과제 1-1, 1-2, 1-3, 2 코드

2020142001 곽종근

과제 1-1. T출력

```
# T 출력
import numpy as np            #numpy를 np라는 이름으로 사용하겠다.
import pandas as pd           #pandas를 pd라는 이름으로 사용하겠다.
import matplotlib.pyplot as plt        #matplotlib.pyplot를 plt라는 이름으로 사용하겠다.

A=np.zeros([100,100])    #A라는 100by100의 zeros배열을 선언해놓는다.


for j in range(0,100,1):          #파일이 총 100개가 있기때문에 이 100개의 파일을 모두 사용하기 위해 0~99까지 for문(반복문)을 이용해 전부 선언, 오름차순이므로 0부터
    fold_dir="C:\\Users\\user\\OneDrive - 한국공학대학교\\바탕 화면\\3학년 1학기\\머신러닝실습\\4주차\\problem_1_data\\"
    file_name=str(j)+".csv"        #반복문 j를 j.csv로 선언하여 0~99.csv가 전부 불러와질 수 있도록 선언한다.
    final_file=fold_dir+file_name      #fold_dir과 fie_name을 합쳐 파일이 해당하는 위치의 0~99개 파일을 불러오는 final_str을 최종 선언

    temp_data=pd.read_csv(final_file, header=None)  #final_file이라는 최종 파일에 있는 데이터를 읽어오는데 헤더파일을 생성하지 않고 읽어온다. dataframe형태
    temp_data=temp_data.to_numpy()  #위에서 불러온 파일은 dataframe형태이므로 슬라이싱이 안되기에 이를 numpy배열로 변환시킨다.
    A[:,j]=temp_data[:,25]  #위에서 numpy배열로 변환해주었기때문에 슬라이싱이 가능하고 dataframe형태는 불가능하다.
                            #25번째 col을 행렬 A의 0번부터 99번까지 col에 삽입한다.


plt.imshow(A, cmap='viridis')
plt.axis('off')
plt.show
```

과제 1-2. U출력

```
#U출력
import numpy as np  #numpy를 np라는 이름으로 사용하겠다.
import pandas as pd #pandas를 pd라는 이름으로 사용하겠다.
import matplotlib.pyplot as plt  #matplotlib.pyplot를 plt라는 이름으로 사용하겠다.

B=np.zeros([100,100]) #B라는 100by100의 zeros배열을 선언해놓는다.


for j in range(0,100,1): #파일이 총 100개가 있기때문에 이 100개의 파일을 모두 사용하기 위해 0~99까지 for문(반복문)을 이용해 전부 선언,
오름차순이므로 0부터
    fold_dir="C:\\Users\\user\\OneDrive - 한국공학대학교\\바탕 화면\\3학년 1학기\\머신러닝실습\\4주차\\problem_1_data\\"
    file_name=str(j)+".csv" #반복문 j를 j.csv로 선언하여 0~99.csv가 전부 불러와질 수 있도록 선언한다.
    final_file=fold_dir+file_name  #fold_dir과 fie_name을 합쳐 파일이 해당하는 위치의 0~99개 파일을 불러오는 final_str을 최종 선언

    temp_data=pd.read_csv(final_file, header=None) #final_file이라는 최종 파일에 있는 데이터를 읽어오는데 헤더파일을 생성하지 않고 읽어온다.
dataframe형태
    temp_data=temp_data.to_numpy() #위에서 불러온 파일은 dataframe형태이므로 슬라이싱이 안되기에 이를 numpy배열로 변환시킨다.
    B[j,:]=temp_data[10,:] #위에서 numpy배열로 변환해주었기때문에 슬라이싱이 가능하고 dataframe형태는 불가능하다.
                           #25번째 col을 행렬 A의 0번부터 99번까지 col에 삽입한다.



plt.imshow(B, cmap='viridis')
plt.axis('off')
plt.show
```

