객체지향설계프로젝트

Team 11 최종 보고서

1. 팀 구성원 및 역할

 # 소프트웨어공학과 김동현

→ 이번 프로젝트의 주제인 광주 버스 정류장 이용객 데이터를 이용하여 시설 입점 위치를 추천해주는 프로그램에서 지도 시각화 및 결론 도출 부분 프로그래밍

→ 회의 내에서 Folium을 이용한 사람 수에 따라 지도에 시각화하는 방법이 나와 이러한 부분을 배우고 버스 정류장 이용객에 따라 지도에 시각화

→ 피드백을 통하여 근처에 있는 여러개의 정류소를 사용하기 위해 노력. 그 결과 버스정류장의 위도 경도를 활용하여 이용객이 많은 버스 정류장뿐만 아니라 주변 정류장까지 표시 함으로써 그 주변의 버스 이용객 수를 시각화하여 알려줄 수 있는 부분을 프로그래밍

→ 연령층별 기준으로 버스 정류장을 이용한 사람 중에서 해당 버스 정류장을 이용하는 사람의 비율을 구하여 이번 프로그램의 결론 부분에 시설 입점 위치 추천과 주변 정류장 및 비율의 정보 도출

 # IoT 인공지능학과 윤경윤

→ 11팀 프로젝트 주제 '광주 버스 데이터에서 유용한 정보 추출하기'를 제안함.

→ 프로젝트 세부 목표사항과 전반적인 사항을 기획함

- '버스정류장 데이터를 분석해서 지도 그래프 등으로 시각화' 의견 제안
- '유동인구를 통해 가게 입점 위치를 추천하는 프로그램을 제작' 의견 제안 등등

→ 프로젝트 전체 데이터 수집 및 정보 검색

- 11팀 프로젝트에 사용된 전체 데이터를 수집
- '정류소 별 첫차막차', '경유버스개수' 등의 활용되지 않았지만 필요할 거라 예상되는 데이터를 수집함.
- 지도 그래프 시각화 프로그래밍 관련 정보를 검색하여 팀원에게 제공

→ PPT 발표 자료 구성 및 제작

- 발표 순서와 목차를 구성하여 제작
- 광주광역시 교통수단 분담률 등의 자료를 찾아 발표자료에 활용
- SWOT 분석법으로 결과 및 결론 부분 작성

 # 소프트웨어공학과 김지민

→ 주제 선정을 위해 팀원들과 각자 데이터와 그에 맞는 활용 아이디어 고안
→ 전체 프로젝트 활동 동안 의견 참여
→ PPT 구성 중 이번 활동 과정에 대한 소감 혹은 평가 부분 필요하다고 의견 제시
→ 시각화 자료에 포함될 막대 그래프와 프로그램 실행 결과에서 연령대별 순위를 나타낸 막대 그래프 코드를 수업 내용과 인터넷 자료를 함께 참고하여 프로그래밍
→ PPT 발표 영상 녹화 참여 (미사용)

 # 소프트웨어공학과 최재원

→ 주제 선정 : 심부전과 관련된 환자의 증상에 따른 사망여부 데이터 수집 (기각)
→ 데이터 전처리 과정 설계 및 수행
- 데이터 전처리 A : 연령대별 버스 정류소 데이터의 전처리 (의견 수렴 후 처리)
- 데이터 전처리 B : 위도와 경도를 추출하는 프로그램 자체 제작 (dataP.py)
  프로그램을 통해 위도와 경도 데이터를 추출하고 정리하여 A 데이터와 병합

→ 프로그래밍
- 3차원 그래프 프로그래밍 : 3차원 그래프에 대한 전체 구성 설계 및 프로그래밍 (graph_3d.py) (미사용)
- 인공지능 프로그래밍 : 인공지능에 대한 전체 구성 설계 및 프로그래밍 (ai.py)
- 지도 프로그래밍 : 군집화 함수, 결론 출력 함수를 제외한 지도 시각화 함수 설계 및 프로그래밍 (map.py)
- 막대 그래프 프로그래밍 : 함수로 구성된 막대 그래프 프로그램을 클래스화 시키고 인공지능과 결합 (bar.py)

→ 발표물 제작 및 영상 녹화
- PPT : PPT 제작 (전반적인 PPT 내용 구체화 및 제작 ¦ 데이터 부분과 프로그램 구현 부분을 중점적으로 제작)
- 발표 : PPT 및 Code 실행 발표 영상 녹화

