

CHATBOTHON

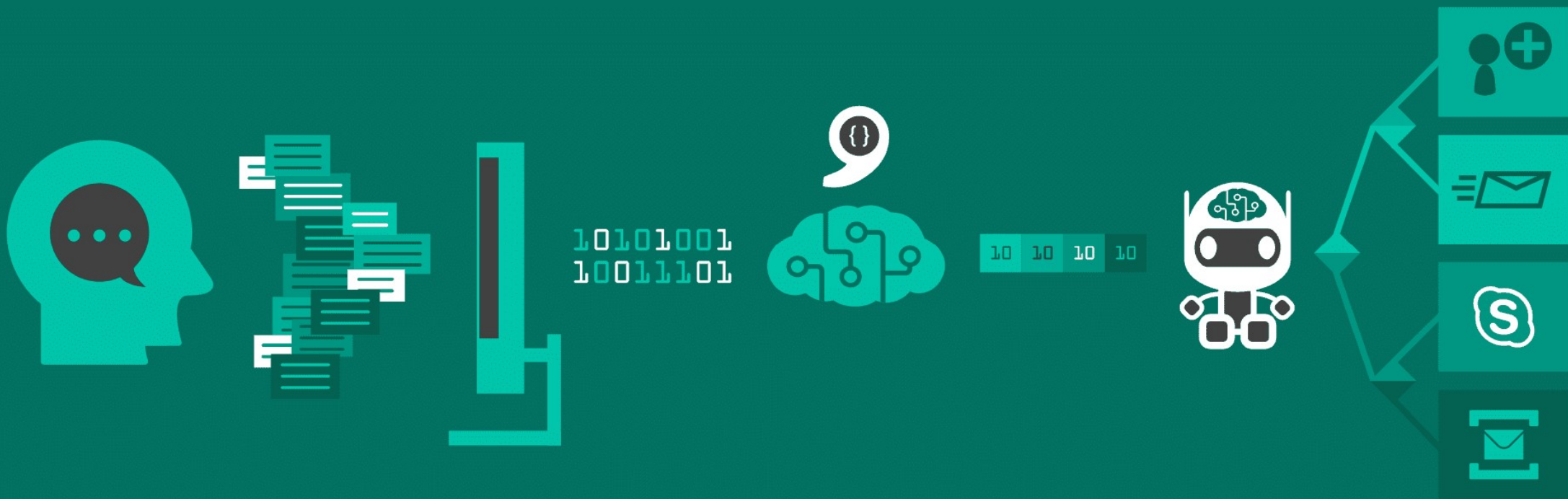
[행사개요](#) [아이디어](#) [시상금](#) [분야](#) [후원사](#)

2018 인공지능 챗봇 해커톤 AI CHATBOTHON 2018

개포디지털혁신파크, 2018.12.01(토)

주최: 한국인공지능연구소

[참가신청](#)





챗봇 기초 코딩 교육

이재석

주관 한국인공지능아카데미

후원 한국인공지능연구소

일정

오늘 일정


01 구름ide가입,설정	09:00 – 09:50(50분)
02 카카오톡친구플러스 설정	10:00 – 10:40(40분)
03 에코봇	11:00 – 11:20(20분)
04 번역봇	11:20 – 11:40(20분)
05 버튼봇	12:00 – 12:30(30분)
06 api봇	12:30 – 01:00(30분)

03 헬로우봇

3.1 헬로우봇

- 여기서 부터는 카카오 톡입니다.
- 방금 전에 설정한 카카오 플러스 친구가 아닙니다.

≡ | - x



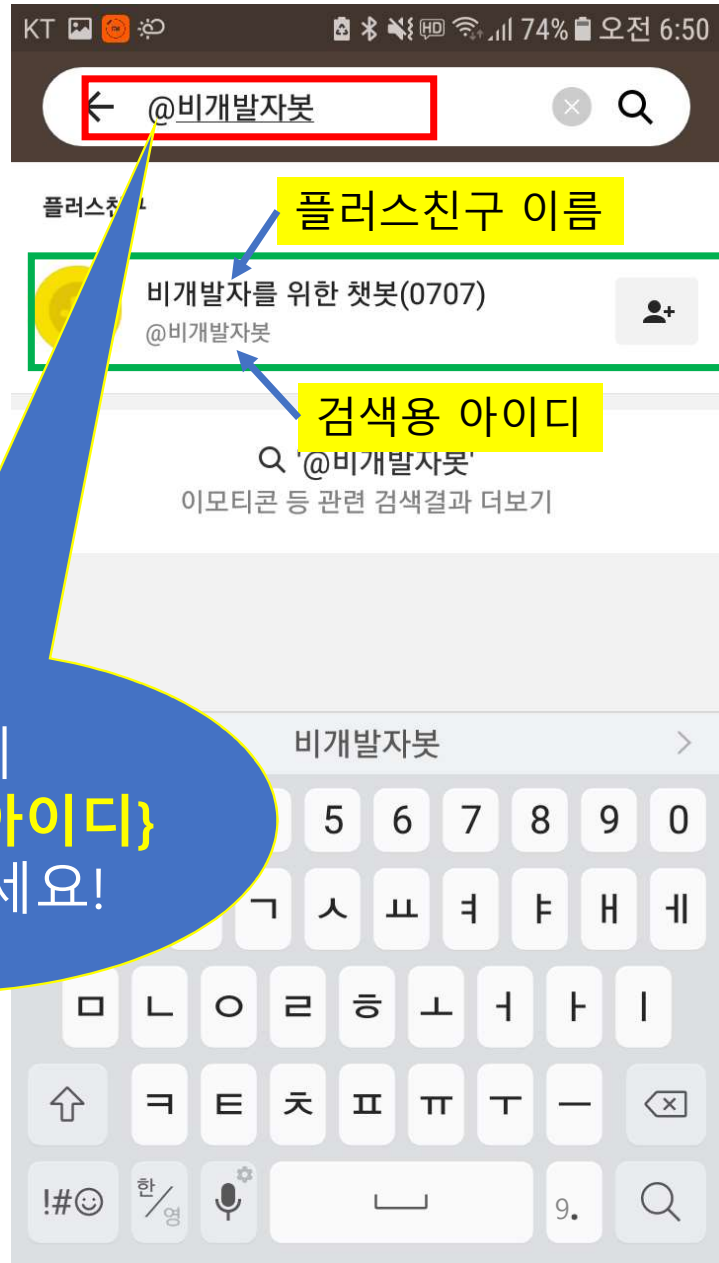
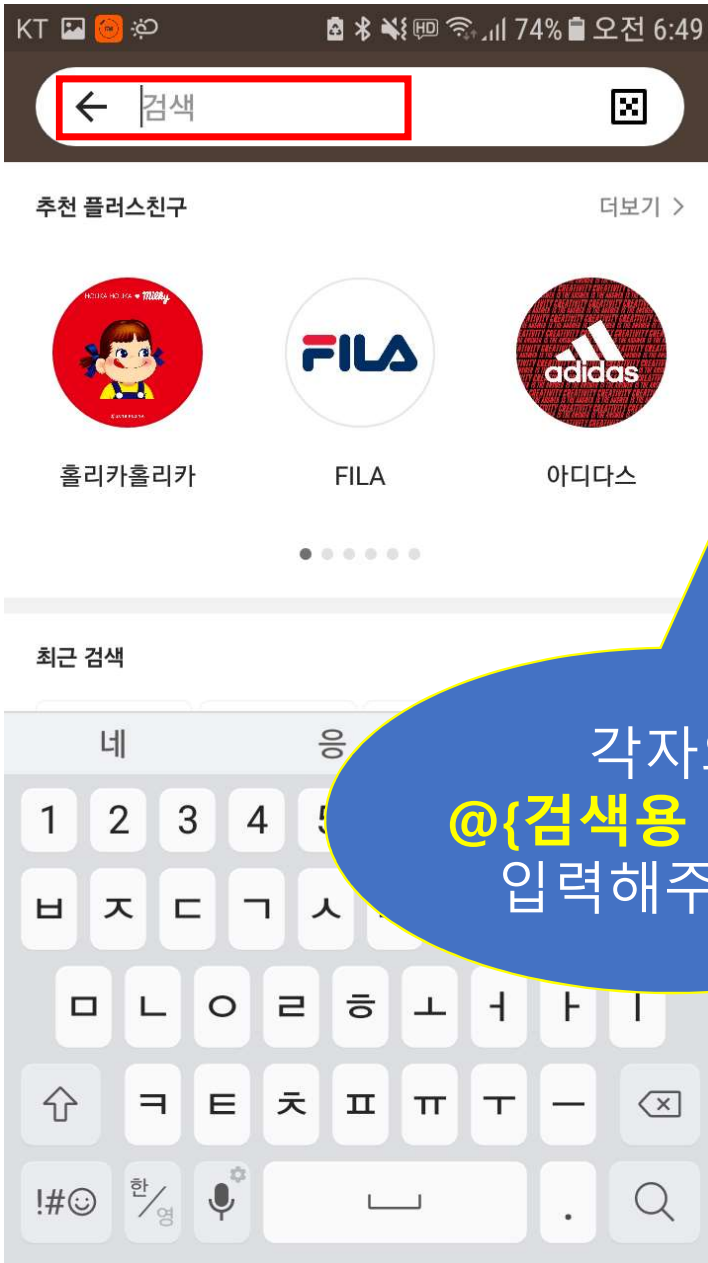
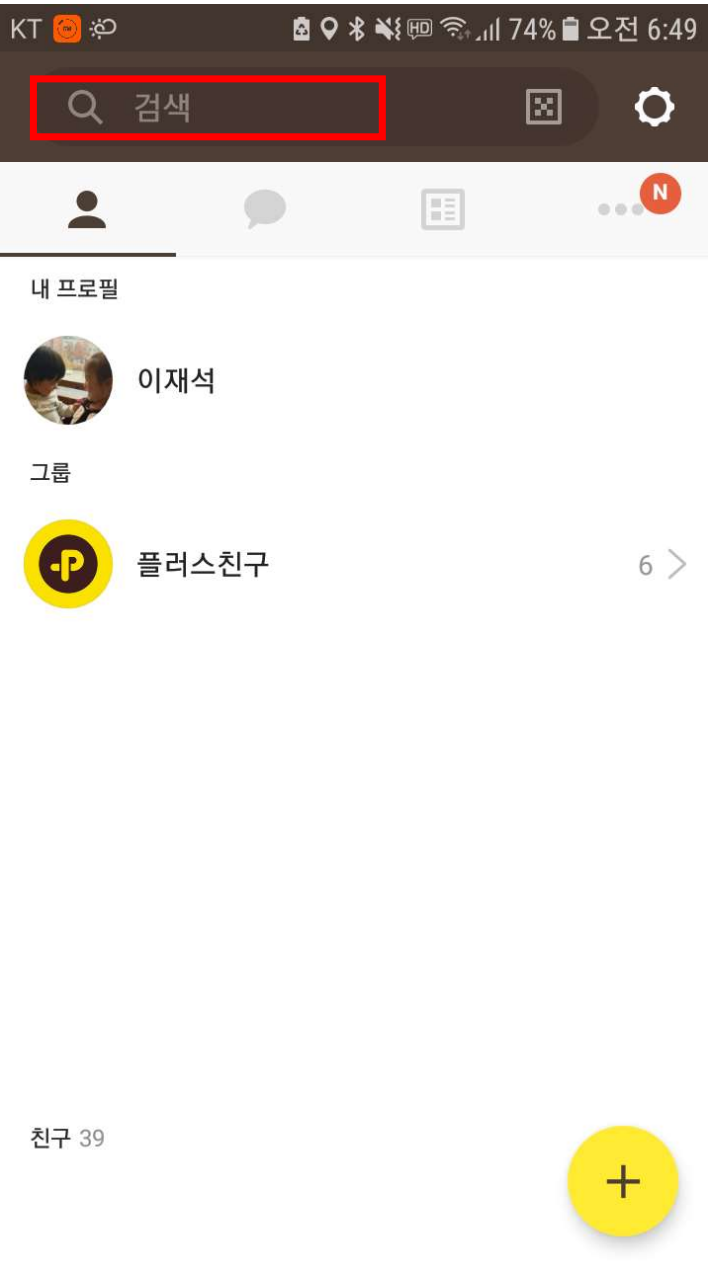
카카오계정 (이메일 또는 전화번호) ▼

