



# Telegram chatbot

Assign	takhee kim  zzulu hwang
Related to 강의 일정 (자료)	<u>D05</u>
Status	In Progress

Flask 수업이 진행되었다고 가정합니다.

요청 응답 처리하는 내용은 하세요.

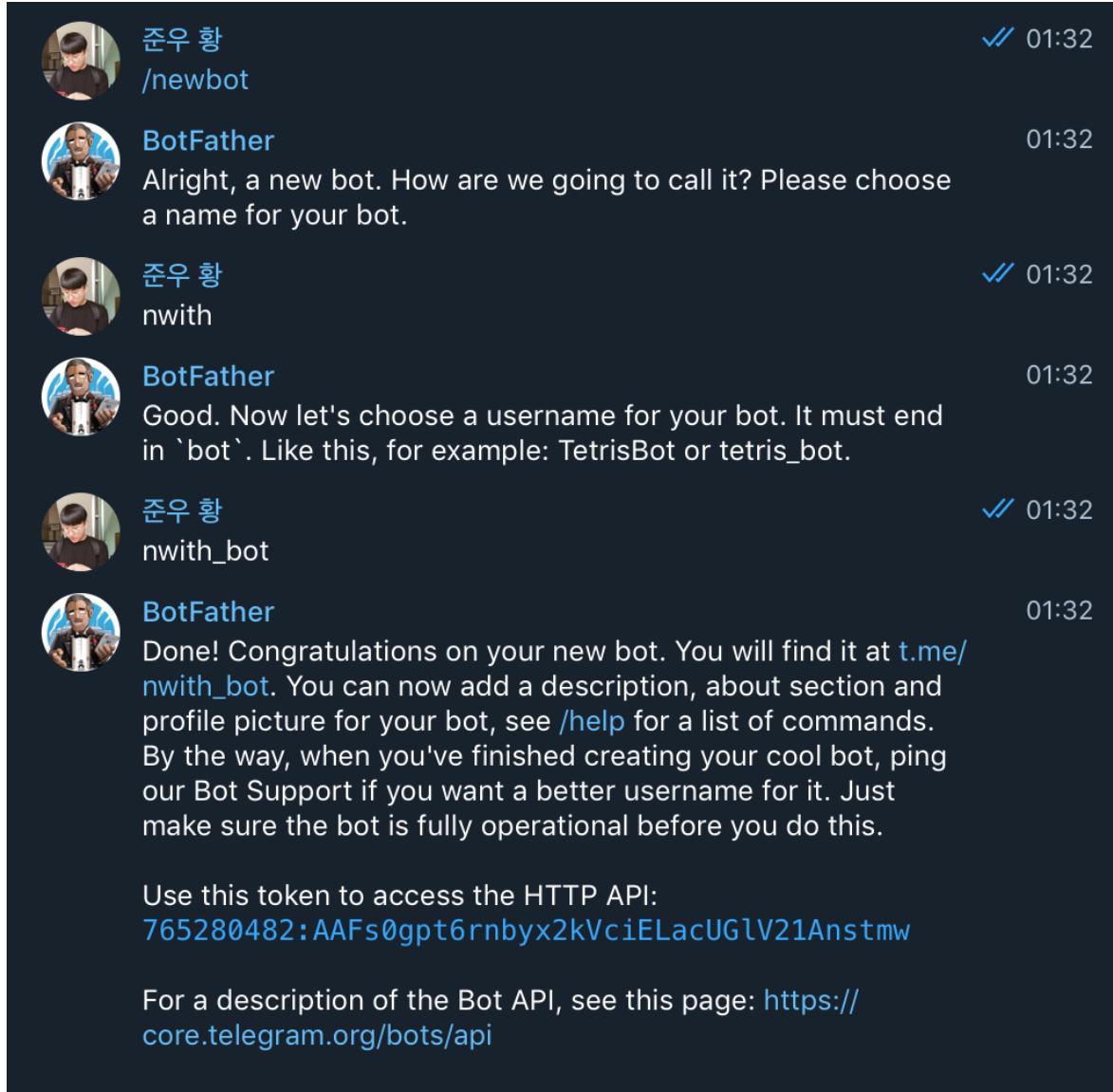
## 1. Telegram API 설정

### Telegram 가입 및 로그인

- <https://telegram.org/>
- 데스크탑 앱 or 웹 버전 사용 권장
- 전화번호 인증 통해서 가입을 해야 이용 가능함.

### bot 만들기

- BotFather이용해서 봇 만들기



- API Token은 비밀번호 같은 것이라고 설명하면서 절대 노출되지 않도록 관리
- 아래 링크는 bot api 공식 문서. api 문서 보는 법 반드시 숙지. 처음에 보면 잘 못 찾음.

오른쪽에 있는 ToC 활용할 것.

## 2. Telegram 요청 보내기

### Telegram 요청 기초

<https://core.telegram.org/bots/api#making-requests> → 위의 문서는 구글 검색으로 부터 찾아가는 것 연습 안하면 수업시간에 직접 못 합니다. 찾기 힘들어요.

- Authorizing Your Bot

읽어볼게요, 봇이 만들어지면 동시에 생성되는 토큰을 통해서 권한을 인증한다고 하네요. 그럼 요청을 한번 보내볼까요.

- Making requests

HTTPS 요청을 통해 처리하고 URL은 이런식으로 만들어진다고 합니다.

```
https://api.telegram.org/bot<token>/METHOD_NAME
기본 도메인
bot 토큰
메소드
```

한번 아래의 URL로 요청을 보내 봅시다. GET 요청은 브라우저를 통해서 보낼 수 있죠!

