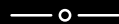


시스피커를 이용한 LoRa 모듈 제어

2019. 01. 26 토요일
박형준 (khuphj@gmail.com)





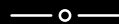
CONTENTS

Python 실행방법

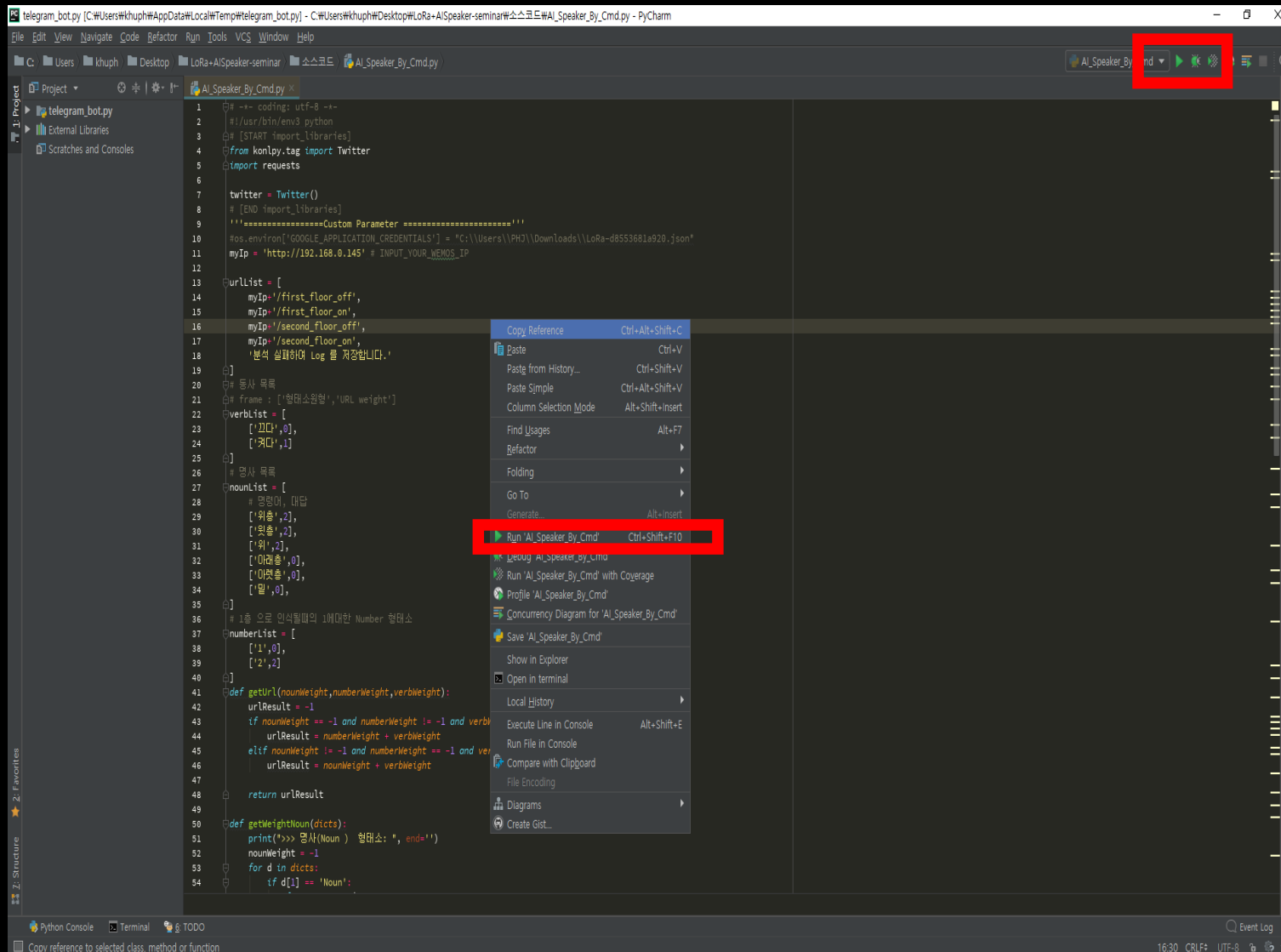
N-Gram

핵심 코드설명

Q & A



01. python 실행방법



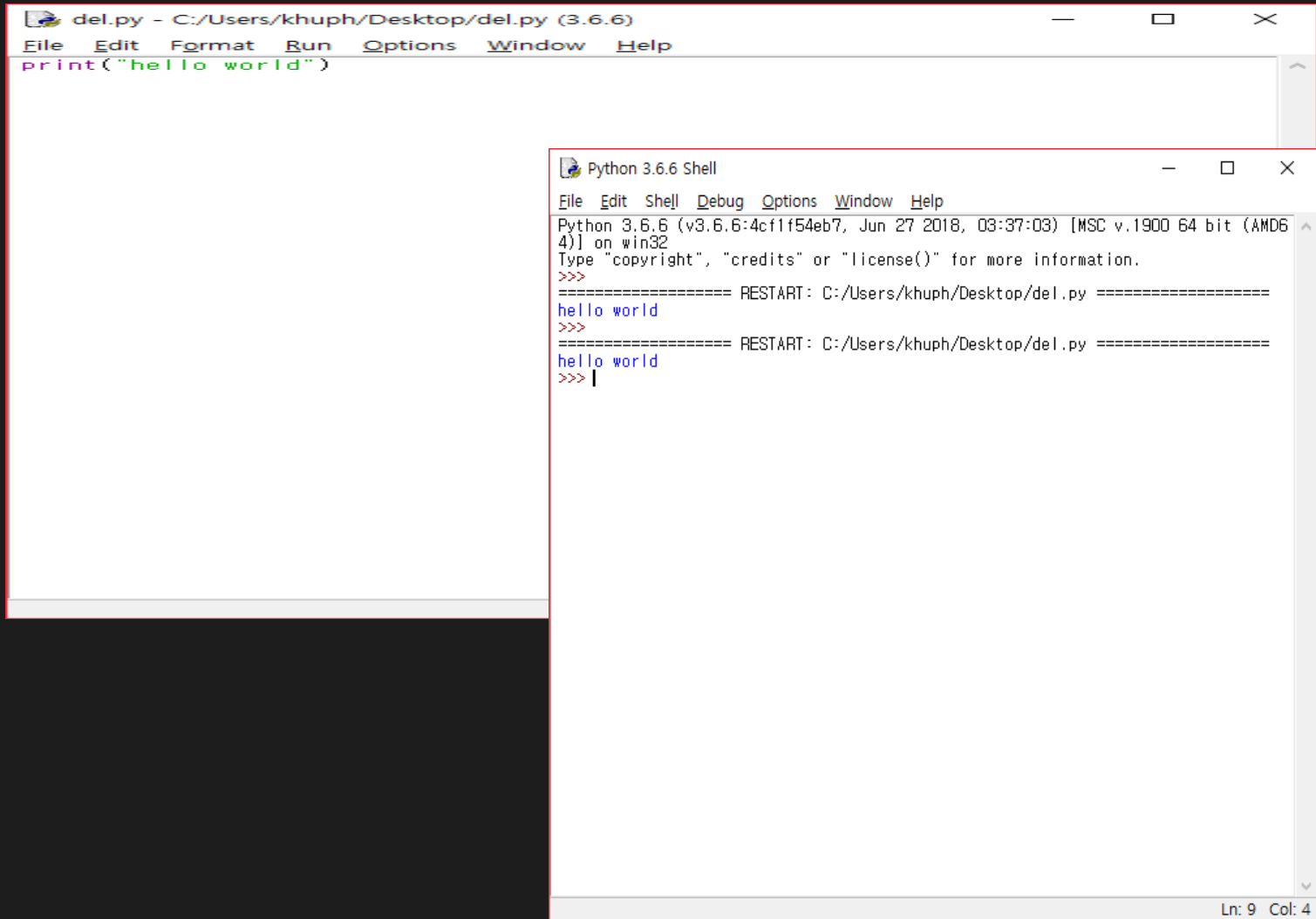
에디터를 사용해서 실행하는법

EX) pycharm community ver.

