비쥬얼프로그래밍

과제 3.

20196014 고지흔

1번

해결방안

-readlines()로 사전의 줄마다 리스트로 가져오고 파일 닫았습니다.

뜻 첫 글자가 대문자 I 인 것들과 뜻에 아무것도 적혀있지 않는 글자 걸러내고 ':' 의 위치 기준 문자열 슬라이싱을 해 단어와 뜻을 분리했습니다. 그리고 단어를 key로 뜻을 value로 놓고 if문을 이용해 단어를 입력하면 사전에 있는지 확인하고 있으면 뜻을 출력, 없으면 찾는 단어가 없다는 메시지를 출력하게 만들었습니다.

소스코드

```
fdic=open('dict_test.TXT','r')
diclist=fdic.readlines()
fdic.close #리스트로 가져오고 파일 닫음
dicdic={} #사전들어갈 딕셔너리, 딕셔너리쓰면 중복 자동 제거
for i in diclist:
    n=i.index(':')
    if n==len(i)-2: #biosph : 제거
        #print("뜻이 담겨있지 않습니다 %s\n" %i[0:n-1])
        continue
    dicdic[i[0:n-1]]=i[n+2:]

    if (dicdic[i[0:n-1]][0]=="I"):
        #print("다음 단어의 뜻이 I 가 아닙니다 : %s, %s" %(i[0:n-1],i[n+2:])) #뜻이 I 로
시작하는 깨진것들 제거
        del dicdic[i[0:n-1]]
while True:
    wantfind=input("단어? ")
    if (wantfind in dicdic):
        print("%s %s" %(wantfind,dicdic[wantfind]),end='')
    else:
        print("찾으려는 단어가 없습니다")
```

결과캡쳐

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\jhk27\OneDrive\바탕 화면\비프과제>  & 'C:\Users\jhk27\AppData\Local\Programs\Python\P
ms-python.python-2022.18.2\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '12880'
'
단어? apple
apple n.사과
단어? avata
찾으려는 단어가 없습니다
단어? star
star n.별
단어? stand
stand vi.서다
단어? hi
hi int.야아
단어? hello
hello int.여보
단어? oh
oh int.야아
단어? slkdfjv
찾으려는 단어가 없습니다
단어? zoo
zoo n.동물원
단어? beer
beer n.맥주
단어? blue
blue a.푸른
단어? black
black a.검은
단어?
```

2번

해결방안

```
dickeylist=list(dicdic.keys())
```
를 이용해 딕셔너리에 키(사전의 단어)만 리스트로 만들었습니다.

```
rannum=random.randrange(7,11)
```
를 사용해 7~10까지의 랜덤한 정수를 추출하고

```
ranlist=random.sample(dickeylist,rannum)
```
를 이용해 랜덤한 7~10개의 단어를 단어들 중에서 뽑은 리스트를 만들었습니다

이 리스트를 for문을 이용해서 조건에 맞는 하나의 문자열로 만들었고

```
datetime.now()
```
를 사용해 해당 문자열이 출력되기 전과 input이 완료된 시점을 기록하고 이 시간을 서로 뺀 이후

```
timedel.total_seconds()
```
를 이용해서 초로 변경했습니다.

타/분 속도는 타이핑한 글자수/걸린시간(초) *60을 했습니다

이후 해당 문자열이 타이핑한 문자열과 같으면 20초가 넘었는지 확인하고 20초가 넘으면 0점 20초이내면 점수부과 방식에 따라 해당 점수를 부여하였습니다

해당 문자열이 타이핑한 문자열과 다르면 문자열 길이와 for문과 ^를 이용해 ^를 가장 처음 틀린 곳에 표시되게 체크했습니다. 그리고 수행이 끝날 때 마다 점수가 누적되게 했고 총 10번 돌아가게 했습니다.

소스코드

```
import random
from datetime import datetime


fdic=open('dict_test.TXT','r')
diclist=fdic.readlines()
fdic.close #리스트로 가져오고 파일 닫음
dicdic={} #사전들어갈 딕셔너리, 딕셔너리쓰면 중복 자동 제거
for i in diclist:
    n=i.index(':')
    if n==len(i)-2: #biosph : 제거
```