여러분의 토큰으로 대체 해야합니다.

```
https://api.telegram.org/bot123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11/getMe
```

```
https://api.telegram.org/bot513474581:AAEyIoPU0ecLl70KaaDg0zG7KfakmOsTu_4/getMe
```

```
{
  "ok": true,
  "result": {
    "id": 513474581,
    "is_bot": true,
    "first_name": "connectedtak",
    "username": "connectedtakbot"
  }
}
```

앞으로 해당 URL 구조를 통해 요청을 보내야하므로 파일 코드로 옮겨 봅시다.

```
# telegram.py
# 0. 기본 설정
token = '토큰'
api_url = f'https://api.telegram.org/bot{token}'
```

## 사용자 chat\_id 찾기

- 나에게 메세지를 보내려면, telegram에서 지정한 내 고유 id값을 알아야 함.
- 우리는 bot의 관리자이니 해당 bot으로 누군가가 message를 보내면, 그 사람의 chat\_id를 알아낼 수 있음. 그래야 bot이 그 사람에게 메세지를 보내줄테니.
- 텔레그램 클라이언트에서 내가 생성한 bot을 이름으로 검색해서 찾은 다음 **start** 버튼을 누르고(그러면 '/start'라는 메세지가 전송됨.), **브라우저에서 아래 주소로 접속**
- `getUpdates` 는 bot이 받은 모든 메세지를 확인 가능한 method임.

<https://core.telegram.org/bots/api#getupdates>

```
https://api.telegram.org/bot<token>/getUpdates
```

```
{
  "ok": true,
  "result": [
    {
      "update_id": 624804488,
      "message": {
        "message_id": 29,
        "from": {
          "id": 388821535,
```

```
        "is_bot": false,
        "first_name": "\u043f\u043e\u0437\u043d\u0430",
        "username": "romdan",
        "language_code": "ko"
    },
    "chat": {
        "id": 388821535,
        "first_name": "\u043f\u043e\u0437\u043d\u0430",
        "username": "romdan",
        "type": "private"
    },
    "date": 1559097997,
    "text": "\ud55c\ud55c"
}
]
```

- 결과는 `JSON` 으로 나오며, 분석해보면 다음과 같을 것이다.
    - `from` : 누구로부터 왔는지. 이 사람이 bot인지. 이름이 무엇이고, username이 무엇인지.
    - `chat` : 메세지 정보
    - `date` : 보낸 날짜 (이상한 숫자로 보이겠지만, 2019년 5월 30일 현재 시간을 컴퓨터에 저장을 하거나 표기하는 방법을 고안해낸 것.) `Unix Time` 혹은 `Epoch Time` 이라고 불림.

## ▼ 관련 자료

<https://ko.wikipedia.org/wiki/유닉스> 시간

유닉스 시간(영어: Unix time)은 시각을 나타내는 방식이다. POSIX 시간이나 Epoch 시간이라고 부르기도 한다. **1970년 1월 1일 00:00:00 협정 세계시(UTC) 부터의 경과 시간을 초로 환산하여 정수로 나타낸 것이다.**[1][note 1] 유닉스 시간에서 윤초는 무시된다.[1][2][note 2] 유닉스 계열 운영 체제나 여러 다른 운영 체제, 그리고 파일 형식들에서 사용된다.

개발자들은 쓸데 없는 것을 기념한다.



유닉스 시간 10억초 달성 기념 행사. (2001-09-09T01:46:40Z) 덴마크 유닉스 사용자 그룹이 현지 시간 03:46:40에 코펜하겐에서 기념하였다.

- `text` : 내용
  - 그러면 이제 우리는 파일로 메세지를 보내야 하니 요청을 통해 사용자의 채팅 id값을 충출 해보자

```
# import requests
## 0. 기본 설정
# token = "토Ken"
# api_url = f'https://api.telegram.org/bot{token}'

# 1. 사용자 정보 가져오기 위한 요청
update_url = f'{api_url}/getUpdates'
response = requests.get(update_url).json()
```

```
# 2. 사용자의 채팅 id 출력
chat_id = response['result'][0]['message']['from']['id']
print(chat_id)
```

## Telegram이 가지고 있는 정보의 Type (Skip 가능)

- Available Types (<https://core.telegram.org/bots/api#available-types>)

JSON으로 표기되는 다양한 응답 정보들에 대해서 담겨있다.

User와 Chat, Message는 직전에 우리가 `getUpdates` 를 통해 응답 받은 정보에 담겨 있었고, 상황에 따라 해당하는 내용들을 받을 수 있다.

## Telegram에서 쓸 수 있는 Methods

- Available Methods (<https://core.telegram.org/bots/api#available-methods>)

HTTP 요청만으로 다양한 기능을 수행하도록 만들 수 있다.

`getMe`는 우리가 했었던 내용이고, 다음에 있는 메시지를 보내기(`sendMessage`)를 해보자.

## 지정 사용자에게 메세지 보내기(`sendMessage`)

- 기본 URL구조는 이렇게 된다고 하였다.

```
https://api.telegram.org/bot<token>/METHOD_NAME
```

- 그리면 아래와 같이 만들어 볼 수 있다.

```
https://api.telegram.org/bot<token>/sendMessage
```

- 브라우저를 통해 요청 보내보면, 다음과 같은 결과를 볼 수 있다.

```
{
  "ok": false,
  "error_code": 400,
  "description": "Bad Request: message text is empty"
}
```

- `getMe` 와는 다르게 `sendMessage` 는 파라미터를 넘겨줘야 한다고 문서에 적혀있다!

네이버 검색 혹은 구글 검색 안한 사람들은 다시 한번 하면서 짚어 주세요.

특히, `required` 가 `True` 인 경우에는 반드시 필요한 것이다.

- 아래의 URL을 통해 브라우저를 통해 요청을 보내보자.

- token을 가진 bot이 chat\_id를 가진 user에게 text를 전달! 한다는 의미.

```
https://api.telegram.org/bot<token>/sendMessage?chat_id=<chat_id>&text=안녕하세요
```

- 텔레그램으로 메시지가 온 것을 확인할 수 있다.