윈도우 : Ctrl + shift + F10

리눅스, 맥계열 : fn+shift+r

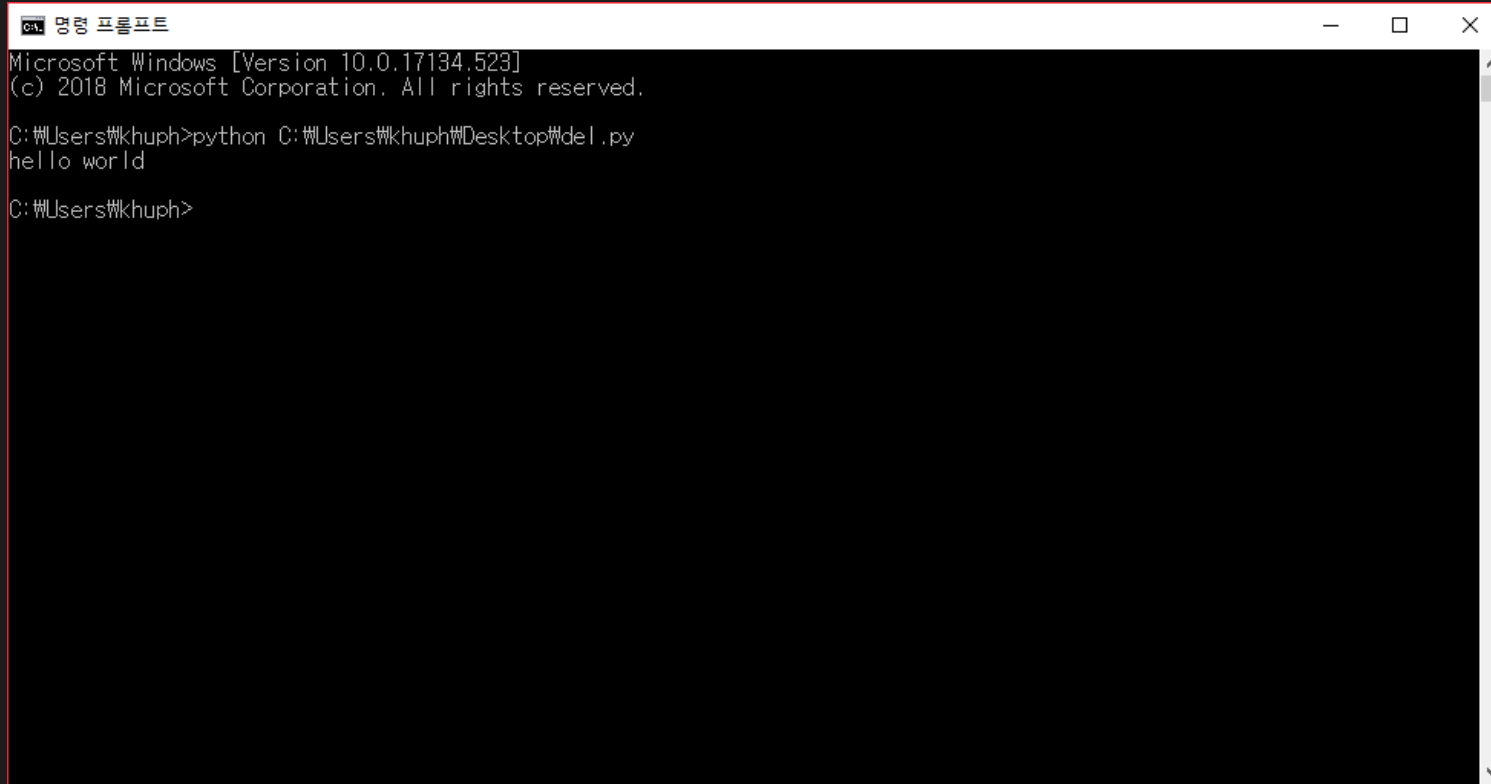
01. python 실행방법



PYTHON IDLE

File - new file 후에 - 저장후 F5로 실행

01. python 실행방법



```
명령 프롬프트
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\khuph>python C:\Users\khuph\Desktop\del.py
hello world

C:\Users\khuph>
```

CMD or Shell 로 실행

➤ Python [경로포함 파일명]

으로 입력하면 된다.