전반적인 팀 활동 평가

→ 개개인에게 따로 과제를 부여하고 이를 수행하였지만 매주 최소 1회의 팀회의 (ZOOM을 활용한 비대면 회의)와 지속적인 연락을 통해 의견을 수렴하며 전반적인 프로젝트 진행이 협력을 통해 진행
→ 전 팀원들의 팀 참여도 및 기여도 : 100%

2. 프로젝트 개요

(a) 주제

광주 버스 데이터에서 유용한 정보 추출하기
설계 인공지능 : 정류소 이용 현황을 바탕으로 유동인구를 파악하여 시설 입점 지역 추천

(b) 주제 선정 이유

광주광역시에서 2번째로 가장 많이 이용되는 교통 수단
광주광역시에서 가장 많이 이용되는 대중 교통
→ 이를 분석하면 유의미한 결론 도출이 가능할 것이라고 판단

3. 활용 데이터

(a) 데이터 A

데이터명	2016년도 광주광역시 시내버스 정류소별 이용객 현황								
내용 (예시)	정류소명	정류소 번호	총계	성인 승차 인원	청소년 승차 인원	어린이 승차 인원	성인 환승 인원	청소년 환승 인원	어린이 승차 인원
	광천터미널	2001	2626061	1612868	186236	10975	704780	105422	5780

전처리

→ 환승 인원의 10%만을 추출하여 승차 인원과 통합

(b) 데이터 B

데이터명	광주광역시 BIS 정류소 정보						
내용 (예시)	연번	한글명	영문명	위도	경도	정류소 번호	방향
	2513	동원촌	Dongwonchon	126.927	35.142	5396	비아동 주민센터

전처리

→ 데이터 A에 있는 정류소 번호를 데이터 B에서 검색하고 정류소 번호 앞에 있는 두 데이터(위도, 경도) 추출

(c) 최종 데이터

데이터명	프로젝트 사용 데이터 bus_stop_data.csv							
내용 (예시)	번호	정류소명	총계	성인 승차 인원	청소년 승차 인원	어린이 승차 인원	위도	경도
	2001	광천터미널	1891677	1683346	196778	11553	128.87	35.16

4. 프로그램 설계 (코딩문은 마지막에 포함)

(a) 프로그램 구성

인공지능 (ai.py) : 프로그램의 전반적인 실행

지도 시각화 (map.py) : 추출된 데이터를 지도에 표시

막대 그래프 시각화 (bar.py) : 추출된 데이터를 막대 그래프로 시각화

→ 지도와 막대 그래프 프로그램은 클래스로 선언하여 인공지능의 활용도 상승

(b) 지도 시각화 프로그램

구조 (Visualize_map)

→ 대상 연령층을 기준으로 전체 정류소의 이용 현황 표시

→ 범위를 지정하여 정류소의 이용 현황 표시

→ 추천, 비추천 지역 정류소 표시

→ 정류소 이용 비율 계산 및 정류소 군집화 지도 표시

(c) 막대 그래프 시각화 프로그램

구조 (Bar_G)