- Python으로 옮겨보자.

```
import random
# import requests
# 0. 기본 설정
```

```

# token = 'токен'
# api_url = f'https://api.telegram.org/bot{token}'

## 1. 사용자 정보 가져오기 위한 요청
# update_url = f'{api_url}/getUpdates'
# response = requests.get(update_url).json()

## 2. 사용자의 채팅 id 출력
# chat_id = response['result'][0]['message']['from']['id']
# print(chat_id)

## 3. 메세지 전송
send_url = f'{api_url}/sendMessage?chat_id={chat_id}&text=안녕하세요'
requests.get(send_url)

```

- 한번 로또 번호를 추천해주는 메시지를 보내보자.

```

import random
# import requests
# 0. 기본 설정
# token = 'токен'
# api_url = f'https://api.telegram.org/bot{token}'

## 1. 사용자 정보 가져오기 위한 요청
# update_url = f'{api_url}/getUpdates'
# response = requests.get(update_url).json()

## 2. 사용자의 채팅 id 출력
# chat_id = response['result'][0]['message']['from']['id']
# print(chat_id)

## 3. 메세지 전송
text = random.sample(range(1, 46), 6)
send_url = f'{api_url}/sendMessage?chat_id={chat_id}&text={text}'
requests.get(send_url)

```

### 3. 자유롭게 강의 내용 추가

- form input 받아서 그대로 메시지 보내기
  - 이거 하면서 자연스럽게 Flask로 플랫폼 갈아타자!
  - write

```

# app.py
@app.route('/write')
def send():
    return render_template('write.html')

```

```

<form action="/send">
    <input type="text" name="message">
    <input type="submit">
</form>

```

- send

```

# app.py
from flask import request
import requests

token = 'токен'
api_url = f'https://api.telegram.org/bot{token}'
chat_id = '...'

@app.route('/send')
def send():
    message = request.args.get('message')
    send_url = f'{api_url}/sendMessage?chat_id={chat_id}&text={message}'

```

```
    requests.get(send_url)
    return '전송 완료!' # html 안 써도 될 듯.
```

- 했던 코드 옮기기

등등

## 4. Flask 서버를 통해 메시지 보내기(선택)

- 플라스크 서버를 통해 `/telegram` 으로 요청이 들어오면 처리하도록 해보자.

```
# app.py

import random
import requests
from flask import Flask

# 0. 기본 설정
app = Flask(__name__)
token = 'токен'
api_url = f'https://api.telegram.org/bot{token}'

@app.route('/telegram')
def telegram():
    # 1. 사용자 정보 가져오기 위한 요청
    update_url = f'{api_url}/getUpdates'
    response = requests.get(update_url).json()

    # 2. 사용자의 채팅 id 출력
    chat_id = response['result'][0]['message']['from']['id']
    print(chat_id)

    # 3. 메세지 전송
    text = random.sample(range(1, 46), 6)
    send_url = f'{api_url}/sendMessage?chat_id={chat_id}&text={text}'
    requests.get(send_url)

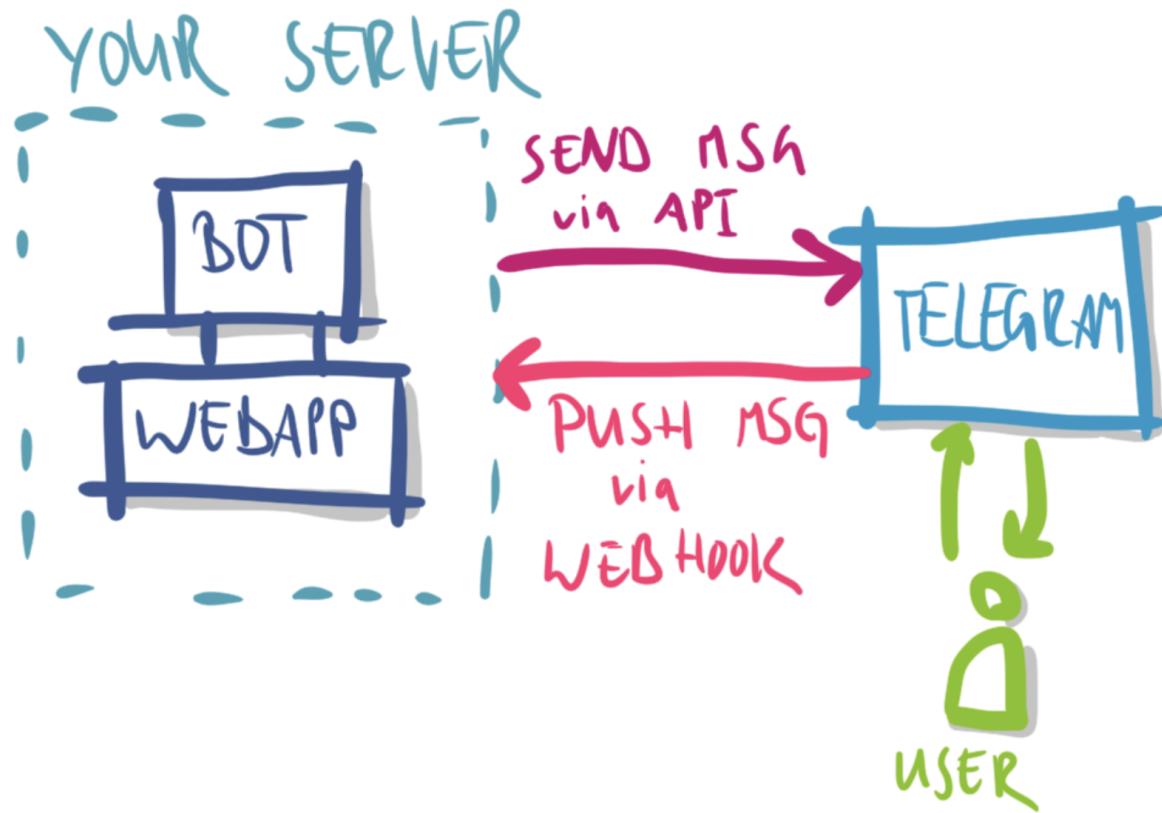
if __name__ == '__main__':
    app.run(debug=True)
```

```
$ python app.py
```