비밀번호 (4~32자)

로그인

☒ 잠금모드로 자동로그인

[카카오계정 찾기](#) | [비밀번호 재설정](#)



@비개발자봇
은 제가 만든 챗봇입니다.

여러분은 여러분이 만든
@{각자 검색용 아이디}
챗봇으로 들어가주세요 ^^!!!



비개발자를 위한 챗
봇(0707)

총 캐시 0원 >
메시지 이용권 0개 >
무료 발송 메시지 1,000건 ?

다른 플러스친구 선택하기 ▾

홈

메시지 +

쿠폰 +

1:1채팅 +

스마트채팅

친구그룹 관리

홍보하기

통계 +

관리 -

상세 설정

플러스친구 정보

• 플러스친구 이름 비개발자를 위한 챗봇(0707) 변경하기

• 검색용 아이디 비개발자봇

• 홈 URL http://pf.kakao.com/_sgULC

• 카테고리 1 교육 ▾ 교육 일반 ▾

카테고리 2 카테고리 선택 ▾ 카테고리 선택 ▾

소개 메시지
플러스친구 홈에 노출될 소개 문구를 작성해주세요 (60자 이내)

부가정보

위치정보 ☐ 국내 ☐ 해외 ☒ 없음

홈페이지 .com

전화번호 예) 010-1234-5678 또는 +82-10-1234-5678

이메일

@검색용
아이디가 생각
나지 않는다면...

카카오 플러스
친구에서

검색용 아이디
를 확인하세요

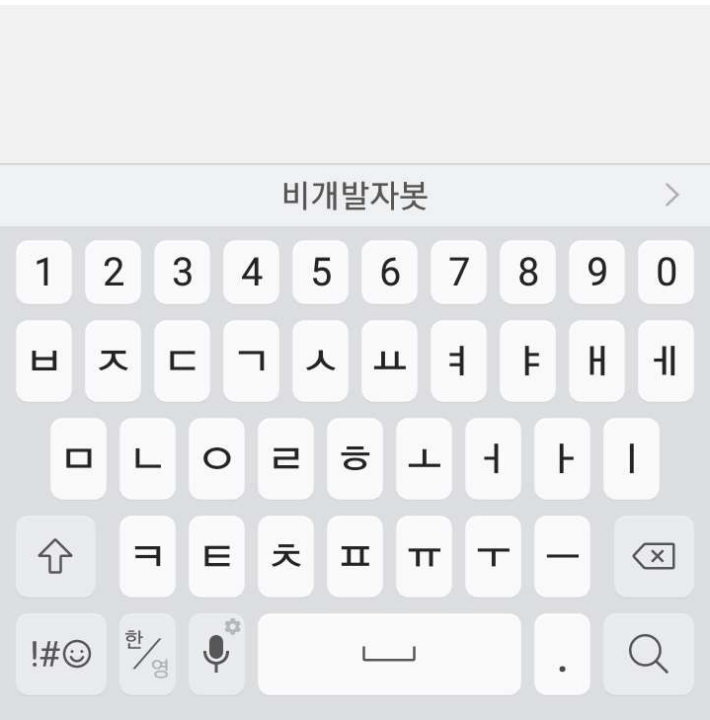
플러스친구



비개발자를 위한 챗봇(0707)
@비개발자봇



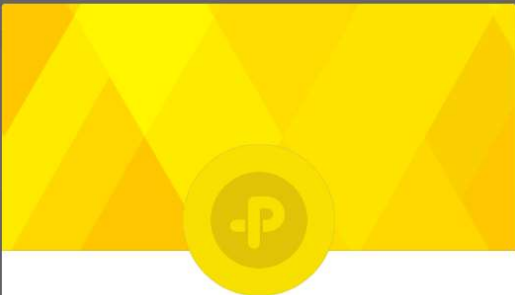
Q '@비개발자봇'
이모티콘 등 관련 검색결과 더보기



플러스친구



비개발자를 위한 챗봇(0707)
@비개발자봇



비개발자를 위한 챗봇(0707)

광고와 마케팅 메시지를
카카오톡으로 받아 볼 수 있습니다.

