

znjxldptj enqjftlrdmfh

dk gksrmfdl dks cuwu

dldlsdyd @ UPnL

36ck dnjzmtiq

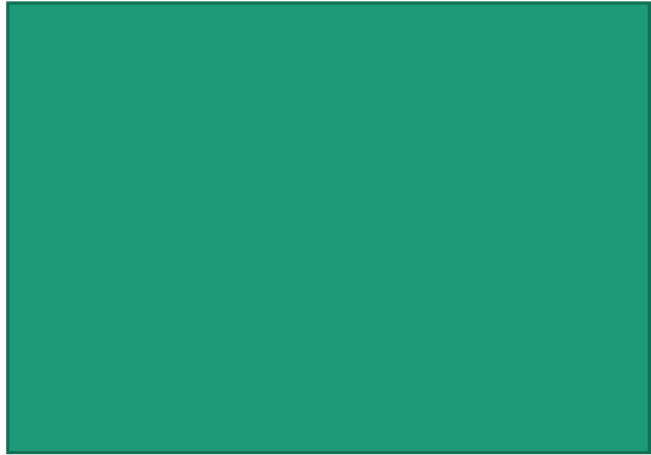
학습 목표

- 이 발표를 들으면 %EB%B0%9C%ED%91%9C가 뭔지 알 수 있음
- trans-bot를 만들 수 있음

사건

- IRC를 하는데 갑자기 한글이 안 쳐짐 (꽤 자주 그럼)
 - 메모장에도 안 쳐짐
 - 크롬 URL 쓰는 곳에서만 쳐짐
- oh shit!
- fuxk!
- 근데 영어로 말을 하기에는 내 영어 실력이 너무 부족함
- rmsid dlfigrp claus dkfdktj qkRnjwnjTdmaus whgrpTek
- (그냥 이렇게 치면 알아서 바꿔줬으면 좋겠다)

문득 생각난 글



가져오기...

2014-04-01 오후 4:06:17 조회 190

기타 북마크

--
sms dudxk z
todrkreh dks gkrh dITdjTsmsep clekahsl duddj skdhkti qkRNrl rnlcksma
rmfotj dufjans
dpaxlfmf todrkrgrkh dITddidy
dltlqdhdlf dltlqdbrdlf
tkdnifdlqselek
anj rhdtlrwjdls rhdwlsmss dkslsIRK
rmfjgekrndy
dd
rm...rmflrh (sksms gkrwjadmfm vhrlgkrhtlvek) sksms rhkdp ehndnamf wnrh tlvek gksms tkfkaemfdms
rhe wkdxjcorrhk dpaxlcordmf tjsqkfgkfrislRK rlekfltpdy ggggg
dlfdms durtl qnaekagodiwl
:)
[Redacted]

댓글(8) | 역인글(0) | 추천(0)

역인글 주소 <http://www.snucse.org/Trackback/405795>

06:31

번역은 [Redacted] [ddd/a.py/say](#) 여기서 했습니다. UPnL은 이런 것도 만드는 동아리예요 많은 활동 해주세요 :) 추천:5

나도 외계어 해야할 것 같잖아

추천:0

는 영타 ㅋㅋ 생각도 안 하고 있었는데 치다보니 영어 나와서 바꾸기 귀찮음 그래서 여러분 엠티를 생각하고 있어요 이십오일 이십육일 사월입니다 뭐 공식적인 공지는 아니니까 그렇다구요 ㅠㅠ 그...그리고 (나는 확절을 포기하고싶다) 나는 과에 도움을 주고 싶다 하는 사람들은 곧 장터책과 엠티책을 선별할거니까 기다리세요 ㅎㅎㅎㅎㅎ 일은 역시 불달해야지 :)

추천:1

번역은 [Redacted] [ddd/a.py/say](#) 여기서 했습니다. UPnL은 이런 것도 만드는 동아리예요 많은 활동 해주세요 :) 추천:5

와ㅋㅋ

추천:0

그나저나 왜 불답이라고 친거지

추천:0

대체 저건 어디서 찾은거...