## 5. Telegram webhook 설정하기

- 갑자기 얘가 왜 필요? 라고 한다면...
  - 이 때까지는 우리가 bot 빙의해서, bot token을 가지고 bot 행세를 하면서 bot 신분으로, 단 한사람(나 자신)에게 메세지를 보내는 것만 했는데, 즉 우리가 디테일하게 이렇게 이렇게 해! 라고 시키는 것만 했는데 우리가 디테일하게 시키는 것만 하는 건 bot스럽지 않다.
  - 우리가 bot 한테, `야!`라고 이야기하면, `왜!`라는 반응이 튀어나와야 bot스럽지 않을까?!
  - 누군가 bot 한테 `야!`라고 말한(telegram app에서 메시지를 입력한) 사실은 telegram이 알고 있고, telegram이 bot에게 말한 내용을 우리에게도 전파 해줘야지, 우리가 코드(python)를 통해서 `왜!`라고 그 사람한테 (bot 빙의를 해서) 대답을 해줄 수 있다!
  - 이 때, 우리한테 전파해주는 행위가 webhook의 개념이다.
- webhook 개념 설정
  - 서버에서 무언가가 실행되었을 때, 실행되었음을 (HTTP POST를 통하여) 알리는 개념. URL의 설정이 요구되며 URL을 설정하면 그 URL로 알려준다.

- Telegram에서는, Telegram 서버가 클라이언트로부터 메세지를 받으면 그에 대한 정보를 webhook으로 등록된 주소로 알려준다.



### 플라스크 서버 구조 만들기

- telegram 서버에서 웹훅에 등록된 URL을 통하여 post request를 보낸다.
- Flask를 통해서 해당 요청을 받아서 처리하는 로직을 구현 하면 된다.
- response는 반드시 status code 200을 리턴하도록 한다. 그렇지 않으면 200 응답이 올때까지 텔레그램이 request를 보낸다. body의 내용은 무엇을 넣어도 상관없다.
  - `return body, status_code` 의 구조.
- token을 route에 붙이는 이유는 아무나 해당 url로 접근하는 것을 막기 위함이다. 만약 /telegram 같은 단순한 url을 사용한다면 주소를 아는 누군가가 bot을 사칭하거나 실제 메시지가 오지 않았는데도 온 것처럼 속일 수 있다.

```
# app.py
from flask import Flask

# 0. 기본 설정
app = Flask(__name__)
token = 'токен'
api_url = f'https://api.telegram.org/bot{token}'

@app.route(f'/{token}', methods=['POST'])
def telegram():
    return '', 200

if __name__ == '__main__':
    app.run(debug=True)
```

```
$ python app.py
```

- 이제 이 플라스크 서버를 플라스크에게 알려줘야 한다.
- 근데 우리는 지금 로컬에서 `localhost` 를 통해 서버를 구동하고 있기 때문에 텔레그램이 요청을 보낼 수가 없다.

### ngrok 을 활용한 서버 url 생성

- 공식 문서에 있는 ngrok 읽는 법

## How do I pronounce ngrok?

*en-grok*

<https://ngrok.com/> 에서 download 받은 .exe 파일을 app.py와 동일 디렉토리에 위치 시킨다.

그리고 회원가입을 통해 토큰 값을 받는다. `connect your account` 부분 복사 및 터미널 붙여 넣기.

새로운 터미널에서 진행!

가입하지 않더라도 가능하지만, 세션 유지 8시간 제한 있다고 합니다. 가입 추천

The screenshot shows the ngrok website's setup guide. On the left, there's a sidebar with links like Status, Reserved, Auth, Team, Admin, and Billing. Below that is a link to 'Want more from ngrok? Upgrade now'. The main content area is titled 'Setup & Installation' and contains four numbered steps:

- 1 Download ngrok**: A section explaining that ngrok is easy to install and provides a 'Download for Mac OS X' button. Below the button are links for Windows, Linux, Mac (32-Bit), Windows (32-Bit), Linux (ARM), Linux (32-Bit), FreeBSD (64-Bit), and FreeBSD (32-Bit).
- 2 Unzip to install**: A section for Linux or OSX users, explaining how to unzip ngrok from a terminal. It includes a command line example: `$ unzip /path/to/ngrok.zip`. It also notes that most people keep ngrok in their user folder or set an alias for easy access.
- 3 Connect your account**: A section for connecting to an account. It explains that running the command adds the account's auth token to ngrok.yml. It includes a command line example: `$ ./ngrok authtoken 3feR1pS2cM9xXCrTFyoKY_4T37UDG1rg9V8taZrxh4L`.
- 4 Fire it up**: A section for starting an HTTP tunnel. It links to documentation and provides command line examples for both Mac OS X and Linux: `$ ./ngrok http 80`.

### ▼ 멀티캠퍼스 혹은 윈도우 일부 컴퓨터에서 permission denied 또는 경우

ngrok download 검색 > <https://dl.equinox.io/> 사이트 접속 > archive > Version 2.2.8

그리고 git bash 관리자 권한으로 실행하면 해결 되는 것까지 확인하였습니다.