```
        continue
    dicdic[i[0:n-1]]=i[n+2:]

    if (dicdic[i[0:n-1]][0]=="I"):
        del dicdic[i[0:n-1]]
dickeylist=list(dicdic.keys())


cnt=0
sum=0
while cnt<10:
    cnt+=1
    rannum=random.randrange(7,11)
    ranlist=random.sample(dickeylist,rannum)
    question=""
    for i in ranlist:
        question=question+i+" "
    question = question[:-1]
    print("(%d/10)타자게임! 사전에서 랜덤으로 선택된 7~10 개의 단어를 그대로 치세요" %cnt)
#사전에 띄어쓰기 있는 단어들이 있어서 10 개 넘어 보일 수도 있음 하지만 10 개 추출
    dtstart = datetime.now()
    print("= %s" %question)
    ans=input("= ")#마지막 띄어쓰기 제거
    dtend=datetime.now()
    timedel=dtend-dtstart
    time=timedel.total_seconds() #타이핑 친 초
    typespeed=len(ans)*60/time #타/초 >>타이핑한 길이/초  //타/분 >> 타/초 * 60
    if question==ans:
        score=(20-time)*1000
        if time>20:
            print("시간 초과! 시간 : %ld" %time)
            score=0
        print("맞았습니다.(%lf 점 획득 %lf 타/분)" %(score, typespeed))
    else:
        score=0
        buf=0
        if len(question)<len(ans):
            min=len(question)
        else:
            min=len(ans)
        for i in range(0,min):
            if question[i]!=ans[i]:
                print(" "*(i+2)+"^")
                buf=1
                break
        if buf==0:
            print(" "*(min+2)+"^")
        print("틀렸습니다.(%lf 타/분)" %typespeed)
    sum=sum+score
print("당신의 총 점수는 %lf 점 입니다." %sum)
```

결과캡쳐

```
(1/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= handcart lowboy jointly aerial beacon geum necrosis dacoit gin mill uncivilized
= handcart lowddfdfdfdfdfdf
             ^
틀렸습니다.(168.679787타/분)
(2/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= detector cannibalize fertilize habituation paramilitary isolation bedridden safety etymology
= detector cannibalize fertilize habituation paramilitary ioslation bedridden safety etymology
                                                          ^
틀렸습니다.(193.121455타/분)
(3/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= kinky affected preventive polaroid sarge line printer beetsugar monarchy fondness endogamy
= kinky affected preventive polaroid sarge line printer beetsugar monarchy fondness endogamy
시간 초과! 시간 : 27
맞았습니다.(0.000000점 획득 196.764334타/분)
(4/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= do over commercialize sleepy etcher chop off intermixture floozie basilicon popery
= do over commercilalize
                ^
틀렸습니다.(92.449942타/분)
(5/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= soporiferous dismally restring condensed milk dishonorable aural insipidity ouzel
= soporiferous dismally restring condensed milk dishonorable aural insipidity ouzel
시간 초과! 시간 : 24
맞았습니다.(0.000000점 획득 196.824938타/분)
(6/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= rapprochement satinette miliary absently aboriginally edgily painful loculus aft reel off
= rapprochement satinette miliary absently aboriginally edgily painful loculus aft reel off
시간 초과! 시간 : 30
맞았습니다.(0.000000점 획득 174.426695타/분)
```

```
(7/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= jr. cabinetwork inclusively wordbook readout sidetrack opalescence
= jr. cabinetwork inclusively wordbook readout sidetrack opalescence
시간 초과! 시간 : 27
맞았습니다.(0.000000점 획득 144.935270타/분)
(8/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= open letter platen devilishness enhancement consolation prize tea cosy laconism adurol
= open letter platen devilishnees enhancement consolation prize tea cosy laconism adurol
                               ^
틀렸습니다.(172.171960타/분)
(9/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= dicebox tabinet vanishing point calisthenic sallowish cut down overdid denounce
= dicebox tabinet vanishing point calisthenic sallowish cut down overdid denounce
시간 초과! 시간 : 23
맞았습니다.(0.000000점 획득 205.017345타/분)
(10/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= apportionment adhere bewitching quinidine black tea solitary natively adjutancy tri-jet airliner
= apportionment adhere bewitching quinidine black tea solitary natively adjutancy tri-jet airliner
시간 초과! 시간 : 33
맞았습니다.(0.000000점 획득 172.419501타/분)
당신의 총 점수는 0.000000점 입니다.
```

타자가 느려서.... 시간내로 친 경우에는