Python2, python3 버전 혼용시

```
> python2 main.py
> py -2 main.py
```

이런식으로 구분 가능

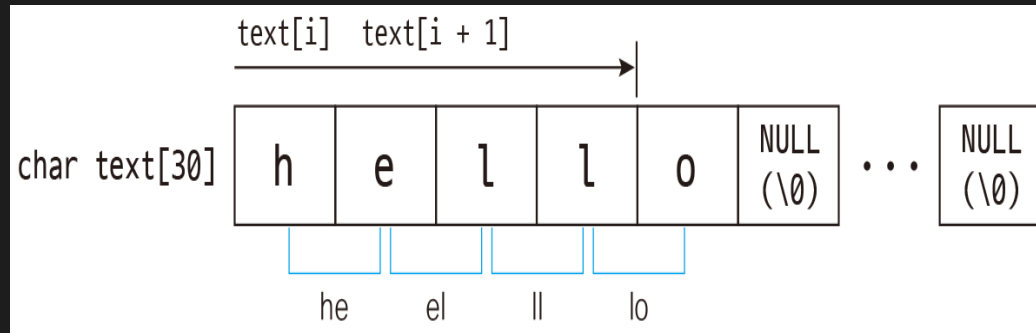
02. N-gram

= 텍스트, 바이너리 등 전체 문자열을 N 값 만큼
서브스트링 (Sub-String)으로 나누어 통계학적으로 사용한 방법

n = 1 : unigram

n = 2 : bigram

n = 3 : trigram



"쓸쓸하던 그 골목" 을 음절 단에서 **bigram** 으로 분석한 결과

['쓸', '쓸'], ['쓸', '하'], ['하', '던'], ['던', ' '], [' ', '그'], ['그', ' '], [' ', '골'], ['골', '목']

"쓸쓸하던 그 골목을 당신은 기억하십니까" 을 어절 단에서 **trigram** 으로 분석한 결과

['쓸쓸하던', '그', '골목을'], ['그', '골목을', '당신은'], ['골목을', '당신은', '기억하십니까'], ['당신은', '기억하십니까', ' '], ['기억하십니까', ' ', ' ']]

02. N-gram 장점

알고리즘이 단순하다

N-Gram인덱스 방식에는 인덱스 및 검색 알고리즘이 단순하다는 이점이 있음
고속 처리되는 것에 단어 인덱스 방식과 같은 복잡한 문서 분석을 하지 않기 때문에
어떤 언어에도 같은 알고리즘으로 대응 할 수 있습니다.

검색누락이 발생하지 않는다

N-Gram인덱스 방식에는 원래 문서에 포함되어 있는 문자라면 어떤 단어라도
검색이 되고, 검색 누락이 생기지 않는다는 이점이 있음

02. N-gram 단점

검색 노이즈가 크다

N-Gram 인덱스 방식은 검색 시 커다란 소음을 발생한다는 결점이 있음
검색 노이즈란 검색자가 의도하지 않은 웹 페이지가 검색 결과에 섞여 버리는 상태
이것은 단어 사전을 기반으로 단어 인덱스와 달리 단어가 문장에서 어떻게 다루어지고 있는지를 자세히 알 수 없기 때문에 문장의 내용을 생각한 스코어 계산힘듦

데이터베이스 용량(인덱스 크기)이 커진다

N-Gram 인덱스 방식에는 데이터베이스 용량이 커진다는 결점이 있음
(전체 문자수 \times N)의 양의 문자 정보를 유지해야하며, 단어 인덱스 방식에 비해 몇 배의 데이터양을 취급 할 필요가 있음
특히 인덱스 크기가 커진다는 결점 때문에 웹 전체를 대상으로 하는 검색 엔진에서 N-Gram 인덱스 방식이 사용되지 않게 되어 버림

02. N-gram 단점

검색 노이즈가 크다

N-Gram 인덱스 방식은 검색 시 커다란 소음을 발생한다는 결점이 있음
검색 노이즈란 검색자가 의도하지 않은 웹 페이지가 검색 결과에 섞여 버리는 상태
이것은 단어 사전을 기반으로 단어 인덱스와 달리 단어가 문장에서 어떻게 다루어지고 있는지를 자세히 알 수 없기 때문에 문장의 내용을 생각한 스코어 계산힘듦