→ 대상 연령층을 기준으로 상위 10개를 추천하는 막대 그래프

→ 대상 연령층을 기준으로 하위 10개를 비추천하는 막대 그래프

(d) 인공지능

구조

1) 프로그램 안내문

2) 사용자 데이터 입력

3) 버스 정류소 데이터 추출

4) 데이터 분석을 통한 막대 그래프 출력

5) 데이터 분석을 통한 지도 출력

→ 전체 이용객을 기준으로 전체 정류소의 이용 현황 표시 지도

→ 대상 연령층을 기준으로 전체 정류소의 이용 현황 표시 자도

→ 대상 연령층을 기준으로 상위 5%와 하위 5%의 이용 정류소 현황 표시 지도

6) 결론

→ 상위 7개의 정류소 정보 출력 : 정류소명 ¦ 해당 정류소의 대상 연령층 이용 비율 ¦ 인접 정류소

→ 군집화 지도

5. 프로그램 분석 및 결론

(a) 프로젝트 분석

결과물의 완성도

(1) 프로젝트 목표 달성

→ 사용자에게 알맞은 내용의 추천 지역과 비추천 지역을 다양한 방식으로 제공

(2) 발표 자료 및 결과물의 완성도

→ 프로그램을 충분히 설명하는 발표 자료와 목적을 잘 수행하는 결과물 완성

(3) 분석 난이도

→ 위도와 경도를 추출하는 프로그램을 추가적으로 만드는 과정으로 난이도 상승

→ 다양한 시각화 자료 구현을 통해 난이도 상승

→ 분석하기에 충분한 데이터와 방식으로 해당 과정 이수 학생에게 맞는 난이도

분석 방법의 적절성

(1) 올바른 방법의 분석

→ 적절한 데이터 분석 방법 사용

(2) 다양한 방법론 시도

→ 여러가지 시각화 자료를 조사하여 직접 수행

→ 데이터 전처리 과정에서 여러가지 방법으로 위도와 경도 데이터 추출

(3) 분석의 신뢰성

→ 신뢰할만한 데이터 사용

→ 적절한 프로그래밍 함수들을 사용하여 Sorting과 추출 실행

분석 결과의 창의성

(1) 분석에 대한 결론

→ 사용자 입력 데이터로부터 적절한 결론 도출

(2) 인사이트

→ 향후 추가 분석이나 여러 데이터들과의 병합으로 프로그램 개선 가능

→ 현 분석으로 통해 유동인구가 많은 지역에 편의시설 설치나
어린이 유동인구가 많은 도시 외각 지역에 치안을 강화하는등 결과 활용 가능

(3) 분석 결과의 활용

→ 분석한 데이터를 다양한 방면(시각화, 추천, 비추천)으로 활용

(b) 프로젝트 결론 (SWOT 분석)

Strength

→ 정보의 신뢰도 : 광주광역시 제공

→ 충분한 정보량 : 3000개 이상의 정보

Weakness

→ 정보의 시대성 : 2016년도 정보

→ 정보의 누락 : 카드 탑승객 이외의 탑승객에 대한 정보 수집 불가

→ 정보의 한계 : 조사 데이터 이외의 자료 미사용

Opportunity

→ 추가적 분석 : 다양한 데이터와 접목을 시켜서 추가적인 데이터 분석 가능

→ 정보의 활용 : 다양한 방면으로 데이터를 활용하여 광주시 환경 개선 가능

Threat

→ 동적인 정보 : 유동인구와 같은 데이터는 주시간의 흐름에 따라 변화

→ 정보의 변화 : 정류소가 신설되거나 철거

6. Code

ai.py

ai.py 프로그램은 본 프로젝트의 인공지능을 담당하는 프로그램 파일입니다.

```
# module
import pandas as pd
import matplotlib as mpl
from bar import Bar_G
from map import Visualize_map
from matplotlib import pyplot

# Main Code

# < 프로그램 안내 >
print('< 버스 정류소 인원 데이터 분석 프로그램 >')
print('-------------------------------------------------------------------------------')
print('본 AI 프로그램은 버스 정류소 이용객을 기준으로 유동인구를 파악합니다.')
print('파악된 유동인구를 바탕으로 사용자가 입점하고자하는 시설을 분석합니다.')
print('분석된 데이터를 통해 입점 추천지와 비추천지를 알려줍니다.')
print('-------------------------------------------------------------------------------\n')

# < 사용자 데이터 입력 >
print('< 사용자 데이터 입력 >')
print('-------------------------------------------------------------------------------')
print('입점 시설의 명칭과 주요 이용 연령층을 입력하세요.')
name = input('입점 시설의 이름을 입력하세요 : ')
print('입점 시설의 주요 이용 연령층을 입력하세요.')
target = int(input('0. 전체  | 1. 성인 | 2. 청소년 | 3. 어린이 : '))
print('-------------------------------------------------------------------------------\n')