추천:2

그러게 말입니다.. 예전에 [Redacted] 만드셨던 것 같은데

추천:0

한영키 변환 정도는 그냥 읽으면 해석 되어야 하는거 아닌가요

추천:0

깨달음

- 아 저걸 만들면 되겠네
- 근데 아무것도 모름 ㅜㅜ
- 차근차근 필요한 걸 생각해보자

필요한 것

dkssud



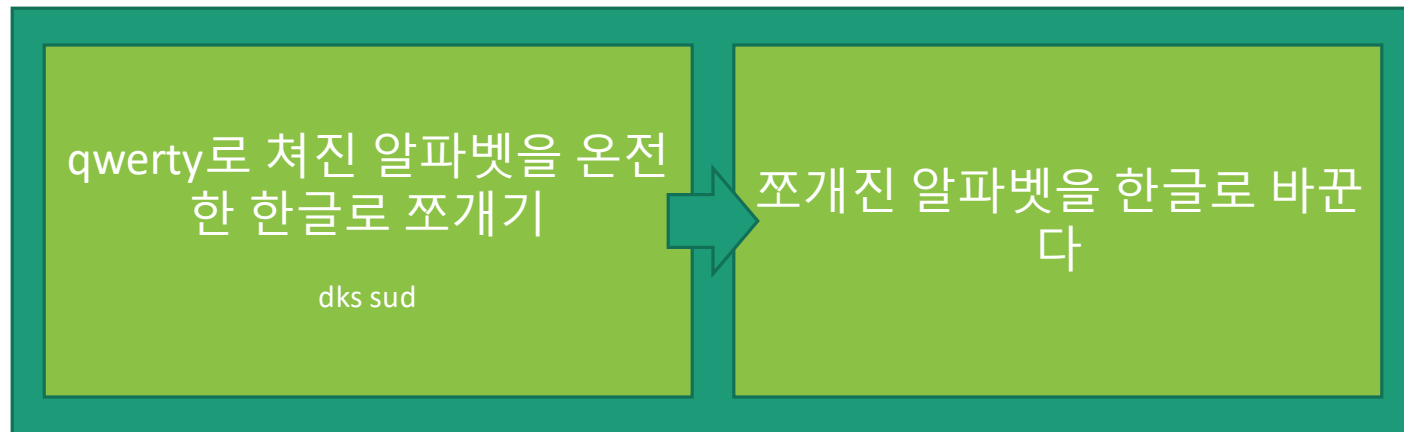
qwerty to 한글 (두벌식)