- 설정

```
$ ./ngrok authtoken 3feR1pS2cM9xXCrTFyoKY_4T37UDG1rg9V8taZrxh4L
```

- `ngrok.yml` 파일을 자동으로 생성해 줌. `~/.ngrok2/ngrok.yml`
- 이제 우리가 활용하고 있는 포트를 연결한다.

- 주의!! flask는 기본이 5000으로 돌리고 있기 때문임.

```
$ ngrok http 5000
```

```
# 결과
Web Interface          http://127.0.0.1:4040
Forwarding             http://cbe0b9ca.ngrok.io -> http://localhost:5000
Forwarding             https://cbe0b9ca.ngrok.io -> http://localhost:5000
```

- 주의!!!! 해당하는 URL은 free plan에서는 계속 변동됨. 따라서 만약 다시 실행하는 경우에 이후에 하는 webhook 설정을 다시 해 줄 것.
- 해당 터미널 반드시 계속 유지할 것

### flask - ngrok 연결 확인

- 반드시 새로운 터미널 탭을 열어서 명령어 진행할 것.

```
$ python app.py
```

- <http://cbe0b9ca.ngrok.io>로 들어가보면 Not Found 뜨는데 flask 서버로그에 요청이 들어오는 것 확인할 수 있음.

### 텔레그램 webhook 설정

<https://core.telegram.org/bots/api#setwebhook>

문서를 보면 아래와 같은 URL 구조를 만들어야 하는 것을 알 수 있다.

```
https://api.telegram.org/bot<token>/setWebhook?url=<flask_url>
```

- webhook URL을 등록할 땐, 반드시 [https](https://) 주소를 사용하여야 한다.

```
// 20190529152824
// https://api.telegram.org/bot656712403:AAGe53HAhDMU1c6ZHLjdgwRaX2fEkEqqFf0/SetWebhook?url=https://f8f45bb2.ngrok.io/656712403:

{
    "ok": true,
    "result": true,
    "description": "Webhook was set"
}
```

- webhook was set 키워드를 반드시 확인하자!
- 덧) 저는 강의할 때 이렇게 그냥 만들어 놓고 돌려서 붙여놓고 돌려서 나온 것 크롬 붙여넣기 시켰습니다.

```
# set_webhookurl.py
# 0. 기본 설정
token = 'токен'
api_url = f'https://api.telegram.org/bot{token}'
webhook_url = input()
print(f'{api_url}/setWebhook?url={webhook_url}')
```

### 메세지 받아보기

- 플라스크랑 ngrok 계속 켜져 있어야 함.

- 텔레그램 봇에 대화를 해보면 서버 로그가 뜨는 것을 확인할 수 있다.

```
127.0.0.1 - - [29/May/2019 11:26:18] "POST /656712403:AAGe53HAhDMU1c6ZHLjdgwRaX2fEkEqqFf0 HTTP/1.1" 200 -
```

## 6. 기능1) 단순 echo

- 들어온 메세지를 그대로 사용자에게 돌려보내는 코드를 작성 해보자.
- telegram이 보내는 request json 구조가 어떻게 생겼는지 분석
- 2 step으로 나눠서 해볼 것.
- `.get('message')` 말고 그냥 `['message']` 으로 해도 동작하나, 예외 상황을 고려하여 전자의 방식 사용 **강력 권장**

```
from flask import Flask, request

# 0. 기본 설정
app = Flask(__name__)
token = 'токен'
api_url = f'https://api.telegram.org/bot{token}'

@app.route(f'/{token}', methods=['POST'])
def telegram():
    # step 1. 구조 print 해보기
    print(request.get_json())
```

- 이제 이 구조를 참고하여, 응답하는 코드를 작성해 보세요.

```
import requests
# from flask import Flask, request

# # 0. 기본 설정
# app = Flask(__name__)
# token = 'токен'
# api_url = f'https://api.telegram.org/bot{token}'

# @app.route(f'/{token}', methods=['POST'])
# def telegram():
#     # step 1. 구조 print 해보기
#     print(request.get_json())
#     # step 2. 그대로 돌려보내기
#     message = request.get_json().get('message')
#     print(message)
#     if message is not None:
#         chat_id = message.get('from').get('id')
#         text = message.get('text')
#         res = requests.get(f'{api_url}/sendMessage?chat_id={chat_id}&text={text}')
#     return '', 200
```

## 7. 기능2) 로또 번호 추천

- **로또**라는 단어가 들어오면, 로또 번호를 추천 하는 코드를 작성하자.
- 기본적으로는 메아리이도록 구성함. 편하게 하세요.

```
def telegram():
    # step 1. 구조 print 해보기
    print(request.get_json())
    # step 2. 그대로 돌려보내기
    message = request.get_json().get('message')
    if message is not None:
        chat_id = message.get('from').get('id')
        text = message.get('text')
        # 로또
```

```

if '로또' == message.get('text'):
    text = random.sample(range(1, 46), 6)
    res = requests.get(f'{api_url}/sendMessage?chat_id={chat_id}&text={text}')
    return '', 200