과제 1-3. K출력.

```
#K출력
import numpy as np                  #numpy를 np라는 이름으로 사용하겠다.
import pandas as pd    #pandas를 pd라는 이름으로 사용하겠다.
import matplotlib.pyplot as plt #matplotlib.pyplot를 plt라는 이름으로 사용하겠다.

C=np.zeros([100,100])  #C라는 100by100의 zeros배열을 선언해놓는다.


for j in range(0,10,1):                    #T와 U의 출력에서는 행 또는 열만 바뀌었는데 K의 출력은 각 행과 열에 해당하는 값을 넣어주어야 하므로 이중배열로 만들어 주기 위해
                                            #이중for문을 만들어 준다. j가 먼저 실행되고 그 뒤 i가 0~9까지 실행이 된 후 j가 1로 넘어가는 순서이므로 j를 10의자리 수, i를 1의자리 수 라고
                                            #판단이 가능하다. 따라서 아래와 같은 for문을 작성하였다.
    for i in range(0,10,1):
        fold_dir="C:\\Users\\user\\OneDrive - 한국공학대학교\\바탕 화면\\3학년 1학기\\머신러닝실습\\4주차\\problem_1_data\\"
        file_name=str(i+j*10)+".csv"      #j에 10을 곱해주면 j는 i가 0~9까지 변할 때 j는 0, 10, 20 ~ 90의 순서로 변화하므로 이는 0~99까지 변화하고 모든 파일을 전부 반복문으로 입력받을 수 있다.
        final_file=fold_dir+file_name      #fold_dir과 fie_name을 합쳐 파일이 해당하는 위치의 0~99개 파일을 불러오는 final_str을 최종 선언
        temp_data=pd.read_csv(final_file, header=None)  #final_file이라는 최종 파일에 있는 데이터를 읽어오는데 헤더파일을 생성하지 않고 읽어온다. dataframe형태로 읽어와진다.
        temp_data=temp_data.to_numpy()       #위에서 불러온 파일은 dataframe형태이므로 슬라이싱이 안되기에 이를 numpy배열로 변환시킨다.
        C[j*10:(j*10)+10,i*10:(i*10)+10]=temp_data[70:80,80:90]         #위에서 numpy배열로 변환해주었기때문에 슬라이싱이 가능하고 dataframe형태는 불가능하다.

 #C에 temp_data의 70~79, 80~89번째의 데이터를 10by10으로 C에 넣어줘야 하는데 이를 위해 직접 배열을 나눠 보았더니
 #        0번째           1번째             2번째        3번째        4번째           5번째           6번째         7번째           8번째          9번째

 # C[0:10, 0:10] [0:10, 10:20] [0:10, 20:30] [0:10, 30:40] [0:10, 40:50] [0:10, 50:60] [0:10, 60:70] [0:10, 70:80] [0:10, 80:90] [0:10, 90:100]
 #         10              11              12              13              14          15              16          17              18          19
```

```
 # C[10:20, 0:10] [10:20, 10:20] [10:20, 20:30] [10:20, 30:40] [10:20, 40:50] [10:20, 50:60] [10:20, 60:70] [10:20, 70:80] [10:20, 80:90] [10:20, 90:100]
 # C[20:30, 0:10] [20:30, 10:20] [20:30, 20:30] [20:30, 30:40] [20:30, 40:50] [20:30, 50:60] [20:30, 60:70] [20:30, 70:80] [20:30, 80:90] [20:30, 90:100]
#이러한 형태의 변화가 나타나는데 여기서 보면, 일단 for문이 j먼저 선언되면 i가 0~9까지 변화 후 j가 변화하는 것을 이용하여 먼저 행 부분에서는 j*10으로 시작해서 (j*10)+10을 해 주면 각 변화하는 값에 맞춰
#10개의 행을 선언하게 되고, 열은 i*10으로 시작하여 (i*10)+10의 형태로 10개의 열이 되어 총 10by10의 행렬에 temp_data값을 넣어줄 수 있다. 이중for문의 특성으로 값이 밀린다고 생각하면 쉽다.


plt.imshow(C, cmap='viridis')
plt.axis('off')
plt.show
```

과제 2. 출력

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt



fold_dir="C:\\Users\\user\\OneDrive - 한국공학대학교\\바탕 화면\\3학년 1학기\\머신러닝실습\\4주차\\problem_2_data.csv"
temp_data=pd.read_csv(fold_dir)


# NaN 값을 ''으로 대체하여 300x5 크기를 유지
new_td = temp_data.fillna('')

total_counts = np.zeros([1,5])     #값을 1by5에 저장하라고 하였으므로 1,5의 빈 배열을 선언한다. 전체 데이터 값
sam_counts = np.zeros([1,5])       #위와 같은 내용이고 이 부분에는 샘플링 값들만 넣어 저장


for i in range(5):
    total_counts[0, i] = sum(new_td.iloc[:, i] != '')     #i열에서 nan을 대체한 ''값을 제외하고 남은 데이터의 개수를 구한다
    sam_counts[0, i] = sum(new_td.iloc[:, i] != '') // 2  #열에서 nan을 대체한 ''값을 제외하고 남은 데이터의 개수를 구한다
                                                          #샘플링한 값이고, 2초의 시간이라 하였으므로 2로 나눠준 값이 저장된다.



fmin = np.min(sam_counts)  #np.mi()을 이용해 괄호 안에 최소값을 구하고자 하는 배열을 넣으면 된다.

DS_data=np.zeros([300,5])  #300by5의 제로스 빈 배열을 선언해준다.
```

```
for i in range(5):
    # 다운샘플링 간격 계산
    # 현재 열의 전체 데이터 수인 total_counts를 30으로 나눈 값을 간격으로 설정

    DS_interval = int(sam_counts[0, i] // fmin)

    # 빈 문자열을 가진 행을 제외하고 다운샘플링하여 DS_data에 저장
    # nan값을 ''으로 설정하였기에 먼저 데이터프레임이므로 iloc[:,i]를 이용해 현재 신호의 i번째 열 선택
    # 그 다음 조건으로 ''을 제외한 값을 선택하고 이 선택된 부분중 다운샘플링 간격으로 샘플링을 한다. 이 모든 조건을 대괄호[]을 붙여서 사용해주면 한줄에 작성이 가능하다.
    DS_col = new_td.iloc[:, i][new_td.iloc[:, i] != ''][::DS_interval]
    DS_data[:len(DS_col), i] = DS_col

plt.figure(figsize=(10,6))
for i in range(5):
    plt.plot(np.arange(0,2,1/fmin),DS_data[0:60,i])
plt.title("Signal Graphs")
plt.xlabel("Time [s]")
plt.ylabel("Value")
plt.grid(True)
plt.legend(["Signal 1","Signal 2","Signal 3","Signal 4","Signal 5"],loc="upper right")
```