취소

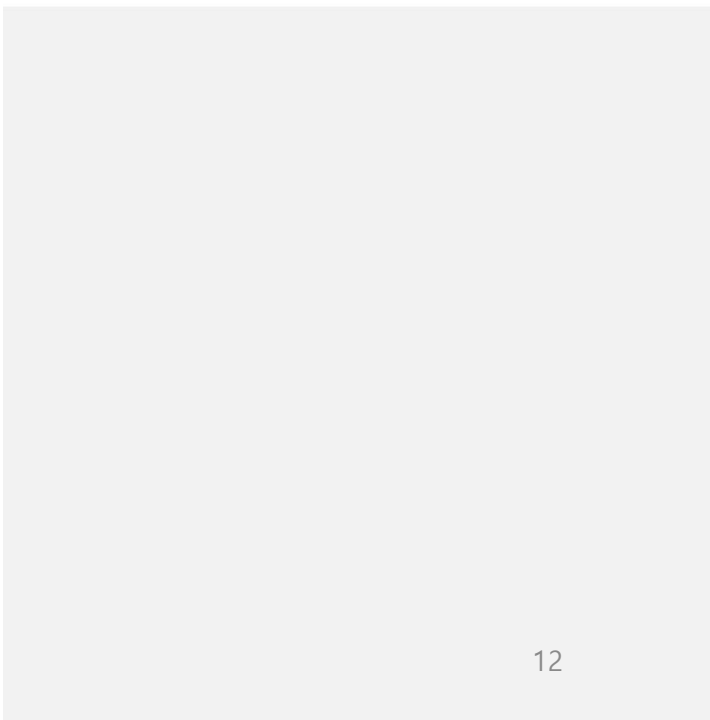
확인

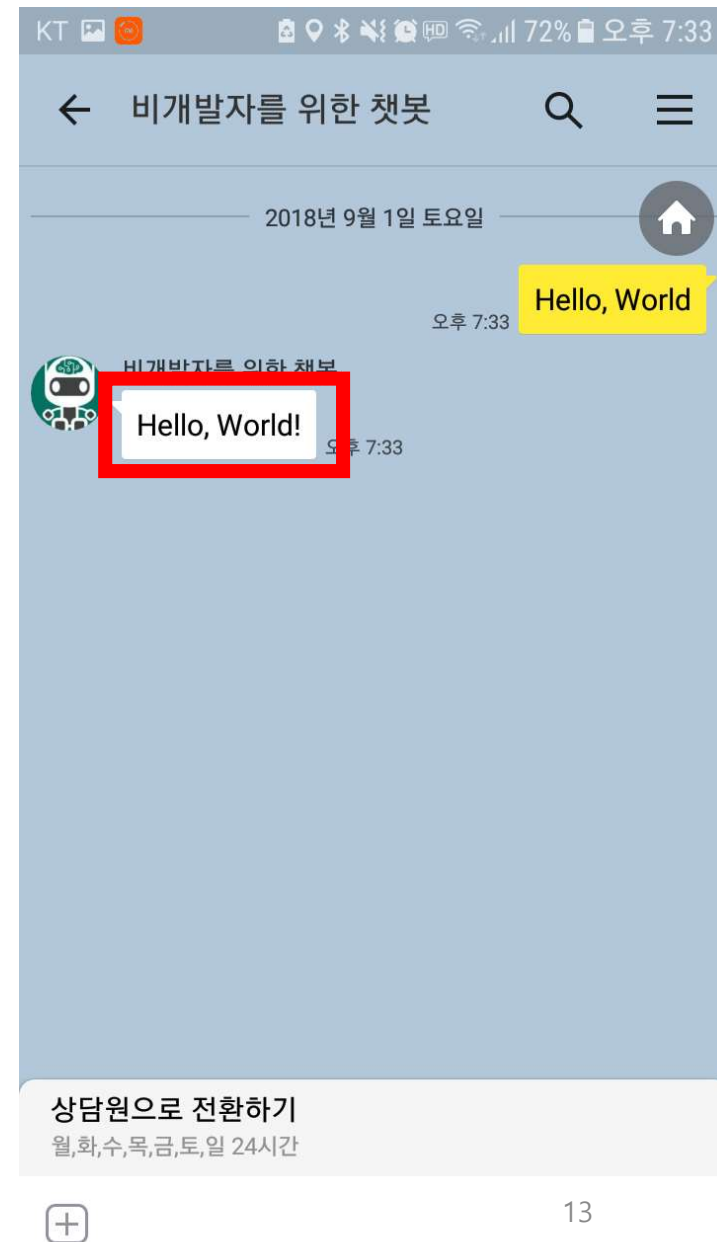
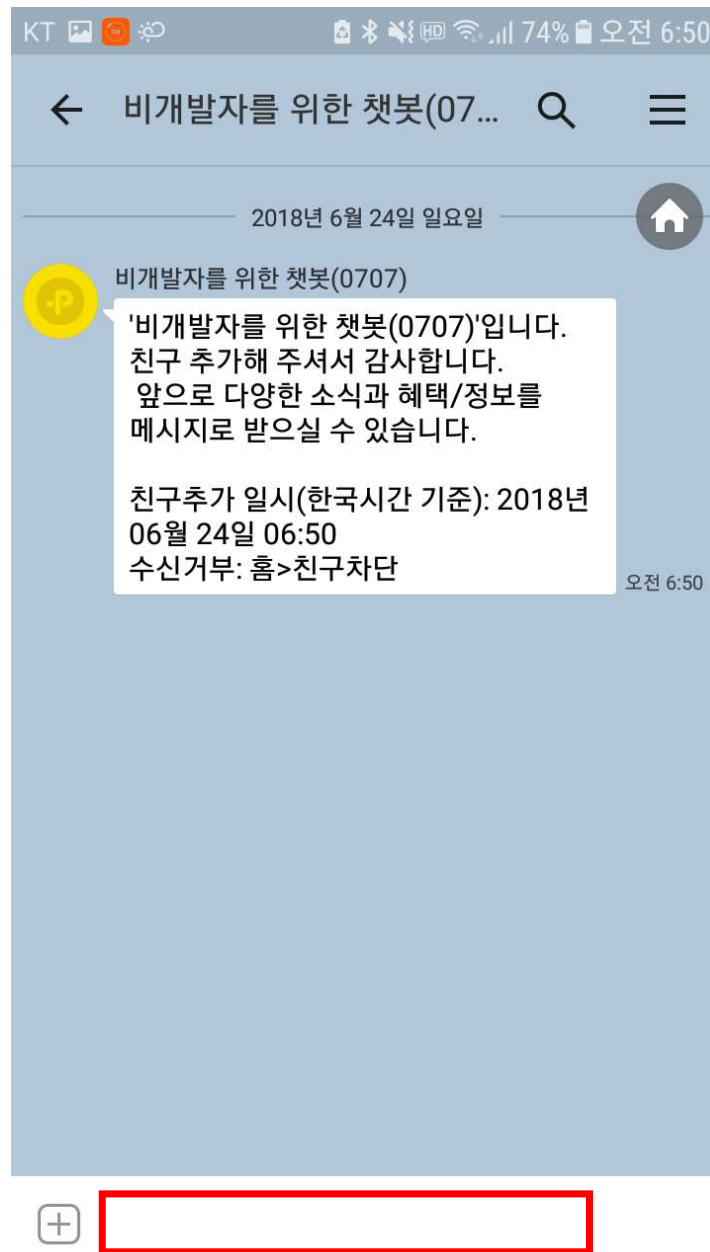
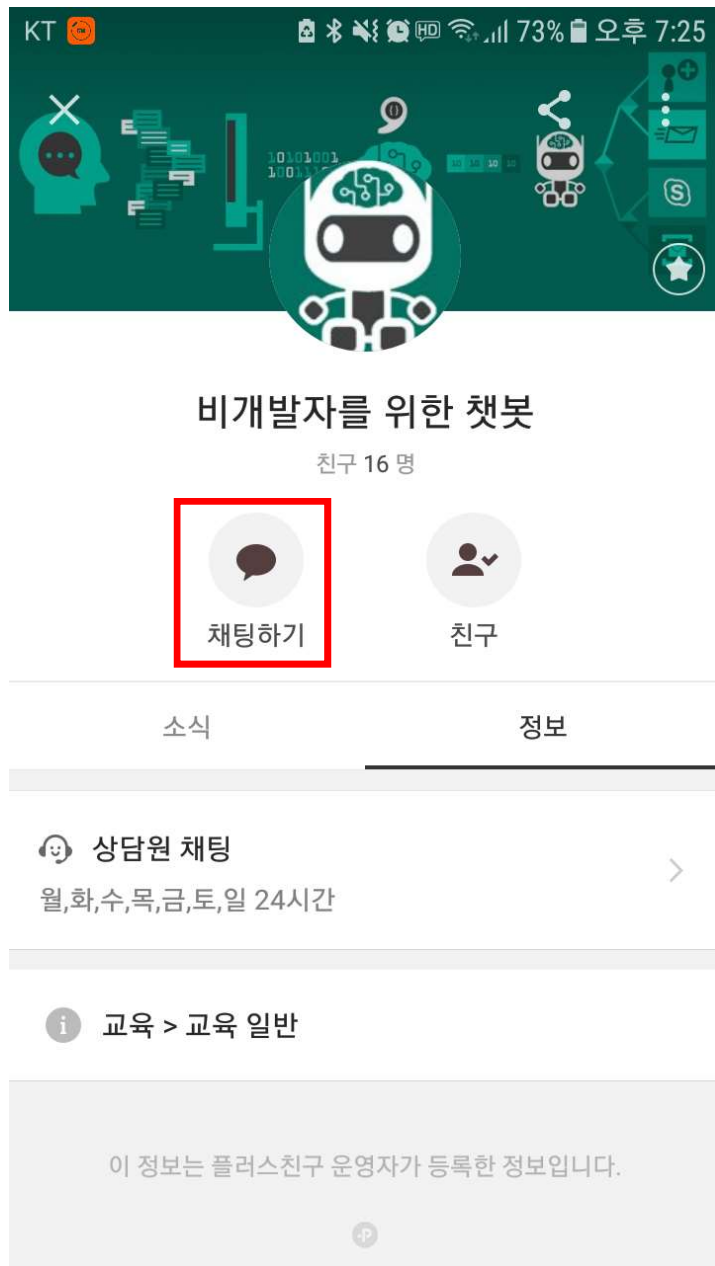
플러스친구