```

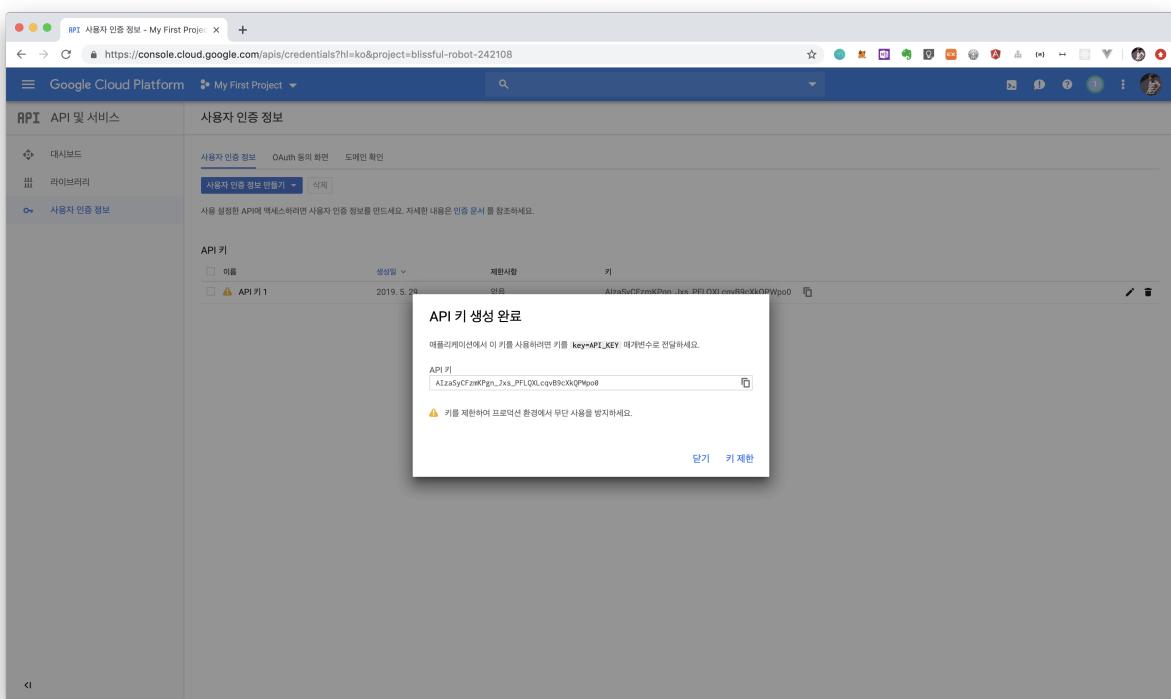
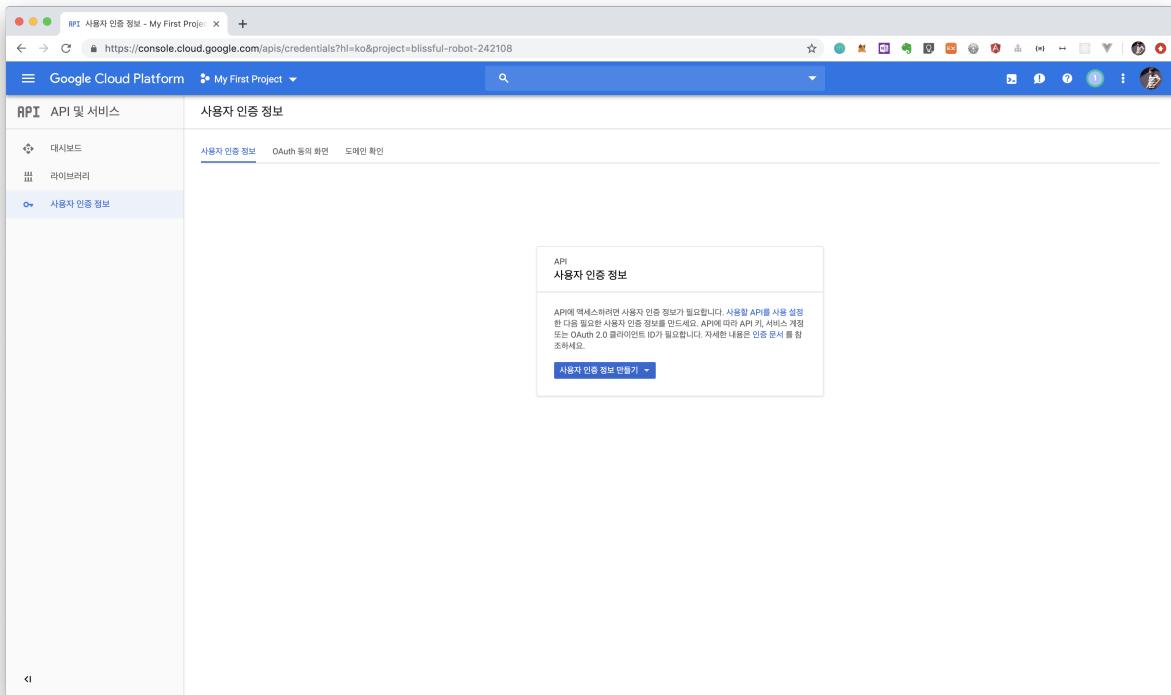
## 8. 기능3) 번역

- 네이버 번역해도 되는데 헤더 설정 안해도 된다는 장점이 있습니다.
- 안녕하세요 말고 재밌는 키워드 알려주세요!
- google cloud platform에서 키를 발급 받는다.

The screenshot shows the Google Cloud Platform API library interface for the Cloud Translation API. At the top, there's a navigation bar with tabs like 'API 및 서비스', 'My First Project', and 'API 라이브러리'. Below the navigation, there's a search bar and a sidebar with project details: 'Cloud Translation API' by 'Google', 'Integrates text translation into your website or application.', and a 'Cloud Translation API' icon. The main content area displays the API details:

유형	개요												
API 및 서비스	Integrates text translation into your website or application.												
최종 업데이트	19. 4. 17. 오후 8:56												
카테고리	미분류												
서비스 이름	translate.googleapis.com												
가격	<table border="1"> <thead> <tr> <th>AutoML Translation Model Training Operations</th> <th>무료</th> <th>\$ 80,00</th> </tr> </thead> <tbody> <tr> <td>AutoML Translation Online Predictions Operations</td> <td>각각/1M characters</td> <td>각각/1M characters</td> </tr> <tr> <td>Neural Translation Model Online Predictions</td> <td>0 - 500K characters/기밀</td> <td>500K+ characters/기밀</td> </tr> <tr> <td>Neural Translation Model Online Predictions In Translation V3</td> <td></td> <td></td> </tr> </tbody> </table> <p><small>참고: API를 호출하는 데 사용한 인프리에서도 추가 요금이 부과될 수 있습니다.. USD 외의 통화로 지불하면 Cloud Platform SKU에 해당 통화로 표기된 가격이 적용됩니다. 최근 가격 정보는 <a href="#">GCP 가격 목록</a>을 참조하세요.</small></p>	AutoML Translation Model Training Operations	무료	\$ 80,00	AutoML Translation Online Predictions Operations	각각/1M characters	각각/1M characters	Neural Translation Model Online Predictions	0 - 500K characters/기밀	500K+ characters/기밀	Neural Translation Model Online Predictions In Translation V3		
AutoML Translation Model Training Operations	무료	\$ 80,00											
AutoML Translation Online Predictions Operations	각각/1M characters	각각/1M characters											
Neural Translation Model Online Predictions	0 - 500K characters/기밀	500K+ characters/기밀											
Neural Translation Model Online Predictions In Translation V3													

At the bottom, there's a link to '기이드 및 문서'.



- 키를 잘 관리하자!

- 이제 공식문서를 통해 활용법을 알아보자.

<https://cloud.google.com/translate/docs/reference/rest/v2/translate?hl=ko>

```
POST https://translation.googleapis.com/language/translate/v2
```

따라서, POST 요청을 보내야 하며, 항상 API Key를 key에 담아줘야 한다.

필수적인 파라미터는 다음과 같다.

- `q` : 필수. 번역 대상 문장
  - `target` : 필수. 번역 대상 언어
  - `source` : 선택. 입력언어.
- 코드를 작성 해보자. (post 가 더 간단한 느낌. 공식 문서에서도 표기하고 있음.)

```
# google.py
import requests
key = 'AIzaSyCFzmkPgn_Jxs_PFLQXLcqvB9cXkQPWpo0'
url = 'https://translation.googleapis.com/language/translate/v2'
data = {'q': '안녕하세요.',
        'target': 'en',
        'source': 'ko'}
# response = requests.get(f'{url}?q=안녕하세요&target=en&source=ko&key={key}'.json()
response = requests.post(f'{url}?key={key}', data).json()
print(response)
```

```
{'data': {'translations': [{'translatedText': 'Hi.'}]}}
```

- 추가로 재밌는 것 있으면, 더 해보고 가면 좋을 듯 합니다.
- 이제 flask를 통해 챗봇에 적용 하자.
- 모든 말을 다 번역할 수 없으니, 챗봇처럼 `/번역`이라고 들어오면 그 다음 말을 번역하자.

```
...
if '/번역' == message.get('text')[0:4]:
    key = 'AIzaSyCFzmkPgn_Jxs_PFLQXLcqvB9cXkQPWpo0'
    url = 'https://translation.googleapis.com/language/translate/v2'
    data = {'q': '안녕하세요.',
            'target': 'en',
            'source': 'ko'}
    }
    response = requests.post(f'{url}?key={key}', data).json()
    text = response.get('data').get('translations')[0].get('translatedText')

    res = requests.get(f'{api_url}/bot{token}/sendMessage?chat_id={chat_id}&text={text}')
```

- 이제 들어온 값을 받아서 처리하자.

```
data = {'q': message.get('text')[4:],  
...}
```

## 크롤링 결과

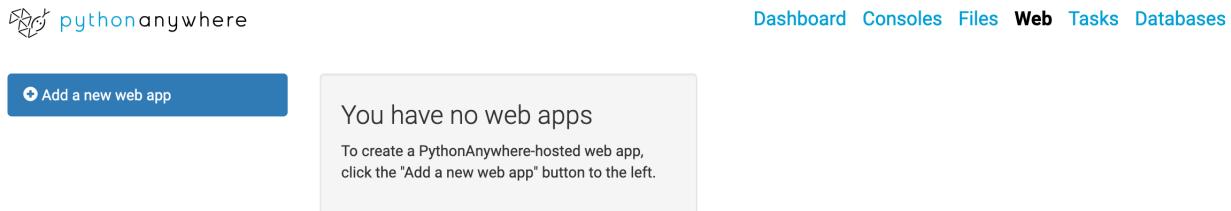
### 추가 컨텐츠 (이미지)

## Pythonanywhere 배포

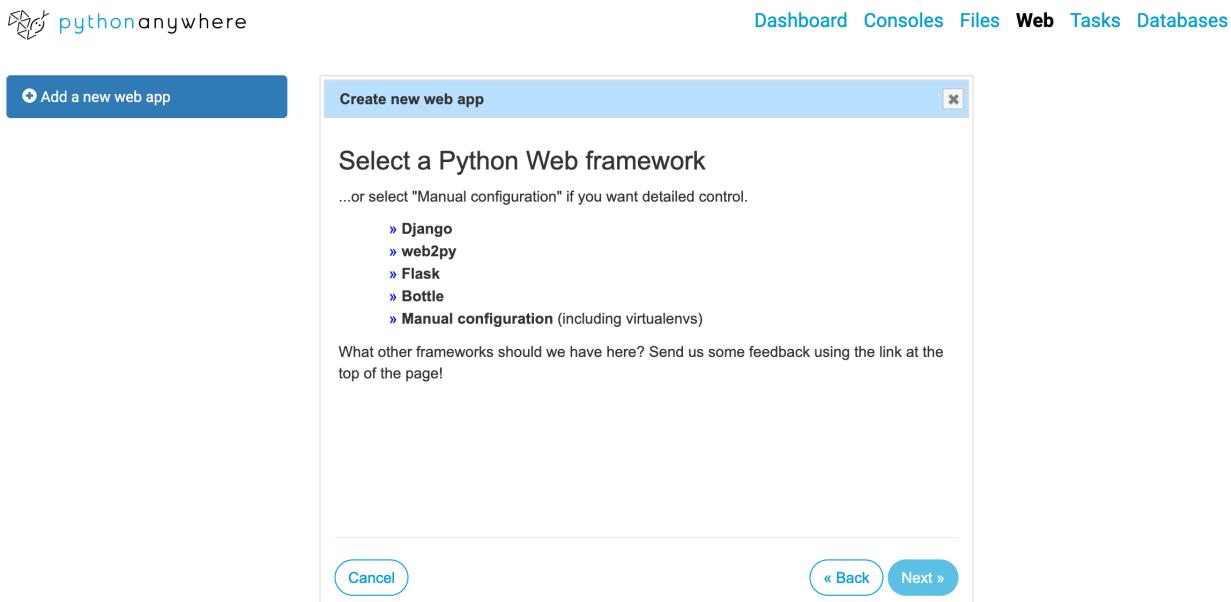
- pythonanywhere 가입하기 / 메일인증 / 로그인 까지 완료