```
(5/10)타자게임! 사전에서 랜덤으로 선택된 7~10개의 단어를 그대로 치세요
= photophase kind liveable parachutist powerful friar scrape
= photophase kind liveable parachutist powerful friar scrape
맞았습니다.(2347.957000점 획득 197.144319타/분)
```

이렇게 나옵니다

3번

해결방안

국가별 그리고 날짜별로 데이터가 최신화 된다는 것을 이용했습니다

```
firstline=covidf.readline()
```
를 이용해 가장 첫줄의 데이터를 가져오고 <<국가...대륙...총 확진자..등

```
korline=covidf.readline()
```
를 이용해 KOR를 찾고 <<한국의 가장 첫날 코로나 확진자 데이터

이후 다시 while 문을 이용해

```
if korline[0:num]!="KOR":
```
이면 break를 해 가장 최신의 코로나 확진자 데이터를 찾을 수 있습니다.

이와 맨 첫줄을 리스트로 바꿔 출력하면 잘 정렬된 데이터로 확인할 수 있습니다.

소스코드

```
covidf=open('owid-covid-data.csv','r')
firstline=covidf.readline()
while True:
    korline=covidf.readline()
    num=korline.find(",")
    if korline[0:num]=="KOR":
        break #한국 가장 처음까지 찾기

while True:
    korline=covidf.readline()
    num=korline.find(",")
    if korline[0:num]!="KOR":
        break #한국 다음 국가가 나오면 중단 ,현황이 나라순으로 정렬됨을 이용
    newestline=korline

firstlinelist=firstline.split(",")
newestlinelist=newestline.split(",")
print("한국의 코로나 최신정보")
for i in range(0,len(firstlinelist)):
    print("%s : %s" %(firstlinelist[i],newestlinelist[i]))

covidf.close()
```

결과캡쳐

```
한국의 코로나 최신정보
iso_code : KOR
continent : Asia
location : South Korea
date : 2022-10-26
total_cases : 25431105.0
new_cases : 34950.0
new_cases_smoothed : 30222.714
total_deaths : 29069.0
new_deaths : 26.0
new_deaths_smoothed : 21.0
total_cases_per_million : 490662.489
new_cases_per_million : 674.318
new_cases_smoothed_per_million : 583.111
total_deaths_per_million : 560.851
new_deaths_per_million : 0.502
new_deaths_smoothed_per_million : 0.405
reproduction_rate :
icu_patients :
icu_patients_per_million :
hosp_patients :
hosp_patients_per_million :
weekly_icu_admissions :
weekly_icu_admissions_per_million :
weekly_hosp_admissions :
weekly_hosp_admissions_per_million :
total_tests :
new_tests :
total_tests_per_thousand :
new_tests_per_thousand :
new_tests_smoothed :
new_tests_smoothed_per_thousand :
positive_rate :
tests_per_case :
tests_units :
total_vaccinations : 129468065.0
people_vaccinated : 45119824.0
people_fully_vaccinated : 44689199.0
total_boosters : 41176694.0
new_vaccinations : 5782.0
new_vaccinations_smoothed : 5685.0
```

```
total_vaccinations_per_hundred : 249.79
people_vaccinated_per_hundred : 87.05
people_fully_vaccinated_per_hundred : 86.22
total_boosters_per_hundred : 79.45
new_vaccinations_smoothed_per_million : 110.0
new_people_vaccinated_smoothed : 193.0
new_people_vaccinated_smoothed_per_hundred : 0.0
stringency_index :
population_density : 527.967
median_age : 43.4
aged_65_older : 13.914
aged_70_older : 8.622
gdp_per_capita : 35938.374
extreme_poverty : 0.2
cardiovasc_death_rate : 85.998
diabetes_prevalence : 6.8
female_smokers : 6.2
male_smokers : 40.9
handwashing_facilities :
hospital_beds_per_thousand : 12.27
life_expectancy : 83.03
human_development_index : 0.916
population : 51830139.0
excess_mortality_cumulative_absolute :
excess_mortality_cumulative :
excess_mortality :
excess_mortality_cumulative_per_million
 :
```