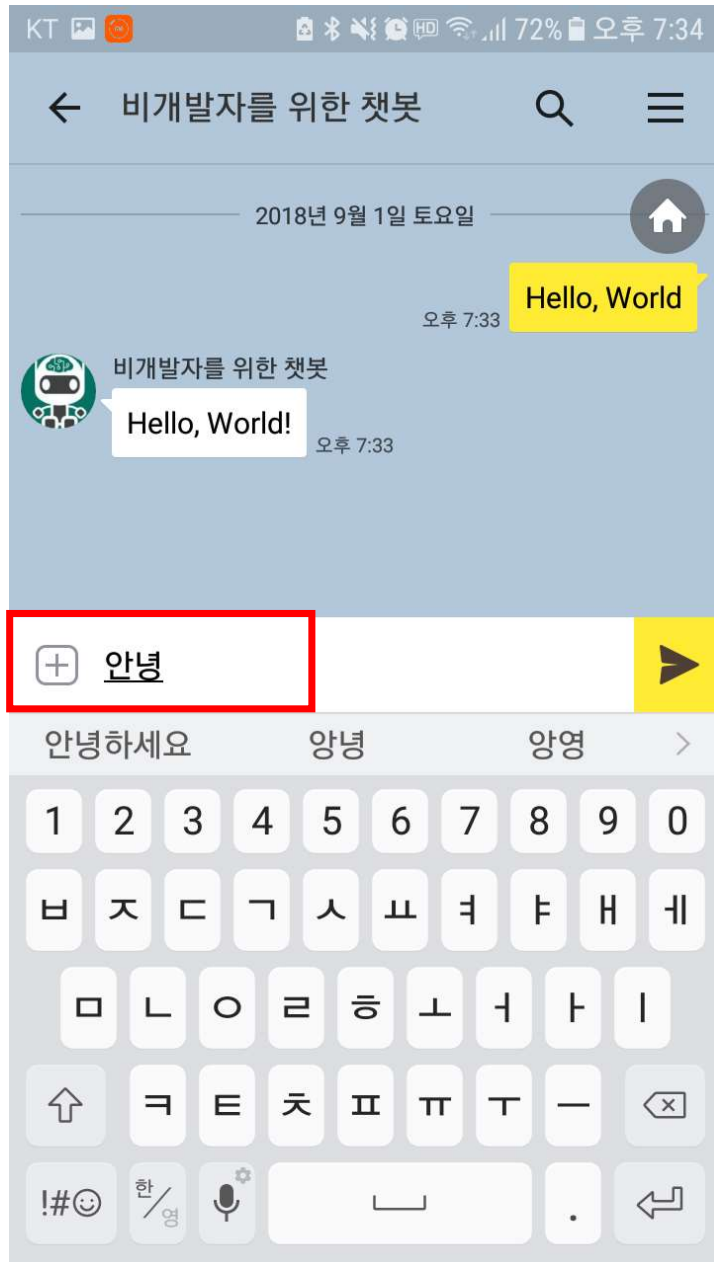


비개발자를 위한 챗봇(0707)
@비개발자봇

Q '@비개발자봇'
이모티콘 등 관련 검색결과 더보기







3.2 헬로우봇 소스코드분석

- 상세한 소스코드 분석은 3-2.2 에코봇에서 설명
- 지금까지 터미널, 웹 브라우저에서 출력되던 "Hello, World" 를 카카오톡에서 출력하려면...

```
from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])
def message():
    response = {
        "message": {
            "text": "Hello, World!"
        }
    }

    response = json.dumps(response, ensure_ascii=False)
    return response

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```



```
from flask import Flask  
from flask import request  
from flask import jsonify  
from flask import json
```

```
app = Flask(__name__)
```

```
@app.route("/keyboard")  
def keyboard():  
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])  
def message():  
    response = {  
        "message": {  
            "text": "Hello, World!"  
        }  
    }  
    response = json.dumps(response, ensure_ascii=False)  
    return response
```

```
if __name__ == "__main__":  
    app.run(host="0.0.0.0", port=5000)
```

https://github.com/TTEarth/chatbothon/blob/master/00.hello-world/kakao_bot.py

```
from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")
```

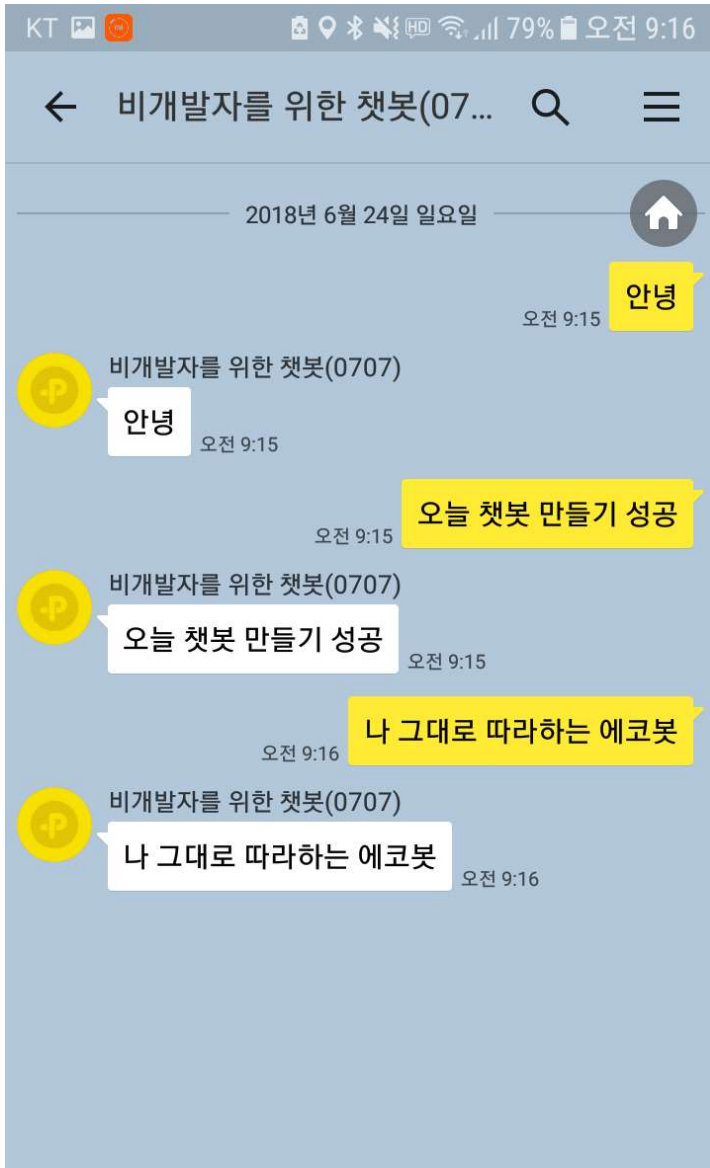
```
@app.route("/message", methods=["POST"])
def message():
    response = {
        "message": {
            "text": "Hello, World!"
        }
    }

    response = json.dumps(response, ensure_ascii=False)
    return response

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

03-2 에코붓

3-2.1 카카오톡에서 에코봇 실행



현재 우리는 헬로우봇을 서비스하고 있습니다.

옆에 보이는 **에코봇**을 서비스하려면 어떻게 해야 할까요?

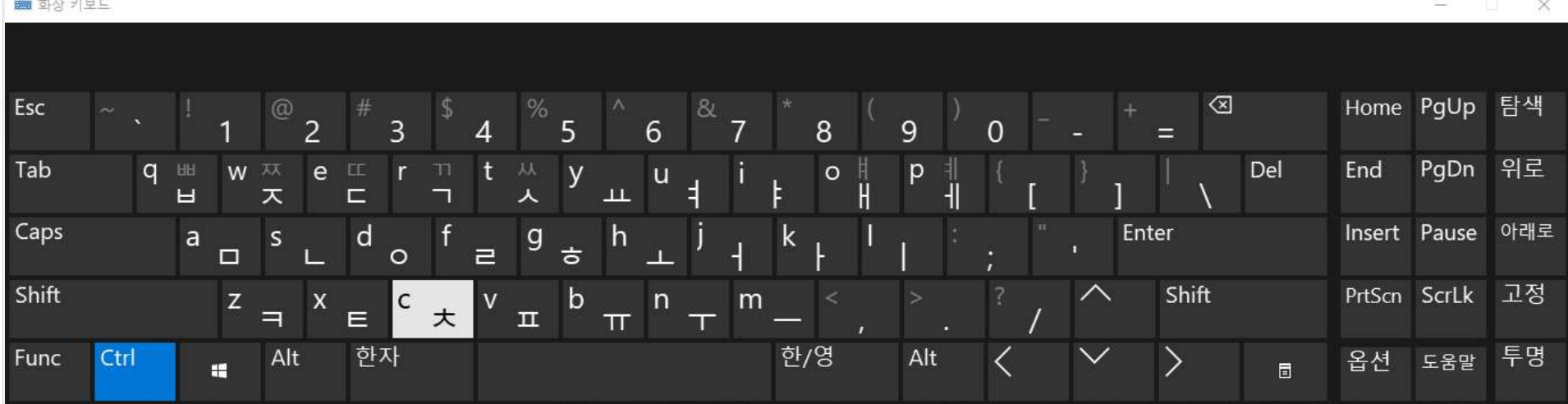
1. 기존 실행하던 헬로우봇 파이썬 프로그램들을 터미널에서 모두 삭제해주세요.
2. 에코봇 파이썬 파일을 실행하세요.
3. 카카오톡에서 테스트합니다.