안녕

필요한 것

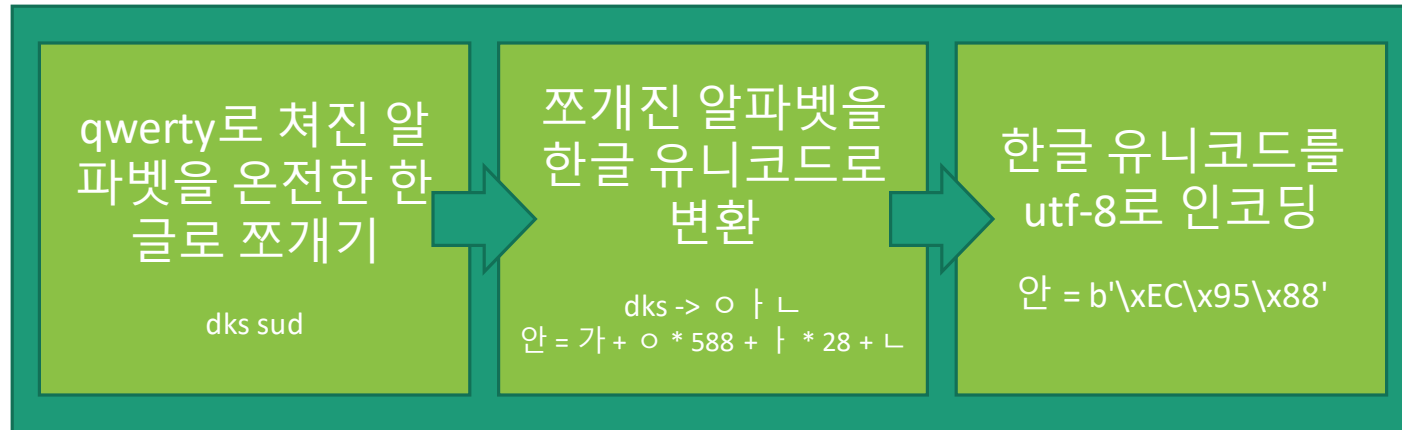
dkssud



안녕

필요한 것

dkssud



안녕

인코딩? 유니코드?

- 글자표현 어떻게?
 - 글자마다 코드 줌
 - 아스키코드
 - a는 97 (맞나), 0은 0x30
- 알파벳은 저거 쓰면 되고 한글은?
 - CP949
 - EUC-KR
 - 유니코드

인코딩? 유니코드?

- 한글 인코딩
 - <https://ko.wikipedia.org/wiki/EUC-KR>
 - https://ko.wikipedia.org/wiki/코드_페이지_949
- 자세한 건 링크 참조
- 뭣보다 문제점
 - 완성형으로 넣어놨는데 없는 글자가 있어서 표현 못 하는 한글이 있음
 - euc-kr에는 'ㄹ' 같은 거 없어서 표현 못 함
 - 코드 충돌 문제
 - 데이터는 똑같은데 코드 페이지가 다르면 글자가 다르게 보임
 - 한글과 가나문자 등 동시에 표현 못 함 (예시일 뿐)

인코딩? 유니코드?

- 유니코드
- 세상 모든 글자에 코드를 박자 (code point라고 함)
- 1991년부터 지금까지 꾸준히 업데이트 되고 있음
- <https://ko.wikipedia.org/wiki/유니코드>
 - 유니코드 목록 항목 참조
- G는 0x47, ω는 0x3C9, بن는 0x69A, ٢는 0x3066, 꺾은 0xAD37

인코딩? 유니코드?

- 인코딩 - 정보를 다른걸로 변환
- 유니코드 인코딩
 - 유니코드를 어떻게 보여줄 건지
- 유니코드 인코딩은 여러 방식이 있음
 - UTF-8
 - UTF-16
 - 너무 많아서 생략
- Python에서 UTF-8을 쓰니까 이것만 알아두자

UTF-8

- <https://ko.wikipedia.org/wiki/UTF-8>
- UTF-8 인코딩은 유니코드 한 문자를 나타내기 위해 1바이트에서 4바이트까지를 사용한다. 예를 들어서, U+0000부터 U+007F 범위에 있는 ASCII 문자들은 UTF-8에서 1바이트만으로 표시된다. **4바이트로 표현되는 문자는 모두 기본 다국어 평면(BMP) 바깥의 유니코드 문자이며, 거의 사용되지 않는다.**

UTF-8

- <https://ko.wikipedia.org/wiki/UTF-8>
- UTF-8은 다음과 같은 구조

코드 범위(십육진법)	UTF-8 표현(이진법)	설명
000000-00007F	0xxxxxxx	ASCII와 동일한 범위
000080-0007FF	110xxxxx 10xxxxxx	첫 바이트는 110 또는 1110으로 시작하고,
000800-00FFFF	1110xxxx 10xxxxxx 10xxxxxx	나머지 바이트들은 10으로 시작함
010000-10FFFF	11110zzz 10zzxxxx 10xxxxxx 10xxxxxx	UTF-16 서러게이트 쌍 영역 (yyyy = zzzzz - 1). UTF-8로 표시된 비트 패턴은 실제 코드 포인트와 동일하다.