- 대시보드 상단의 **web** 탭 이동

- Add a new web app** 클릭



- next 후 Flask 클릭



- 파이썬 버전 선택 (테스트에서는 3.7 사용)

## Create new web app

# Select a Python version

- » **Python 2.7 (Flask 1.0.2)**
- » **Python 3.4 (Flask 1.0.2)**
- » **Python 3.5 (Flask 1.0.2)**
- » **Python 3.6 (Flask 1.0.2)**
- » **Python 3.7 (Flask 1.0.2)**

**Note:** If you'd like to use a different version of Flask to the one listed here, you can use a virtualenv for your web app. There are [instructions here](#).

- 파일이름 설정하기 (디플트 설정으로 안바꾸고 넘어가도 무난)

## Create new web app



# Quickstart new Flask project

Enter a path for a Python file you wish to use to hold your Flask app. If this file already exists, its contents will be overwritten with the new app.



Path

/home/eduChang/mysite/flask\_app.py

- `{username}.pythonanywhere.com` 주소로 요청보내서 배포 확인!
- 위에서 개발한 `app.py` 를 여기에 복붙해야함
- 상단의 `Files` 탭으로 이동 ⇒ 왼쪽의 `Directories` 의

The screenshot shows the PythonAnywhere dashboard with the following details:

- Dashboard**, **Consoles**, **Files** (highlighted), **Web**, **Tasks**, **Databases**
- File path: /home/eduChang
- Space usage: 0% full - 80.0 KB of your 512.0 MB quota
- Directories** section: Shows .local/ and mysite/ with icons for upload and delete.
- Files** section: Shows a list of files:
 

File	Last Modified	Size
.bashrc	2019-05-29 10:48	559 bytes
.gitconfig	2019-05-29 10:48	266 bytes
.profile	2019-05-29 10:48	79 bytes
.pythonstartup.py	2019-05-29 10:48	77 bytes
.vimrc	2019-05-29 10:48	4.6 KB
README.txt	2019-05-29 10:48	232 bytes
- Buttons: **New directory**, **New file**, **Upload a file** (with 100MB maximum size note).

- 디폴트로 만들어져있던 `flask_app.py` 클릭
- 전체 코드 바꿔주기 후 `Save`

The code editor shows the `flask_app.py` file content:

```

1 import requests
2 import random
3 import random
4 import requests
5 from flask import Flask, request
6
7 # app.py
8
9
10 # 0. 기본 설정
11 app = Flask(__name__)
12 token = '860908348:AAFGvomm1V1J7P5Yw3_72VNaTF0jPgVI4gw'
13 api_url = f'https://api.telegram.org/bot{token}'
14
15 @app.route('/telegram')
16 def telegram():
17     # 1. 사용자 정보 가져오기 위한 요청
18     update_url = f'{api_url}/getUpdates'
19     response = requests.get(update_url).json()
20
21     # 2. 사용자의 채팅 id 출력
22     chat_id = response['result'][0]['message']['from']['id']
23     print(chat_id)
24
25     # 3. 메세지 전송
26     text = random.sample(range(1, 46), 6)
27     send_url = f'{api_url}/sendMessage?chat_id={chat_id}&text={text}'
28
29

```

- 코드 수정 `save`를 한다고 바로 반영안됨
- 대시보드의 **Web** 탭 클릭 후 나오는 Reload 버튼 눌러야 서버 다시 시작됨

eduChang.pythonanywhere.com

Configuration for eduChang.pythonanywhere.com

[Add a new web app](#)

Reload:

[Reload eduChang.pythonanywhere.com](#)

Best before date:

We're happy to host your free website – and keep it free – for as long as you want to keep it running, but you'll need to log in at least once every three months and click the "Run until 3 months from today" button below. We'll send you an email a week before the site is disabled so that you don't forget to do that. [See here for more details.](#)

- 라이브러리 설치

- Consoles > Bash

- 여기에는 파이썬 버전 2도 설치되어 있어서, 파이썬 3를 사용하려면 `python3`을 사용해야하고 `pip3`로 입력해야 `python3`의 `pip`을 사용하게 된다.

```
$ pip3 install decouple
```

- 유명한 것들은 이미 있음. 추가적으로 설치가 필요할 때 사용!

- 웹훅 바꿔주기

- 위에서 웹훅 설정한 것 처럼 서버가 바뀌었으니 다시 웹훅을 설정해줘야 함
  - 주의!!!!!! https로 설정해야함
  - ex) `https://educhang.pythonanywhere.com/`

```
# set_webhookurl.py
# 0. 기본 설정
token = 'токен'
api_url = f'https://api.telegram.org/bot{token}'
webhook_url = input()
print(f'{api_url}/setWebhook?url={webhook_url}')
```

- 죽지 않는 서버 생겼다!