# < 버스정류소 데이터 불러오기 >
bus_stop_data = pd.read_csv('Project/bus_stop_data.csv')
dfo = pd.DataFrame(bus_stop_data)

# < 막대 그래프를 통한 분석 결과 시각화 >
# -----------------------------------------------------------------------------
print('< 막대 그래프 시각화 >')
print('-------------------------------------------------------------------------------')
# Font
pyplot.rc('font', family = 'Malgun Gothic')
mpl.rcParams['axes.unicode_minus'] = False

# Z. Bar
bar_target = Bar_G(dfo)
```

```
# A. Target 기준 상위 10개
print('# 타겟 연령층의 상위 10개 정류소 그래프입니다.')
bar_target.bar_rcmd(target)
print('-------------------------------------------------------------------------------\n')
# -----------------------------------------------------------------------

# < Map을 통한 분석 결과 시각화 >
print('< Map을 통한 시각화 >')
# -----------------------------------------------------------------------
print('-------------------------------------------------------------------------------')

# A. 전체 데이터를 시각화 (총계 기준)
print('1. 총계 기준 전 버스 정류소 이용객 현황 지도입니다.')
map_all = Visualize_map(dfo)
map_all.mapping_range(0, 0, 2000, 'green')
map_all.mapping_show('map_all')

# B. Target의 전체 분포 현황
target_str = ['전체', '성인', '청소년', '어린이']
print('2.', target_str[target], '이용객 비율별 정류장 표시 지도입니다.')
map_target_all = Visualize_map(dfo)
map_target_all.mapping_target(target, 'purple')
map_target_all.mapping_show('map_target_all')

# B. 입점 추천 위치 시각화 (상위 5% 하위 5%)
print('3. 타겟 연령층 상위 5%(Blue) | 타겟 연령층 하위 5%(Red)입니다.')
map_target_cmd = Visualize_map(dfo)
map_target_cmd.mapping_rcmd(target, 100, 'blue')
map_target_cmd.mapping_dcmd(target, 1900, 'red')
map_target_cmd.mapping_show('map_target_cmd')
print('-------------------------------------------------------------------------------\n')
# -----------------------------------------------------------------------

# < 결론 도출 >
# -----------------------------------------------------------------------
print('< 시설 입점지를 추천합니다. >')
print('-------------------------------------------------------------------------------')
target_color = ['red', 'orange', 'yellow', 'green', 'blue', 'navy', 'purple']
map_seven = Visualize_map(dfo)
for i in range(7):
    map_seven.mapping_dis(target, i, target_color[i], target_str[target],
map_seven.map_rate(target, i))
map_seven.mapping_show('map_seven')
```

```
print('-------------------------------------------------------------------------------')
# -------------------------------------------------------------------------
```