UTF-8

- G는 0x47, ω는 0x3C9, بنس는 0x69A, ٲ는 0x3066, 꺠은 0xAD37
- $G < 0x80$ 이므로 $G = b'\backslash x47'$
- $\omega < 0x800$ 이고, 011 1100 1001 이므로
- '110xxxxx 10xxxxxx'에 넣으면 11001111 10001001
- $\omega = b'\backslash xCF\backslash x89'$
- 꺠은 $< 0x10000$ 이고, 1010 1101 0011 0111
- '1110xxxx 10xxxxxx 10xxxxxx'에 넣으면 11101010 10110100 10110111
- 꺠 = $b'\backslash xEB\backslash xB4\backslash xB7'$

Python에서 출력

- python3 기준
- `print('\u[code point]')`
- `print('\uAD37')`
- 꺾
- 근데 0xAD37를 유니코드로 출력하려고 봤더니
- 그런 거 못 하는 것 같더라
 - 있는진 모르겠지만 못 찾음
 - `print('\u%x' % (0xAD37))` 같은 걸 하고 싶었음

Python에서 출력

- 인코딩 손수 해주기

```
def bytearray_to_utf8(l):  
    '''  
    convert byte array to utf-8 string  
  
    byte array contains utf-8 byte  
    '''  
    assert len(l) > 0, '빈 걸 주냐'  
    if l[0] < 0b10000000:  
        assert len(l) >= 1, '하나 이상 이어야 함'  
        return bytearray(l[:1]).decode('UTF-8')  
    elif l[0] < 0b11100000:  
        assert len(l) >= 2, '둘 이상 이어야 함'  
        return bytearray(l[:2]).decode('UTF-8')  
    elif l[0] < 0b11110000:  
        assert len(l) >= 3, '셋 이상 이어야 함'  
        return bytearray(l[:3]).decode('UTF-8')  
    else:  
        assert len(l) >= 4, '넷 이상 이어야 함'  
        return bytearray(l[:4]).decode('UTF-8')
```

```
def unicode_to_bytearray(u):  
    '''  
    convert unicode to utf-8 bytearray  
    '''  
    assert 0 <= u and u <= 0xFFFF, '다른 건 처리 안 해요 ㅜ'  
    if u < 0x80:  
        return [u]  
    elif u < 0x800:  
        d0 = u % 64  
        d1 = (u >> 6) % 32  
        return [d1 + 0b11000000, d0 + 0b10000000]  
    elif u < 0x10000:  
        d0 = u % 64  
        d1 = (u >> 6) % 64  
        d2 = (u >> 12) % 16  
        return [d2 + 0b11100000, d1 + 0b10000000, d0 + 0b10000000]  
  
def unicode_to_utf8(u):  
    return bytearray_to_utf8(unicode_to_bytearray(u))
```

Python에서 출력

```
glglgozz@kari:~/alphabet-hangul-translator$ python3
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from translator import unicode_to_utf8
>>> print(unicode_to_utf8(0x47))
G
>>> print(unicode_to_utf8(0x3C9))
ω
>>> print(unicode_to_utf8(0x69A))
꺄
>>> print(unicode_to_utf8(0xAD37))
꺄
>>> █
```

필요한 것



한글 유니코드

- 우리의 관심사 한글의 유니코드는 어디에 있나
- 한글은 조합형 부분과 호환 자모 부분, 그리고 완성형 부분이 있다
- 조합형: 유니코드 0x1100-0x11FF
 - 초성 중성 종성마다 값이 있고 이들을 이으면
 - 초성 ㄱ인 0x1100과 중성 ㅏ인 0x1161를 출력하면 '가'가 되는 것
 - 하지만 `print('\u1100\u1161')`을 해봐도 '가'가 나오지 않았다...
 - <http://allog.tistory.com/41> 여기서 됨 (뭘 한 거지?)
 - 옛 한글도 지원