**기억이 안나시면,
앞 슬라이드 처음부터 차례 차례 학습해주세요.**

📁 프로젝트
📁 파일
📁 디버그
📁 컨테이너
📁 소스 코드 관리
📁 배포
📁 창
📁 도움말

chatbothon

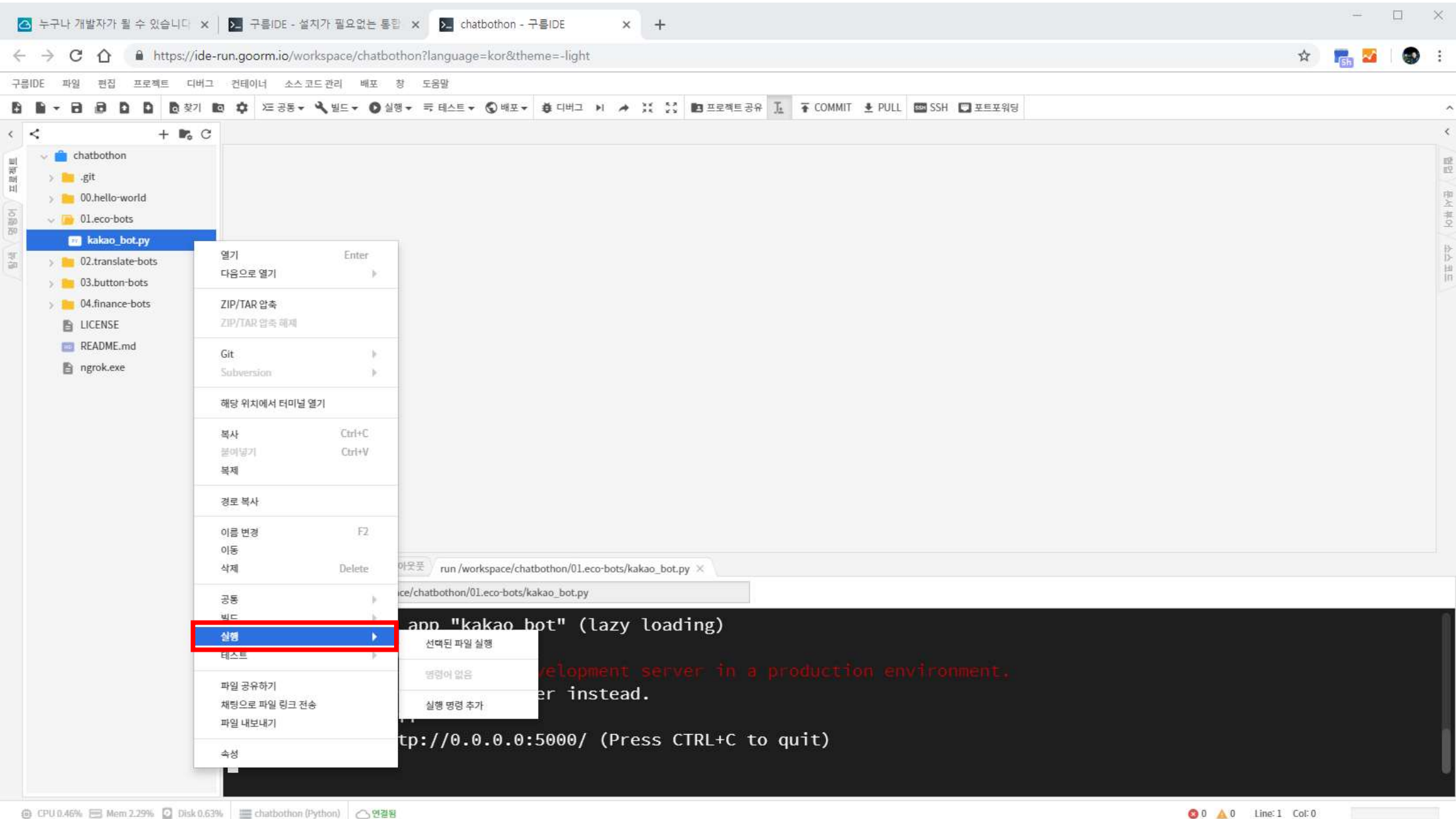
- 📁 .git
- 📁 00.hello-world
 - 📄 kakao_bot.py
 - 📁 01.eco-bots
 - 📁 02.translate-bots
 - 📁 03.button-bots
 - 📁 04.finance-bots
 - 📄 LICENSE
 - 📄 README.md
 - 📄 ngrok.exe



```
18 @app.route("/message", methods=["POST"])
19 def message():
20     response = {
21         "message": {
22             "text": "Hello, World!"
```

초기화 종료 python3 /workspace/chatbothon/00.hello-world/kakao_bot.py

```
* Serving Flask app "kakao_bot" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
^C root@goorm:/workspace/chatbothon(master) #
```



chatbothon

> .git

> 00.hello-world

> 01.eco-bots

kakao_bot.py

> 02.translate-bots

> 03.button-bots

> 04.finance-bots

LICENSE

README.md

ngrok.exe

열기 Enter

다음으로 열기 ▶

ZIP/TAR 압축

ZIP/TAR 압축 해제

Git ▶

Subversion ▶

해당 위치에서 터미널 열기

복사 Ctrl+C

붙여넣기 Ctrl+V

복제

경로 복사

이름 변경 F2

이동

삭제 Delete

공통 ▶

빌드 ▶

실행 ▶

테스트 ▶

파일 공유하기

채팅으로 파일 링크 전송

파일 내보내기

속성

run /workspace/chatbothon/01.eco-bots/kakao_bot.py x

ce/chatbothon/01.eco-bots/kakao_bot.py

app "kakao_bot" (lazy loading)

development server in a production environment.

er instead.

tp://0.0.0.0:5000/ (Press CTRL+C to quit)

누구나 개발자가 될 수 있습니다 x > 구름IDE - 설치가 필요없는 통합 x chatbothon - 구름IDE x +

https://ide-run.goorm.io/workspace/chatbothon?language=kor&theme=-light

구름IDE 파일 편집 프로젝트 디버그 컨테이너 소스 코드 관리 배포 창 도움말

찾기 > 공유 > 빌드 > 실행 > 테스트 > 배포 > 디버그 > 프로젝트 공유 > COMMIT > PULL > SSH > 포트포워딩

chatbothon

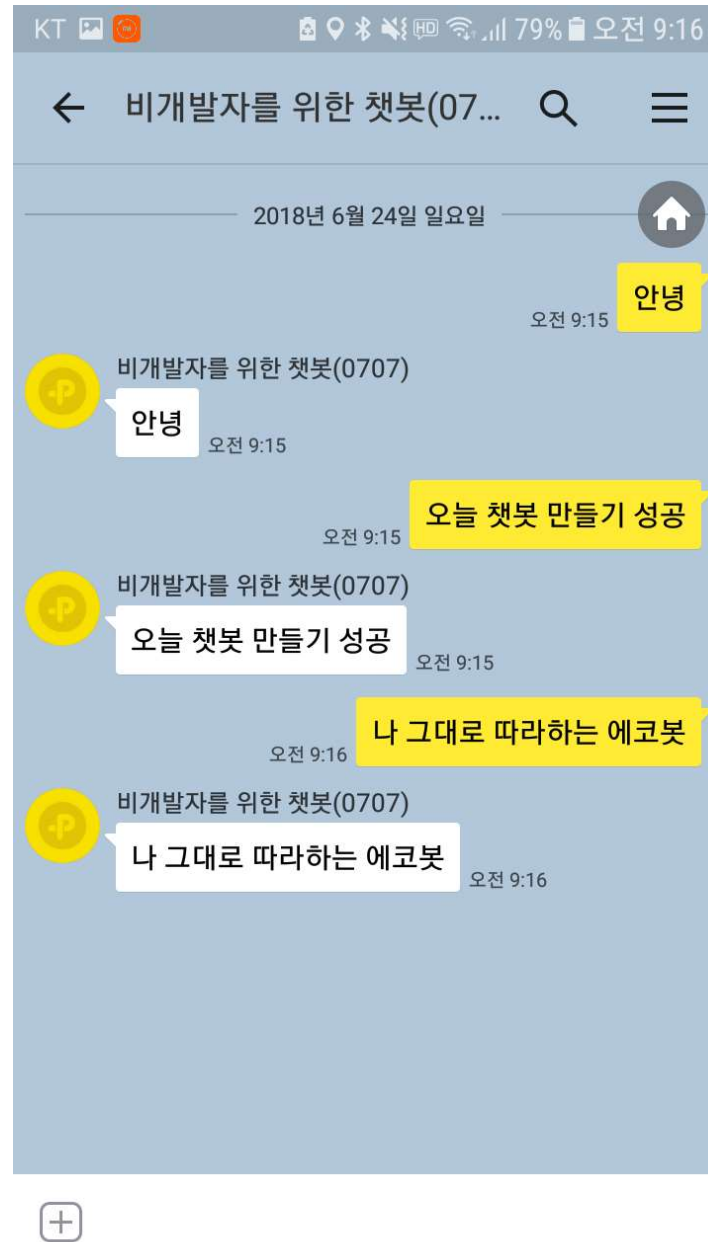
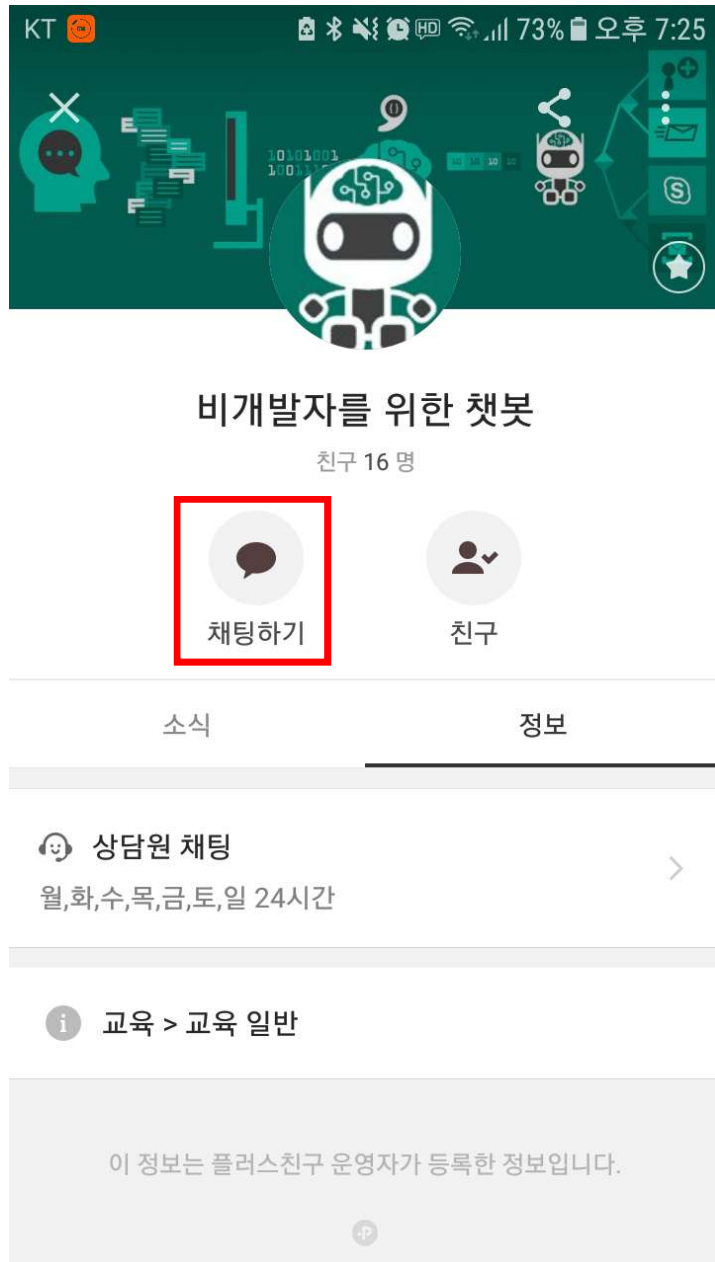
- > .git
- > 00.hello-world
- > 01.eco-bots
 - kakao_bot.py
- > 02.translate-bots
- > 03.button-bots
- > 04.finance-bots
- LICENSE
- README.md
- ngrok.exe