map.py

map.py 프로그램은 본 프로젝트에서 지도를 시각화하는 프로그램 파일입니다.

```
# Module
import folium
import copy
import math
import webbrowser
import pandas as pd
from pandas import DataFrame

# 전역 함수
def distance(x, y, x1, y1):
    d = math.sqrt((((x - x1) * 10000) ** 2) + (((y - y1) * 10000) ** 2))
    return d

# Class
class Visualize_map :
    # Constructor
    def __init__(self, df) :
        self.df = copy.deepcopy(df)
        self.map_osm = folium.Map(location = [35.16177000, 126.87969568], zoom_start = 13)
        self.target = ['총계', '성인 승차 인원', '청소년 승차 인원', '어린이 승차 인원']
        self.divide_rcmd = [50000, 50000, 10000,1000]
        self.divide_dcmd = [100, 100, 10, 1]

    # Target Map
    def mapping_target(self, select, colorM) :
        df_temp = self.df.sort_values(by = self.target[select], ascending = False)
        for item in df_temp.index:
            lat = df_temp.loc[item, '위도']
            long = df_temp.loc[item, '경도']
            folium.CircleMarker([lat, long],
                                radius = df_temp.loc[item, self.target[select]] /
self.divide_rcmd[select],
                                popup = df_temp.loc[item, '정류소명'],
                                color = colorM,
                                fill = True).add_to(self.map_osm)

    # Recommend Map 0 ~ index
    def mapping_rcmd(self, select, index, colorM) :
        df_temp = self.df.sort_values(by = self.target[select], ascending = False)
        df_temp = df_temp[0 : index]
        for item in df_temp.index:
            lat = df_temp.loc[item, '위도']
```

```
            long = df_temp.loc[item, '경도']
            folium.CircleMarker([lat, long],
                                radius = df_temp.loc[item, self.target[select]] /
self.divide_rcmd[select],
                                popup = df_temp.loc[item, '정류소명'],
                                color = colorM,
                                fill = True).add_to(self.map_osm)

    # Decommend Map index ~ 2000
    def mapping_dcmd(self, select, index, colorM) :
        df_temp = self.df.sort_values(by = self.target[select], ascending = False)
        df_temp = df_temp[index : 2000]
        for item in df_temp.index:
            lat = df_temp.loc[item, '위도']
            long = df_temp.loc[item, '경도']
            folium.CircleMarker([lat, long],
                                radius = df_temp.loc[item, self.target[select]] /
self.divide_dcmd[select],
                                popup = df_temp.loc[item, '정류소명'],
                                color = colorM,
                                fill = True).add_to(self.map_osm)

    # Range Map indexA ~ indexB
    def mapping_range(self, select, indexA, indexB, colorM) :
        df_temp = self.df.sort_values(by = self.target[select], ascending = False)
        df_temp = df_temp[indexA : indexB]
        for item in df_temp.index:
            lat = df_temp.loc[item, '위도']
            long = df_temp.loc[item, '경도']
            folium.CircleMarker([lat, long],
                                radius = df_temp.loc[item, self.target[select]] /
self.divide_rcmd[select],
                                popup = df_temp.loc[item, '정류소명'],
                                color = colorM,
                                fill = True).add_to(self.map_osm)

    # 주요 정류소의 연령층 이용 비율 함수
    def map_rate(self, select, index):
        df_temp = self.df.sort_values(by = self.target[select], ascending = False)
        df_temp = df_temp.reset_index()
        del df_temp["index"]
        p = (df_temp.loc[index, self.target[select]]*100)/
df_temp.loc[0:len(df_temp.index), self.target[select]].sum(axis = 0)
```

```
        return p

    # 주요 정류소의 인접 정류소 병합 함수
    def mapping_dis(self, select, index, colorM, name, rate):
        df_temp = self.df.sort_values(by = self.target[select], ascending = False)
        df_temp = df_temp.reset_index()
        del df_temp["index"]
        df_tempd= DataFrame({'정류소번호':[], '정류소명':[], '총계':[], '성인 승차 인원':[],
'청소년 승차 인원':[], '어린이 승차 인원':[], '위도':[], '경도':[]})
        df_tempd.loc[df_tempd.shape[0]] = [df_temp.loc[index, '정류소번호'],
df_temp.loc[index, '정류소명'], df_temp.loc[index, '총계'], df_temp.loc[index, '성인 승차
인원'], df_temp.loc[index, '청소년 승차 인원'], df_temp.loc[index, '어린이 승차 인원'],
df_temp.loc[index, '위도'], df_temp.loc[index, '경도']]
        i=0
        j=0
        for i in range(1800):
            if (distance(df_temp.loc[index, '위도'], df_temp.loc[index, '경도'],
df_temp.loc[i+1, '위도'], df_temp.loc[i+1, '경도']) < 100):
                df_tempd.loc[df_tempd.shape[j]] = [df_temp.loc[i+1, '정류소번호'],
df_temp.loc[i+1, '정류소명'], df_temp.loc[i+1, '총계'], df_temp.loc[i+1, '성인 승차 인원'],
df_temp.loc[i+1, '청소년 승차 인원'], df_temp.loc[i+1, '어린이 승차 인원'], df_temp.loc[i+1,
'위도'], df_temp.loc[i+1, '경도']]
                j+1
        for item in df_tempd.index:
            lat = df_tempd.loc[item, '위도']
            long = df_tempd.loc[item, '경도']
            folium.CircleMarker([lat, long],
                            radius = df_tempd.loc[item, self.target[select]] /
(self.divide_rcmd[select]),  # -item
                            popup = df_tempd.loc[item, '정류소명'],
                            color = colorM,
                            fill = True).add_to(self.map_osm)
        print("TOP", index+1, " :", df_temp.loc[index, "정류소명"])
        print(name + "의 버스 이용객 중 해당 정류소를 이용하는", name + "의 비율은 %3.2f%%
입니다." %rate)
        print("주변 정류장 :", df_tempd.loc[2, "정류소명"], df_tempd.loc[3, "정류소명"],
df_tempd.loc[4, "정류소명"], ">>", colorM, "\n")

    # Show
    def mapping_show(self, name) :
        self.map_osm.save(name + '.html')
        webbrowser.open_new_tab(name +'.html')
```