한글 유니코드

- 한글 호환 자모: U+3130-U+318E
 - https://en.wikipedia.org/wiki/Hangul_Compatibility_Jamo
 - **Hangul Compatibility Jamo** is a Unicode block containing Hangul characters for compatibility with Korean Standard [KS X 1001:1998](#).
 - 와..... 이거 한국어 문서 없음 ㅋㅋㅋ
 - 흔히 보는 'ㄱ' 'ㄴ' 'ㅇ' 'ㄷ' 'ㄹ' 'ㅁ' 'ㅂ' 다 여기에 속하는 글자

한글 유니코드

- 완성형: U+AC00-U+D7AF
 - '가'부터 '힉'까지
 - 현대 한글이 나타낼 수 있는 모든 글자 넣음
 - 순서는 다음과 같다
- '안'의 유니코드는
 - 가 + ㅇ * 21 * 28 (=588) + ㅏ * 28 + ㄴ
 - $0xAC00 + 11 * 588 + 0 * 28 + 4$

```
>>> from translator import unicode_to_utf8
>>> print(unicode_to_utf8(0xAC00 + 11 * 588 + 4))
안
>>> █
```

Chosung		Joongsung		Zongsung	
ㄱ 0	ㅅ 10	ㅏ 0	ㅑ 11	Null 0	ㅗ 14
ㄴ 1	ㅇ 11	ㅓ 1	ㅕ 12	ㅜ 1	ㅛ 15
ㄷ 2	ㅈ 12	ㅖ 2	ㅗ 13	ㅠ 2	ㅜ 16
ㄹ 3	ㅊ 13	ㅙ 3	ㅜ 14	ㅡ 3	ㅝ 17
ㅁ 4	ㅌ 14	ㅚ 4	ㅛ 15	ㅞ 4	ㅞ 18
ㅂ 5	ㅋ 15	ㅜ 5	ㅜ 16	ㅟ 5	ㅟ 19
ㅅ 6	ㅍ 16	ㅟ 6	ㅠ 17	ㅠ 6	ㅠ 20
ㅇ 7	ㅊ 17	ㅡ 7	ㅡ 18	ㅢ 7	ㅢ 21
ㅈ 8	ㅎ 18	ㅣ 8	ㅣ 19	ㅤ 8	ㅤ 22
ㅊ 9		ㅤ 9	ㅣ 20	ㅥ 9	ㅥ 23
		ㅦ 10		ㅦ 10	ㅦ 24
				ㅧ 11	ㅧ 25
				ㅨ 12	ㅨ 26
				ㅩ 13	ㅩ 27

알파벳을 한글로

- 주석으로 표현된 코드표를 만듦 (유니코드 순서와 같음)
- 각 알파벳 입력마다 맵 정보

```
# 0          1          2
# 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
# ㄱ ㅋ ㄴ ㄷ ㄹ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅁ ㅂ ㅃ ㅄ ㅅ ㅆ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ
# ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ
# ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ
# 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

first_only_map = {
    'r': 0, 'R': 1, 'rt': 2, 's': 3, 'sw': 4,
    'sg': 5, 'e': 6, 'E': 7, 'f': 8, 'fr': 9,
    'fa': 10, 'fq': 11, 'ft': 12, 'fx': 13, 'fv': 14,
    'fg': 15, 'a': 16, 'q': 17, 'Q': 18, 'qt': 19,
    't': 20, 'T': 21, 'd': 22, 'w': 23, 'W': 24,
    'c': 25, 'z': 26, 'x': 27, 'v': 28, 'g': 29,
}
```

```
first_map = {
    'r': 0, 'R': 1, 's': 2, 'e': 3, 'E': 4,
    'f': 5, 'a': 6, 'q': 7, 'Q': 8, 't': 9,
    'T': 10, 'd': 11, 'w': 12, 'W': 13, 'c': 14,
    'z': 15, 'x': 16, 'v': 17, 'g': 18,
}

middle_map = {
    'k': 0, 'o': 1, 'i': 2, 'O': 3, 'j': 4,
    'p': 5, 'u': 6, 'P': 7, 'h': 8, 'hk': 9,
    'ho': 10, 'hl': 11, 'y': 12, 'n': 13, 'nj': 14,
    'np': 15, 'nl': 16, 'b': 17, 'm': 18, 'ml': 19,
    'l': 20,
}

last_map = {
    None: 0,
    ':': 0, 'r': 1, 'R': 2, 'rt': 3, 's': 4,
    'sw': 5, 'sg': 6, 'e': 7, 'f': 8, 'fr': 9,
    'fa': 10, 'fq': 11, 'ft': 12, 'fx': 13, 'fv': 14,
    'fg': 15, 'a': 16, 'q': 17, 'qt': 18, 't': 19,
    'T': 20, 'd': 21, 'w': 22, 'c': 23, 'z': 24,
    'x': 25, 'v': 26, 'g': 27,
}
```

