("UTF-8")
>>> a
'\xef\x95\x88\xeb\x85\x95'
```

UTF8 인코딩 값은 유니코드를 대입해서 만들기 때문에 위의 코드가 동작하기 위해서는

"안녕".decode("EUC-KR") 함수는 유니코드값을 반환한다는 의미가 되는데, 이것도 실제로 확인해봅니다.

```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = "안녕".encode("UTF-8")
>>> a
'\xef\x95\x88\xeb\x85\x95'
>>> b = a.decode("EUC-KR")
>>> b
u'\uc548\u155'
```

그렇다면 일단 유니코드로 저장하게 되면 EUC-KR이나, UTF8, UTF16 등등 다른 인코딩 값으로 쉽게 변환 가능하다는 소린데,

실제로 문자열을 유니코드로 저장한 뒤 종류별로 인코딩을 수행해봅니다. 오류없이 잘 동작하네요.

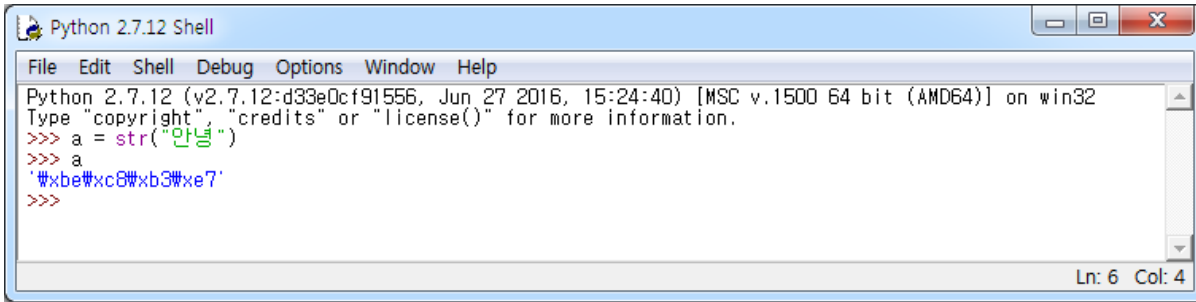
```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = u"안녕"
>>> a.encode("UTF-8")
'\xef\x95\x88\xeb\x85\x95'
>>> a.encode("UTF-16")
'\xff\xfe\xef\x50\x01'
>>> a.encode("EUC-KR")
'\xef\x95\x88\xeb\x85\x95'
```

## 2) str, unicode 함수

먼저 str 함수에 한글을 넣으면 어떻게 저장될까요? 저의 경우에는 EUC-KR(or CP949)로 저장됩니다.

아마 이렇게 EUC-KR로 저장되는건 시스템의 기본 로케일이 적용되었을 가능성이 높습니다.

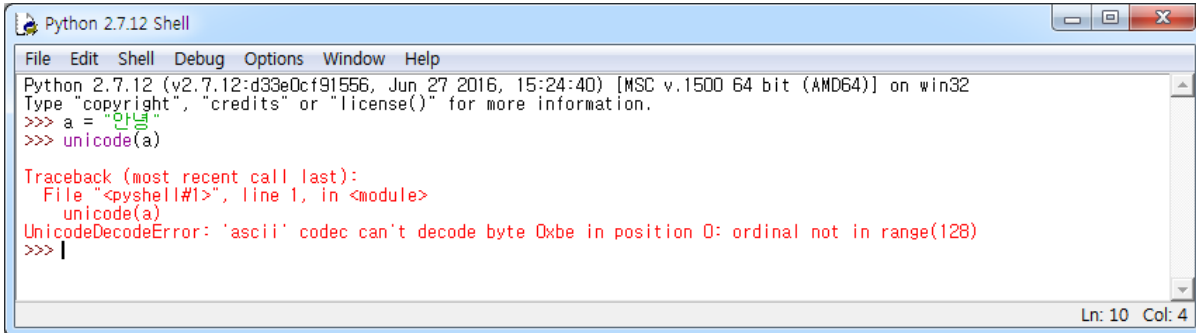