bar.py

bar.py 프로그램은 본 프로젝트에서 막대 그래프를 다루는 프로그램 파일입니다.

```
# Module
import copy
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

# Class
class Bar_G :
    # Constructor
    def __init__(self, df) :
        self.df = copy.deepcopy(df)
        self.target = ['총계', '성인 승차 인원', '청소년 승차 인원', '어린이 승차 인원']

    # Recommend Bar Top 10 SELECT
    def bar_rcmd(self, select) :
        df_rcmd = self.df.sort_values(by = self.target[select], ascending = False)
        df_rcmd = df_rcmd.iloc[0:10]
        df_rcmd = df_rcmd.loc[:, ['정류소명', self.target[select]]].set_index('정류소명')
        bar = df_rcmd.plot(kind = 'bar', figsize = (10, 7), legend = True, fontsize = 15)
        bar.set_title(self.target[select] + '기준 승차 데이터 TOP 10', fontsize = 20)
        bar.set_xlabel('버스 정류장명', fontsize = 15)
        bar.set_ylabel('승차인원(수)', fontsize = 15)
        bar.legend([self.target[select]], fontsize = 15)
        plt.show()

    # Decommend Bar Bottom 10 SELECT
    def bar_dcmd(self, select) :
        df_dcmd = self.df.sort_values(by = self.target[select], ascending = True)
        df_dcmd = df_dcmd.iloc[0:10]
        df_dcmd = df_dcmd.loc[:, ['정류소명', self.target[select]]].set_index('정류소명')
        bar = df_dcmd.plot(kind = 'bar', figsize = (10, 7), legend = True, fontsize = 15)
        bar.set_title(self.target[select] + '기준 승차 데이터 TOP 10', fontsize = 20)
        bar.set_xlabel('버스 정류장명', fontsize = 15)
        bar.set_ylabel('승차인원(수)', fontsize = 15)
        bar.legend([self.target[select]], fontsize = 15)
        plt.show()
```

graph_3d.py (사용 X)

graph_3d.py 프로그램은 데이터를 3차원 그래프로 출력하는 프로그램 파일입니다.
해당 파일은 초기에 프로그래밍 하였지만 이후 사용을 하지 않기로 결정하였습니다.

```
# Module
import numpy as np
import pandas as pd
import copy
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Class
class Visualize_3D :
    # Constructor
    def __init__(self, df) :
        self.df = df
        self.t = list(np.array(df['총계'].tolist()))
        self.x = list(np.array(df['성인 승차 인원'].tolist()))
        self.y = list(np.array(df['청소년 승차 인원'].tolist()))
        self.z = list(np.array(df['어린이 승차 인원'].tolist()))

    # All Data 3d Graph
    def graph_3d_all(self) :
        self.graph_show(self.x, self.y, self.z)

    # Range Data 3d Graph
    def graph_3d_range(self, maxI, minI) :
        t_temp = copy.deepcopy(self.t)
        x_temp = copy.deepcopy(self.x)
        y_temp = copy.deepcopy(self.y)
        z_temp = copy.deepcopy(self.z)

        if maxI != 1 :
            r = 0
            while (r < maxI - 1) :
                maximum = t_temp.index(max(t_temp))
                del t_temp[maximum]
                del x_temp[maximum]
                del y_temp[maximum]
                del z_temp[maximum]
                r += 1

        if minI != 2000 :
            r = 0
            while(r < 2000 - minI) :
```

```
                minimum = t_temp.index(min(t_temp))
                del t_temp[minimum]
                del x_temp[minimum]
                del y_temp[minimum]
                del z_temp[minimum]
                r += 1

        self.graph_show(x_temp, y_temp, z_temp)

    # Show Data 3d Graph
    def graph_show(self, x, y, z) :
        fig = plt.figure(figsize = (10, 10))
        ax = fig.gca(projection = '3d')
        ax.scatter(x, y, z, marker = 'o', s = 15, c = 'darkgreen')
        ax.set_xlabel('Adult')
        ax.set_ylabel('Teenager')
        ax.set_zlabel('Child')
        plt.show()
```

dataP.py

dataP.py 프로그램은 위도와 경도를 추출하는 전처리 과정에서 자체 제작한 프로그램입니다.

```
import pandas as pd
import numpy as np
import copy

def isEnglishOrKorean(input_s):
    k_count = 0
    e_count = 0
    for c in input_s:
        if ord('가') <= ord(c) <= ord('힣'):
            k_count+=1
        elif ord('a') <= ord(c.lower()) <= ord('z'):
            e_count+=1
    return k_count + e_count

file = open('Study/Input.txt', 'r', encoding = 'utf8')
a = file.read().split()

dataCSV = pd.read_csv('Study/Output.csv')
dfa = pd.DataFrame(dataCSV)
df = copy.deepcopy(dfa)
listN = list(np.array(df['정류소번호'].tolist()))

for i in range(len(listN)) :
    listN[i] = str(listN[i])
listLat = []
listLong = []
temp = 0

for i in range(2000, 2316) :
    for j in range(3, len(a)) :
        if (a[j] == listN[i]) & (isEnglishOrKorean(a[j - 1]) == 0) & (isEnglishOrKorean(a[j
- 2]) == 0) & (isEnglishOrKorean(a[j - 3]) > 0) :
            listLong.append(float(a[j - 1]))
            temp = 1
            break
    if  temp == 0:
        listLong.append(0)
    temp = 0

print(listLong)
```