알파벳을 한글로

- 잘라진 알파벳 한글 코드로

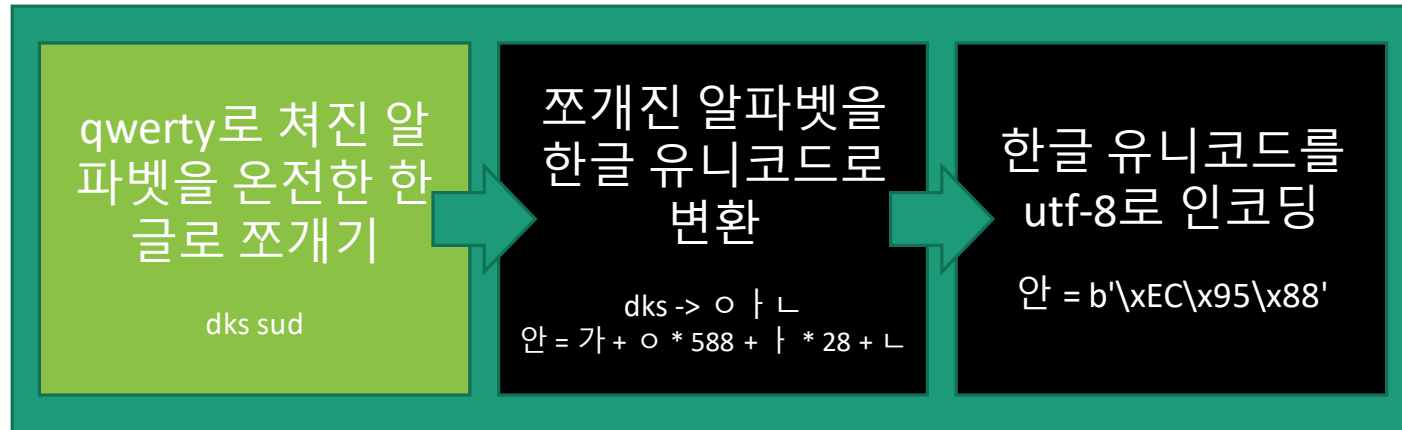
```
def to_hangul_code(first, middle, last):  
    '''  
    알파벳을 위의 맵핑 정보를 이용해서  
    한글 코드로 변환한다  
    '''  
    t_first = None  
    t_middle = None  
    t_last = None  
    assert first is not None or middle is not None  
    if middle is None:  
        t_first = first_only_map[first]  
    else:  
        if first is None:  
            t_first = None  
        else:  
            t_first = first_map[first]  
        t_middle = middle_map[middle]  
    t_last = last_map[last]  
    return (t_first, t_middle, t_last)
```