```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = str("안녕")
>>> a
'안녕'
>>> a.encode('latin1')
'\xb5\xbe\xba\x80'
>>>
```

그럼 앞에서 유니코드로 저장해두면 다른 인코딩으로 변환하기 쉽다고 하였으니,

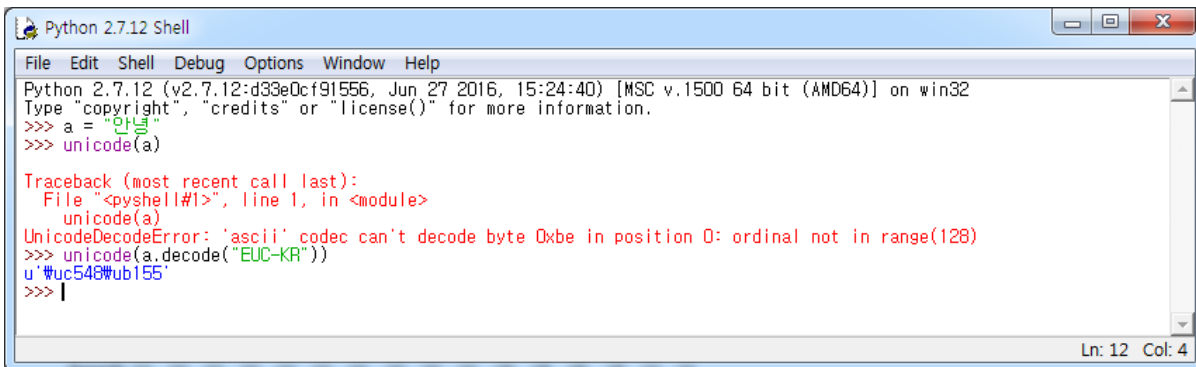
"안녕"이라는 문자열을 유니코드로 저장하기 위해 unicode 메소드를 실행하면 어떻게 될까요?



```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = "안녕"
>>> unicode(a)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    unicode(a)
UnicodeDecodeError: 'ascii' codec can't decode byte 0xb5 in position 0: ordinal not in range(128)
>>>
```

쉽게 넘어가는데 하나 없네요. 이 환경에서 unicode 함수를 이용하여 유니코드 값을 얻기 위해선 디코딩을 해줘야 합니다.

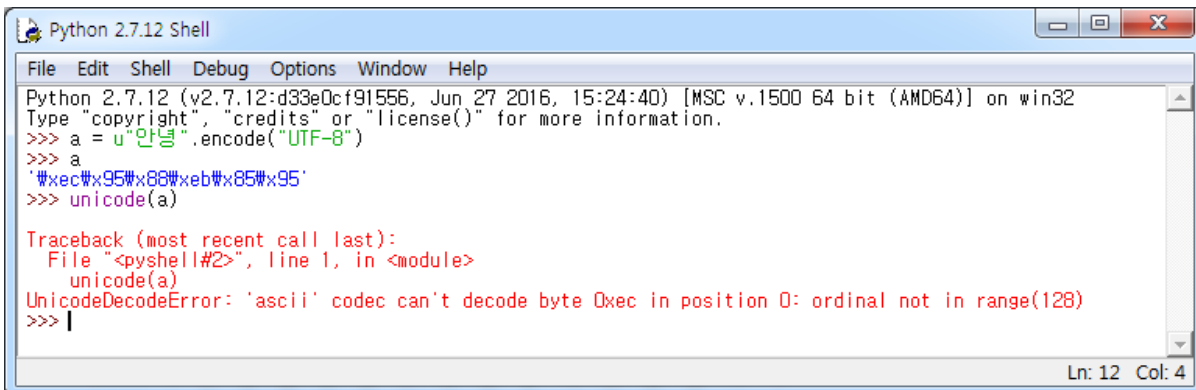
차라리 선언부터 `a = u"안녕"` 이라고 선언하는게 더 쉬울지도 모르겠지만, 다른곳에서 문자를 받아오는 경우도 많으니까요.