데이터베이스 용량(인덱스 크기)이 커진다

N-Gram 인덱스 방식에는 데이터베이스 용량이 커진다는 결점이 있음
(전체 문자수 \times N)의 양의 문자 정보를 유지해야하며, 단어 인덱스 방식에 비해 몇 배의 데이터양을 취급 할 필요가 있음
특히 인덱스 크기가 커진다는 결점 때문에 웹 전체를 대상으로 하는 검색 엔진에서 N-Gram 인덱스 방식이 사용되지 않게 되어 버림

Smoothing methods

n-gram: α

- Change the freq. of occurrences
 - Laplace smoothing (add-one):

$$P_{add_one}(\alpha | C) = \frac{|\alpha| + 1}{\sum_{\alpha_i \in V} (|\alpha_i| + 1)}$$

- Good-Turing

change the freq. r to $r^* = (r + 1) \frac{n_{r+1}}{n_r}$

n_r = no. of n-grams of freq. r

03. 핵심코드설명

AI_Speaker_By_Cmd.py

```
1  # -*- coding: utf-8 -*-
2  #!/usr/bin/env python
3  # [START import_libraries]
4  from konlpy.tag import Twitter
5  import requests
6
7  twitter = Twitter()
8  # [END import_libraries]
9  '''=====Custom Parameter ====='''
10 #os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = "C:\\Users\\PHJ\\Downloads\\LoRa-d8553681a920.json"
11 myIp = 'http://192.168.0.145' # INPUT_YOUR_WEMOS_IP
12
13 urlList = [
14     myIp+'/first_floor_off',
15     myIp+'/first_floor_on',
16     myIp+'/second_floor_off',
17     myIp+'/second_floor_on',
18     '분석 실패하며 Log 를 저장합니다.'
19 ]
20 # 동사 목록
21 # frame : ['형태소원형', 'URL weight']
22 verbList = [
23     ['끄다',0],
24     ['켜다',1]
25 ]
26 # 명사 목록
27 nounList = [
28     # 명령어, 대답
29     ['위층',2],
30     ['윗층',2],
31     ['위',2],
32     ['아래층',0],
33     ['아랫층',0],
34     ['밑',0],
35 ]
36 # 1층 으로 인식될때의 1에대한 Number 형태소
37 numberList = [
38     ['1',0],
39     ['2',2]
40 ]
```

03. 핵심코드설명

AI_Speaker_By_Mic.py

```
163 """
164 리턴이 0이면 종료
165 """
166 def CommandProc(stt):
167     # 문자 양쪽 공백 제거
168     cmd = stt.strip().replace(' ', '')
169     # 입력 받은 문자 화면에 표시
170     print('나 : ' + cmd)
171
172     #NLP get weight
173     dicts = twitter.pos(cmd, norm=True, stem=True)
174
175     print(">>> 전체 형태소      : ",dicts)
176     nounWeight = getWeightNoun(dicts)
177     numberWeight = getWeightNumber(dicts)
178     verbWeight = getWeightVerb(dicts)
179     print(">>> Weight 값      : Noun = {}, Number = {}, Verb = {}".
180           format(nounWeight, numberWeight, verbWeight))
181     urlResult = getUrl(nounWeight,numberWeight,verbWeight)
182
183     # 결과
184     print(">>> GET : ",urlList[urlResult])
185     try:
186         requests.get(urlList[urlResult])
187     except:
188         pass
189     print("____"*30) # 줄 분리
190
191     return 1 # 60초 동안 계속 사용하기위해
```

03. 핵심코드설명

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

const char* ssid = "SSID";
const char* password = "PW";

ESP8266WebServer server(80);

const int LED_PIN = 14;
```

02_simple_wifi

03. 핵심코드설명

```
server.on("/", handleRoot);

//on
server.on("/on", [](){
    digitalWrite(LED_PIN, HIGH);
    Serial.println("POWER ON");
    server.send(200, "text/html", s);
});

//off
server.on("/off", [](){
    digitalWrite(LED_PIN, LOW);
    Serial.println("POWER OFF");
    server.send(200, "text/html", s);
});

server.onNotFound(handleNotFound);
```

02_simple_wifi

Setup 안에서의

wemosD1URL / (root)
 / on
 / off

Host 뒤의 on, off 등의 경로들에 대해
트리거를 달아 인라인함수로 원하는 동작을 넣음

⇒ 이 아이디어에서 영감을 받아
이 트리거를 이용해 LoRa 모듈을 제어하는 것
을 다음시간에 진행 !