디버그 터미널 검색 아웃풋 run /workspace/chatbothon/01.eco-bots/kakao_bot.py x

초기화 종료 python3 /workspace/chatbothon/01.eco-bots/kakao_bot.py

```
root@goorm:/workspace/chatbothon(master)# python3 /workspace/chatbothon/01.eco-bots/kakao_bot.py
* Serving Flask app "kakao_bot" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

CPU 0.49% Mem 2.29% Disk 0.63% chatbothon (Python) 연결됨 0 0 Line: 1 Col: 0



3-2.2 에코봇 코드 분석

```
from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])
def message():
    data = json.loads(request.data)
    content = data["content"]

    response = {
        "message": {
            "text": content
        }
    }

    response = json.dumps(response, ensure_ascii=False)
    return response

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

```
from flask import Flask  
from flask import request  
from flask import jsonify  
from flask import json
```

```
app = Flask(__name__)
```

```
@app.route("/keyboard")  
def keyboard():  
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])  
def message():  
    data = json.loads(request.data)  
    content = data["content"]
```

```
    response = {  
        "message": {  
            "text": content  
        }  
    }
```

```
    response = json.dumps(response, ensure_ascii=False)  
    return response
```

```
if __name__ == "__main__":  
    app.run(host="0.0.0.0", port=5000)
```

https://github.com/TTEarth/chatbothon/blob/master/01.eco-bots/kakao_bot.py

```
from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])
def message():
    data = json.loads(request.data)
    content = data["content"]

    response = {
        "message": {
            "text": content
        }
    }

    response = json.dumps(response, ensure_ascii=False)
    return response

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

```
from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])
def message():
    data = json.loads(request.data)
    content = data["content"]

    response = {
        "message": {
            "text": content
        }
    }

    response = json.dumps(response, ensure_ascii=False)
    return response

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

```
from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])
def message():
    data = json.loads(request.data)
    content = data["content"]

    response = {
        "message": {
            "text": content
        }
    }

    response = json.dumps(response, ensure_ascii=False)
    return response

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```


3-2.3 카카오 플러스 친구 API

https://github.com/plusfriend/auto_reply



Search or jump to...



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)



[plusfriend](#) / [auto_reply](#)

Watch ▾

24

Star

260

Fork

98

Code

Issues 160

Pull requests 4

Projects 0

Wiki

Insights

플러스친구 자동응답 API

25 commits

1 branch

0 releases

6 contributors

Branch: master ▾

[New pull request](#)

[Create new file](#)

[Upload files](#)

[Find file](#)

[Clone or download ▾](#)

plusfriendteam Update README.md

Latest commit 0e0a18c on 19 Apr

README.md

Update README.md

3 months ago

README.md

카카오톡 플러스친구 API v. 2.0 개요

이 문서는 플러스친구를 통하여 자동응답 기능을 이용하고자 할 때 사용되는 API에 대해 기술합니다.

1. 이용에 대한 참고사항

1. 서버간 통신은 보안을 위하여 HTTPS를 쓰도록 권장합니다.
2. HTTPS를 통하여 API를 이용하는 파트너사 서버는 유효한 공인인증서를 사용해야 합니다.
3. user_key는 외부에 노출되지 않도록 주의해 주시기 바랍니다.
4. 플러스친구 API 운영에 대한 정책 및 가이드라인은 [플러스친구 이용약관](#)의 API 플랫폼 운영정책을 참고해 주시기 바랍니다.

2. 개인정보 수집 및 이용에 대한 주의사항

1. API 이용에 대한 주의사항은 [서비스 이용약관](#) '제 26 조 API 플랫폼 서비스'를 따르며, 이를 준수하지 않을 경우 API 서비스 제공을 중단할 수 있습니다.
2. 회원이 API 플랫폼 서비스를 이용하는 과정에서 이용자의 개인정보 수집이 필요한 경우, 이용자로부터 개인정보 수집 및

3. 이용 시작하기

3.1. 플러스친구 앱 생성

플러스친구에서 자동응답 기능을 사용하기 위해서는 먼저 운영툴을 통해 key 발급을 위한 앱을 등록해야 합니다.

1. 플러스친구 운영툴의 좌측 '스마트채팅' 메뉴에서 'API형'을 선택합니다.
2. 앱 정보와 모니터링 메시지를 수신할 전화번호를 입력하고, 개인정보 수집 및 이용에 동의한 뒤 정보를 저장하여 앱을 생성합니다.

3.2. 자동응답 서비스 시작

자동응답 서비스가 개발 완료되어 서비스 가능한 상태가 되면, 다음 단계에 따라 서비스를 시작할 수 있습니다.

1. API TEST : 등록된 앱 url 우측의 API TEST 버튼을 클릭하여 서비스 시작 가능여부를 체크합니다. 필수 API인 Home Keyboard API가 정상적인 응답을 받지 못하는 경우 서비스 시작이 불가합니다.
 2. 서비스 시작 : API 테스트를 통과한 경우, 서비스 시작 버튼을 클릭하면 자동응답 서비스가 시작됩니다.
- 기존에 (구) 플러스친구 API를 통해 사용중이던 봇이 있는 경우, 신규 API를 통해 자동응답 서비스를 시작하게 되면 (구)API를 이용한 봇은 자동으로 종료되며 다시 (구) 봇의 형태로 이용할 수 없습니다.
 - (구) 플러스친구에서 제공하던 테스트 플친은 더이상 제공되지 않습니다. 실서비스 적용 전 테스트를 원하시는 경우 테스트 전용으로 이용하실 별도의 플러스친구 계정을 생성하여 테스트를 진행해 주시기 바랍니다.
 - 신규 API 자동응답은 키워드형 자동응답과 동시에 사용할 수 없습니다. 다만 API 자동응답의 서비스 중지 후 키워드형 자동응답을 사용하는 것은 가능합니다.

3.3. 자동응답 서비스 중지

- 직접 중지 : 더 이상 자동응답 서비스를 이용하지 않으시려면, 운영툴의 자동응답 > 자동응답 API 메뉴에서 '서비스 중지'를 통해 자동응답 기능을 중지할 수 있습니다.
- 오류 횟수 초과에 따른 중지 : 자동응답 서비스에서 오류가 발생하는 경우, 카카오는 앱 정보에 등록된 전화번호를 통해 모니터링 메시지를 발송합니다. 만일 오류 횟수가 일 1000건을 초과하게 되면 카카오는 자동으로 해당 앱의 서비스를 중지합니다. 오류 횟수 초과로 인해 중지된 앱은 운영툴에서 오류 수정 후 직접 다시 서비스를 시작할 수 있습니다.
- 관리자의 차단에 따른 중지 : 파트너사에서 플러스친구 API의 운영정책에 반하는 내용을 API를 통해 서비스하는 것이 확인된 경우, 카카오의 관리자는 해당 앱의 서비스를 중지할 수 있습니다. 관리자에 의해 차단된 앱은 직접 서비스를 다시 시작할 수 없으므로, 앱의 차단 사유를 확인하신 뒤 문제가 되는 내용을 수정하여 플러스친구 고객센터(1544-4293)를 통해 차단 해제를 요청해 주시기 바랍니다.

3.4. 클라이언트 테스트

자동응답 기능이 서비스가 시작되면 카카오톡 채팅방에서 직접 자동응답 기능을 실행할 수 있습니다. 신규 자동응답 API를 통

5.1. Home Keyboard API

- 이용자가 최초로 채팅방에 들어올 때 기본으로 키보드 영역에 표시될 자동응답 명령어의 목록을 호출하는 API입니다.
- 채팅방을 지우고 다시 재 진입시에도 호출됩니다. 다만 카카오톡 서버에서도 1분동안 캐쉬가 저장되기 때문에 유저가 채팅방을 지우고 들어오는 행동을 반복하더라도 개발사 서버를 1분에 한번씩 호출하게 됩니다. 즉, 개발사 서버에서 정보가 변경되어도 최대 1분뒤에 유저들에게 반영이 됩니다.
- 유저가 자동응답으로 메시지를 주고 받았을 경우는 마지막 메시지에 담겨있던 자동응답 명령어 목록이 표시됩니다. 다만 메시지에 저장된 자동응답 명령어는 10분간 유효합니다. 10분이 지난 다음에는 다시 keyboard api를 호출하여 자동응답 목록을 초기화하게 됩니다.

Specification

- Method : GET
- URL : `http(s)://:your_server_url/keyboard`
- Content-Type : `application/json; charset=utf-8`
- 예제