```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = "안녕"
>>> unicode(a)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    unicode(a)
UnicodeDecodeError: 'ascii' codec can't decode byte 0xb5 in position 0: ordinal not in range(128)
>>> unicode(a, 'EUC-KR')
u'\uc548\uac15'
>>>
```

그렇다면 유니코드 계열인 UTF8 인코딩 문자를 유니코드로 다시 돌리는건 에러 없이 스무스하게 될까요?

아니요. 에러가 납니다...**이쯤되면 인코딩 하려면, 허락받고 해야될 거 같습니다..**

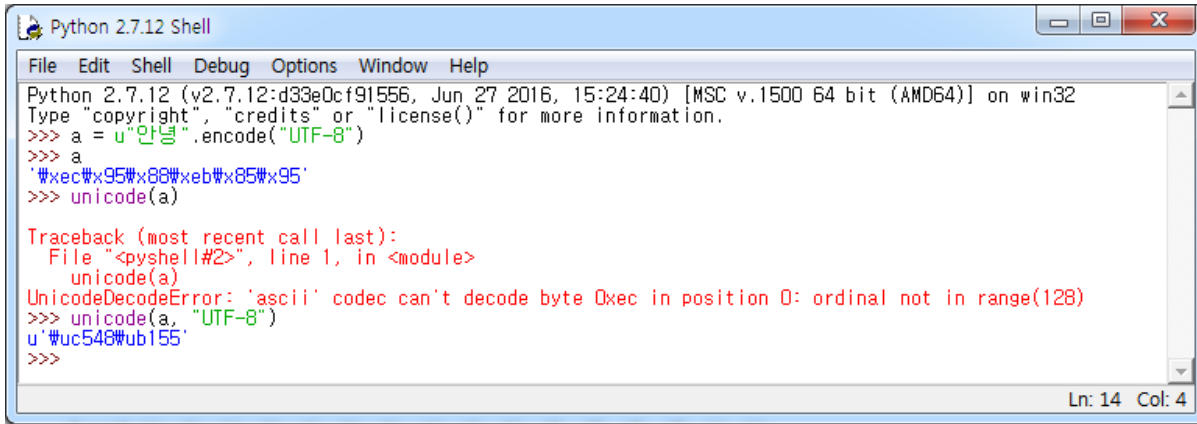


```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = u"안녕".encode("UTF-8")
>>> a
'\xb5\xbe\xba\x80'
>>> unicode(a)
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    unicode(a)
UnicodeDecodeError: 'ascii' codec can't decode byte 0xb5 in position 0: ordinal not in range(128)
>>>
```

하지만 UTF8의 경우에는 유니코드 계열이나 보니, **"UTF8"** 이라고 인자를 넣어주면 친절하게 변환해줍니다. (휴)



아니면 `decode("UTF-8")` 함수를 사용해도 됩니다.



```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = u"안녕".encode("UTF-8")
>>> a
'\xec\x95\x88\xeb\x85\x95'
>>> unicode(a)

Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    unicode(a)
UnicodeDecodeError: 'ascii' codec can't decode byte 0xec in position 0: ordinal not in range(128)
>>> unicode(a, "UTF-8")
u'uc548ub155'
>>>
```

#### 4. 마무리..

여기저기 인코딩에 대해 검색하다 가장 기억에 남는 말이 있습니다.

**"이놈의 인코딩은 해결해도 본전이며 못하면 시간낭비" 라고요... ㅎㅎ**

아마 개발을 하시는 분들이라면 한번쯤은 정리하셨겠지만... 뒤늦은 기초의 중요성을 체감합니다.

아무쪼록 인코딩 문제로 머리 싸매고 계시던 분들에게 조금이라도 도움이 되셨길 바랍니다.