- 초성 중성 종성을 유니코드로

```
def to_unicode(first, middle, last):  
    '''  
    초성 중성 종성을 받아서 (위 주석 표를 참고)  
    한글 유니코드로 변환한다  
    '''  
    if first is None or middle is None:  
        if first is not None:  
            return 0x3131 + first  
        if middle is not None:  
            return 0x3131 + 30 + middle  
    else:  
        if last is None:  
            last = 0  
        return 0xAC00 + first * 21 * 28 + middle *  
  
def to_hangul(first, middle, last):  
    f, m, l = to_hangul_code(first, middle, last)  
    return unicode_to_utf8(to_unicode(f, m, l))
```


필요한 것

dkssud



안녕

어떻게 만드나?

- 그냥 긴 qwerty string을 하나하나 보고 만들면 됨
- 글자를 치다 보면 글자가 완성되는 때가 있음
- 그 동안 모은 초성 중성 종성을 위에서 만든 거에서 내보내줌
- FSM을 이용하자?

FSM 만들기

- 상태 종류

- 초기상태 (시작, 알파벳 말고 다른 글자가 나와 초기화된 상태)
- 초성만 온 상태 (ㄱ ㄴ ㅂ)
- 중성만 온 상태 (ㄷ ㄹ ㅈ)
- 초성과 중성이 온 상태 (가 수 와)
- 초성 중성 종성이 온 상태 (갈 굶 째)

- 입력

- qwertyuiopasdfghjklzxcvbnmQWERTOP
- 기타 글자

입력 분류

- 들어올 수 있는 인풋들을 미리 적절하게 나눔
 - 자음 rRseEfaqQtTdwWxzcvg
 - 모음 koiOjpuPhynbml
- 대소문자가 다른 입력인 글자
 - 편의상 애네 빼고는 소문자로 고칠 예정
 - 물론 입력으로 ASDF도 올 수 있음
- 하나 ㄱ처럼 알파벳 2개를 소비하는 글자
 - 뒤에 뭔가 나올 수 있는 애들이 나오면 그 다음 것도 보자
- 절대 종성으로 올 수 없는 자음
 - 바자ㄷ 는 종성에 못 옴
 - 종성 처리할 때 애네를 만나면 초성으로 강제로

```
consonants = 'rRseEfaqQtTdwWxzcvg'
vowels = 'koiOjpuPhynbml'
case_sensitive = 'QWERTOP'
cant_be_last = 'EQW'

followed_consonants = {
    'r': 't',
    's': 'wg',
    'f': 'raqtxvg',
    'q': 't',
}

followed_vowels = {
    'h': 'kol',
    'n': 'jpl',
    'm': 'l',
}
```

FSM 동작

- 초기상태일 때
 - 자음이 들어오면
 - 초성만 있는 상태로
 - 모음이 들어오면
 - 중성만 있는 상태로
- 중성만 있는 상태일 때
 - 일단 모음이 완성 글자가 됨
 - 자음이 들어오면
 - 초성만 있는 상태로
 - 모음이 들어오면
 - 중성만 있는 상태로

```
if state == State.empty:
    if ch in consonants:
        first = ch
        if ch in followed_consonants.keys():
            if next_ in followed_consonants[ch]:
                first += next_
                i += 1
        state = State.first_only
    elif ch in vowels:
        middle = ch
        if ch in followed_vowels.keys():
            if next_ in followed_vowels[ch]:
                middle += next_
                i += 1
        state = State.middle_only
    else:
        pass
elif state == State.middle_only:
    new_str.append(to_hangul(first, middle, last))
    first = None; middle = None; last = None
    if ch in consonants:
        first = ch
        if ch in followed_consonants.keys():
            if next_ in followed_consonants[ch]:
                first += next_
                i += 1
        state = State.first_only
    elif ch in vowels:
        middle = ch
        if ch in followed_vowels.keys():
            if next_ in followed_vowels[ch]:
                middle += next_
                i += 1
        state = State.middle_only
    else:
        pass
```