```
curl -XGET 'https://:your_server_url/keyboard'
```

- Response

필드명	타입	필수여부	설명
keyboard	Keyboard	Required	키보드 영역에 표현될 버튼에 대한 정보. 생략시 text 타입이 선택된다.

- 예제

```
{
  "type" : "buttons",
  "buttons" : ["선택 1", "선택 2", "선택 3"]
}
```

5.2. 메시지 수신 및 자동응답 API

- 사용자가 선택한 명령어를 파트너사 서버로 전달하는 API입니다.
- 자동응답 명령어에 대한 답변은 응답 메시지(Message)와 응답 메시지에 따른 키보드 영역의 답변 방식(Keyboard)의 조합으로 이루어집니다. 답변 방식은 주관식(text)과 객관식(buttons) 중 선택할 수 있습니다.
- 자동응답을 통해 친구에게 미디어 타입(사진/동영상/오디오)을 받고자 하는 경우 주관식 키보드(text)를 선택하세요. 메시지를 통해 파트너사 서버로 보내 미디어를 전송하세요. 약간의 애플리케이션이 필요하느 이점입니다.

카카오 플러스 친구 API

1) 키보드 : GET /keyboard

- 이용자가 **최초로 채팅방에 들어올 때**, 기본으로 키보드 영역에 표시될 자동응답 명령어의 목록을 호출하는 API입니다.

2) 메시지 : POST /message

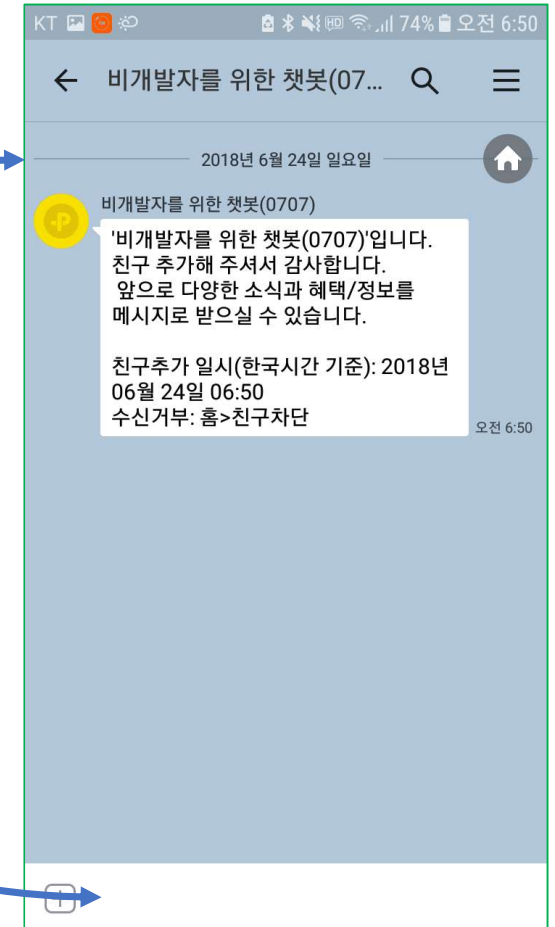
- 자동응답 API에서 제일 핵심인 내용입니다. 사용자가 **보내는 메시지/사진/동영상**을 이 API를 통해서 보내고 서버가 처리해서 그에 맞는 응답을 해주면 됩니다.

최초로 채팅방에 들어올 때

```
from flask import jsonify
```

```
@app.route("/keyboard")  
def keyboard():  
    return jsonify(type="text")
```

Flask에서는 jsonify라는 유틸리티 함수가 있어서
사전이나 배열을 던져서 쉽게 JSON 타입의 HTTP Response를 생성해준다



```
from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])
def message():
    data = json.loads(request.data)
    content = data["content"]

    response = {
        "message" : {
            "text" : content
        }
    }

    response = json.dumps(response, ensure_ascii=False)
    return response

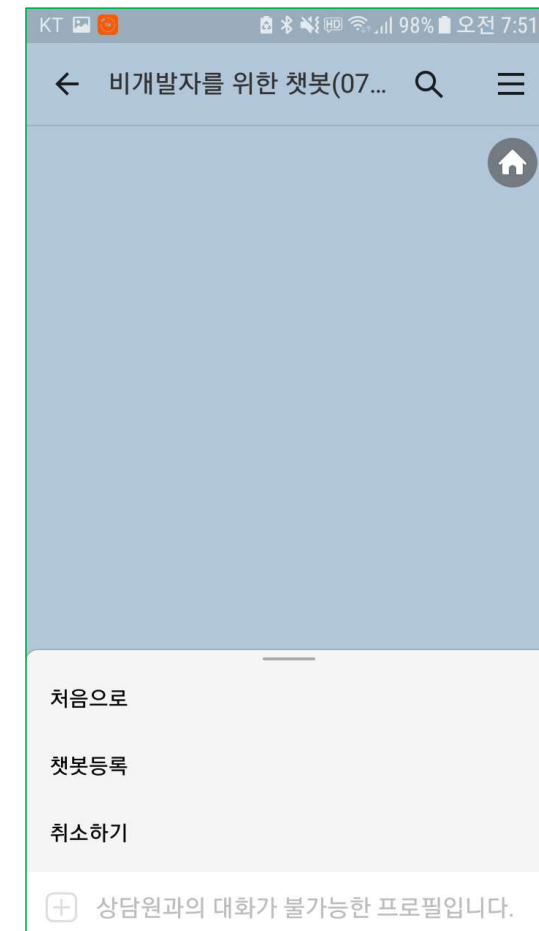
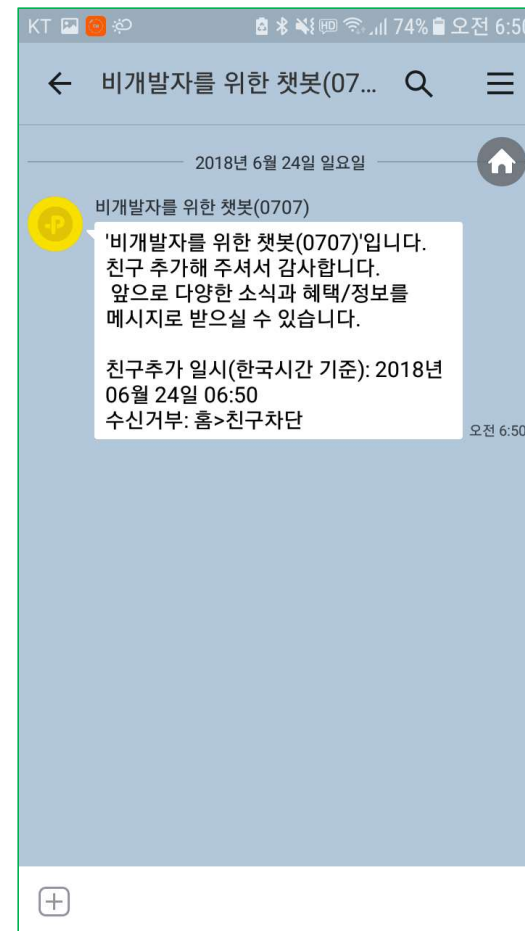
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