FSM 동작

- 초성만 있는 상태일 때
 - 자음이 들어오면
 - 글자 완성
 - 초성만 있는 상태로
 - ㄱ ㄱ
 - 모음이 들어오면
 - 혹시 ㄱ같이 2개면
 - ㄱ 혼자 완성 시키고 ㄴ은 현재 초성으로
 - ㄱ사
 - 초성과 중성이 온 상태로

FSM 동작

- 초성과 중성이 온 상태
 - 자음이 들어오면
 - 종성이 될 수 없는 자음이면
 - 글자 완성
 - 초성만 있는 상태로
 - 아ㅈ
 - 초성 중성 종성이 온 상태로
 - 모음이 들어오면
 - 글자 완성
 - 모음만 있는 상태로
 - 우ㅏ

FSM 동작

- 초성 중성 종성이 온 상태
 - 자음이 들어오면
 - 글자 완성
 - 초성이 온 상태로
 - 모음이 들어오면
 - 종성 중 뒤의 겹 초성으로
 - 나머지로 글자 완성

FSM 동작

- 알파벳이 아닌 글자는?
- 사람들이 분명히 트롤링을 해댈 것
- Python3를 믿고 그냥 보내주자
- 무슨 상태이든 간에
 - 자음이나 모음이 아닌 글자가 오면
 - 글자 완성
 - 들어온 글자 추가
 - 초기상태로

완성!

```
glglgozz@kari: ~  
glglgozz@kari:~/alphabet-hangul-translator$ python3  
Python 3.4.3 (default, Oct 14 2015, 20:28:29)  
[GCC 4.8.4] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> from translator import qwerty_to_2_set_korean  
>>> print(qwerty_to_2_set_korean('znjxldptj enqjftlrdmfh'))  
퀵 티 에 서 두 벌 식 으 로  
>>> print(qwerty_to_2_set_korean('dhkstjd!'))  
완 성 !  
>>> print(qwerty_to_2_set_korean('dkRk cjdmadp skdhs dbxldpvm8dms qkfvyduTdma'))  
  
아 까 처 음 에 나 온 유 티 에 프 8은 발 표 였 음  
>>> █  
  
[hangul] 0:python3- 1:python3* "kari" 21:41 18-Sep-16
```

완성!

```
glggozz@kari: ~  
교 회 로 처 치 (+100)  
학 교 갈 준 비 \  
SE> 좋 네  
학 교 를 가 도 되 고  
> 요 시 나 먹 어 도 되 고  
~kkyung@1.230.51.60] has joined #snucse16  
[~mumu@1.230.51.60] has joined #snucse16  
/#snucse16 [+o 공 ] by pps789_c  
/#snucse16 [+o mumu] by pps789_c  
/#snucse16 [+o 메 구 밍 ] by pps789_c  
!h dlwp dlrjf dbdydgkrp Tmf Eork ehoTtmq  
bot> 이제 이걸 유용하게 쓸 때가 됐습  
th> where  
!h gksrmf dks cuwlazzzz  
bot> 한글 안 쳐 짐 ㅋㅋㅋㅋ  
fuxk  
th> 한글 이 안 쳐 저 요  
!h qhteh gksrmfdmf clsmsep  
bot> 붓 도 한글 을 치 는 데  
!h dnjzmtiq wkfy ek aksema(?)  
bot> 워크 샵 자 료 다 만 듦 (?)  
(+i) [4:uriirc/#snucse16(+ns)]  
[0] 0:irssi* 1:irssi- "kari" 20:56 18-Sep-16
```

소감

- 코드를 정리하고 깃헙에나 올려야겠다
 - 클래스로 만들어야겠다
- 웹 서비스로 올려야겠다
- 세벌식은 내가 안 써서 모르겠다
- 유니코드 이런 거 무슨 말인지 알아듣게 되었다