https://github.com/TTEarth/chatbothon/blob/master/01.eco-bots/kakao_bot.py

keyboard 타입

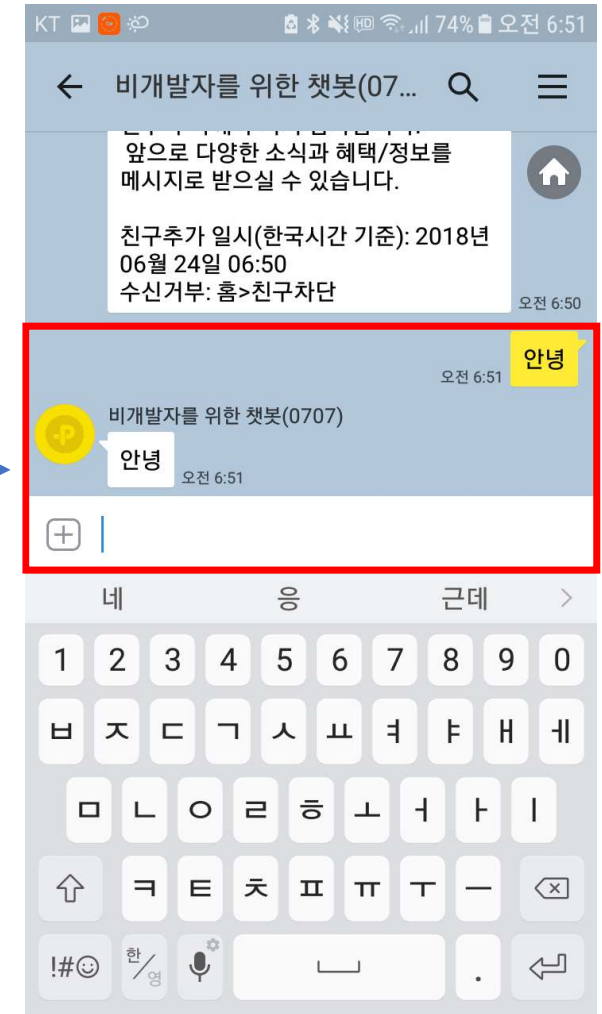
keyboard 타입은 두 가지 뿐

- type="text"
- type="buttons"



자동응답 메시지 구현

```
@app.route("/message", methods=["POST"])  
def message():  
    return response
```



```
from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])
def message():
    data = json.loads(request.data)
    content = data["content"]

    response = {
        "message" : {
            "text" : content
        }
    }

    response = json.dumps(response,
ensure_ascii=False)
    return response

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

json 이란

- JSON (JavaScript Object Notation)은 경량의 DATA - 교환 형식이다.
- 특히 웹 브라우저와 웹서버 사이에 데이터를 교환하는데 많이 사용하고 있다.
- name / value 형태의 쌍으로 collection 타입이다.
- `from flask import json`

```
{ "name" : "value" }
```

```
from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])
def message():
    data = json.loads(request.data)
    content = data["content"]

    response = {
        "message" : {
            "text" : content
        }
    }

    response = json.dumps(response, ensure_ascii=False)
    return response

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

python 딕셔너리

```
response = {  
    "message": {  
        "text": content  
    }  
}
```

json 관련 메소드

```
from flask import json
```

```
json.loads()
```

```
json.dumps()
```

```
from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")
```

```
@app.route("/message", methods=["POST"])
def message():
    data = json.loads(request.data)
    content = data["content"]

    response = {
        "message": {
            "text": content
        }
    }

    response = json.dumps(response, ensure_ascii=False)
    return response

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```


JSON 디코딩 - json.loads()

- JSON 문자열을 Python 타입 (Dictionary, List, Tuple 등) 으로 변경하는 것을 말한다

```
from flask import request  
from flask import json
```

```
@app.route("/message", methods=["POST"])
```

```
def message():
```

```
    data = json.loads(request.data)
```

```
    content = data["content"]
```

Python Flask에서 원시
POST 본문을 가져 오는것

{"content": "안녕"}



메시지 수신 및 자동응답 API

- **Method** : POST
- **URL** : http(s)://:your_server_url/message
- **Content-Type** : application/json; charset=utf-8
- **Parameters**

필드명	타입	필수여부	설명
user_key	String	Required	메시지를 발송한 유저 식별 키
type	String	Required	text, photo
content	String	Required	자동응답 명령어의 메시지 텍스트 혹은 미디어 파일 uri

- https://github.com/plusfriend/auto_reply#52-메시지-수신-및-자동응답-api

메시지 수신 및 자동응답 API

- **Method** : POST
- **URL** : http(s)://:your_server_url/message
- **Content-Type** : application/json; charset=utf-8
- **Parameters**

필드명	타입	필수여부	설명
user_key	String	Required	메시지를 발송한 유저 식별 키
type	String	Required	text, photo
content	String	Required	자동응답 명령어의 메시지 텍스트 혹은 미디어 파일 uri

- https://github.com/plusfriend/auto_reply#52-메시지-수신-및-자동응답-api

메시지 수신 및 자동응답 API

```
curl -XPOST 'https://:your_server_url/message' -d '{  
  "user_key" : "encryptedUserKey",  
  "type" : "text",  
  "content" : "차량번호등록"  
}'
```

- https://github.com/plusfriend/auto_reply#52-메시지-수신-및-자동응답-api

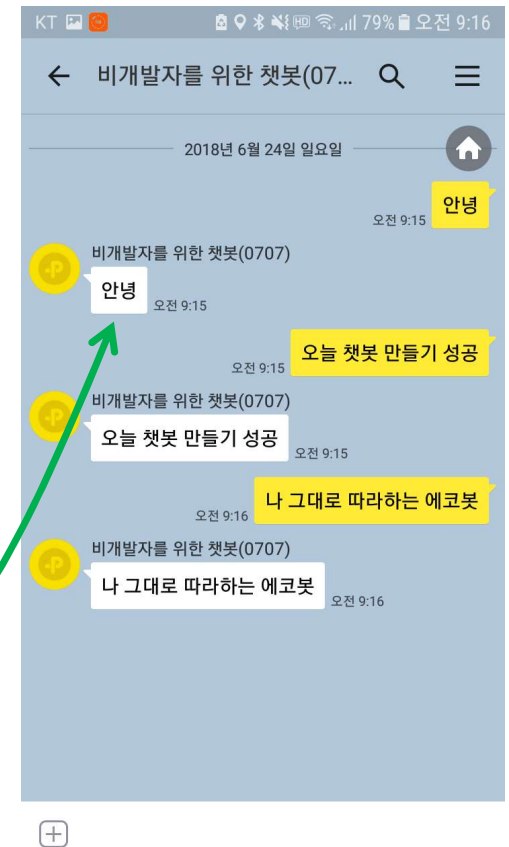
JSON 인코딩 - json.dumps()

- Python Object (Dictionary, List, Tuple 등) 를 JSON 문자열로 변경하는 것을 말한다.

```
from flask import json
```

```
response = {  
    "message" : {  
        "text" : content  
    }  
}
```

```
response = json.dumps(response, ensure_ascii=False)  
return response
```



메시지 자동응답 API - Response

필드명	타입	필수여부	설명
message	Message	Required	자동응답 명령어에 대한 응답 메시지의 내용.
keyboard	Keyboard	Optional	키보드 영역에 표현될 명령어 버튼에 대한 정보. 생략 시 text 타입(주관식 답변 키보드)이 선택된다.

```
{  
  "message" : {  
    "text" : "귀하의 차량이 성공적으로 등록되었습니다. 축하합니다!"  
  }  
}
```

- https://github.com/plusfriend/auto_reply#52-메시지-수신-및-자동응답-api

ensure_ascii=False

json, 즉 딕셔너리에 한글을 넣어서 출력하면, 유니코드로 나온다.

```
>>> dic = dict()
>>> dic['key'] = "한글"
>>> print dic
{'key': '\xed\x95\x9c\xea\xb8\x80'}
```

딕셔너리에서 value 값에 들어 있는 한글이 제대로 나오게 하려면...
JSON 파일 내에 "읽을 수 있는" 형태로 문자열을 그대로 저장하고 싶다면...

```
>>> import json
>>> print json.dumps(dic, ensure_ascii=False)
{"key": "한글"}
```

- <http://khanrc.tistory.com/entry/한글-in-the-dictionary-feat-pretty>

에코 봇이 되는 이유

```
@app.route("/message", methods=["POST"])  
def message():
```

```
    data = json.loads(request.data)  
    content = data["content"]
```

```
    response = {  
        "message": {  
            "text": content  
        }  
    }
```

```
    response = json.dumps(response, ensure_ascii=False)  
    return response
```



아래 코드는 같은 결과가 나올까요?

```
data = json.loads(request.data)
content = data["content"]
```

```
data = request.get_json()
content = data['content']
```

채팅창에 글자가 깨져서 나오는 경우

```

from flask import Flask
from flask import request
from flask import jsonify
from flask import json

app = Flask(__name__)

@app.route("/keyboard")
def keyboard():
    return jsonify(type="text")

```

```

@app.route("/message", methods=["POST"])
def message():
    data = json.loads(request.data.decode('utf8'))
    content = data["content"]

    response = {
        "message": {
            "text": content
        }
    }

    response = json.dumps(response, ensure_ascii=False)
    return response

```

윈도우 사용자의 경우,
글자가 깨질 경우,
.decode('utf8')
를 추가해주세요

```

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)

```



take a break

JAESEOK LEE
www.ttearth.com / www.facebook.com/EonNow