[SWCON104-00] 2021 년 1 학기 기말고사

비대면 시험의 특성상 시험감독이 문제에 대한 질문을 받지 않습니다. 이 문서는 시험지이고, 답안파일(ipynb)이 따로 있으니 반드시 열어서 확인하고, 해당 파일에 코드를 작성하여 제출하세요. 문제에 명시된 조건을 모두 만족하는 python 코드를 주어진 jupyter notebook 답안파일에 구현하세요. 명시되지 않은 부분은 필요에 따라 자유롭게 구현하세요. 코드의 길이, 실행시간, 수업 때 배우지 않은 내용의 포함 등은 채점에 영향을 끼치지 않습니다. 답안파일의 실행예시 코드가 문제없이 실행되어야 하며, 출력내용이 정확히 동일해야 합니다.
함수명, 변수명은 문제에 명시된 것과 반드시 동일해야 합니다. 대소문자, 스펠링 틀리면 감점합니다.

[문제 1] 도시 정보를 관리하는 class City 와 나라 정보를 관리하는 class Country 를 각각 구현하시오.

1) class City 구현

- Class City 는 __init__ , __str__ , density 함수(method)를 가진다. 다른 추가함수 구현금지.
- class City 는 __init__ 함수에서 멤버변수(instance variable) name, population, area 를 초기화한다. 이 멤버변수들은 객체를 생성하며 받은 파라미터 값으로 초기화된다. (실행예시 참조) 그외 멤버변수는 없다.
- __str__ 함수를 구현하시오. 실행예시를 참고하여 동일한 문장으로 출력되도록 한다. 띄어쓰기, 스펠링, 마침표 등이 모두 동일하도록 주의한다.
- density 함수를 구현하시오. 멤버변수 population 에서 area 를 나누어 소수점 둘째자리까지의 값으로 반올림하여 반환한다.

2) class Country 구현

- Class Country 는 아래 설명된 6 개의 함수(method)를 가진다. 다른 추가함수 구현금지.
- Class Country 는 __init__함수에서 멤버변수 name 과 cities 를 초기화한다. 이 멤버변수들은 객체를 생성하며 받은 파라미터 값으로 초기화되지만, cities 는 주어지지 않을 경우 empty list 로 초기화된다. (실행예시 참조) 그외 멤버변수는 없다.
- num_cities 함수를 구현하시오. 멤버변수 cities 는 list 인데, 이 list 의 길이를 반환한다.
- total_population 함수를 구현하시오. 멤버변수 cities 에 포함된 city 들의 population 을 모두 더하여 반환한다.
- total_area 함수를 구현하시오. 멤버변수 cities 에 포함된 city 들의 area 를 모두 더하여 반환한다.
- density 함수를 구현하시오. total_population 반환값에서 total_area 반환값을 나누어 소수점 둘째자리까지의 값으로 반올림하여 반환한다.
- __str__ 함수를 구현하시오. 멤버변수 cities 에 포함된 city 들의 __str__ 반환값들을 모두 연결하여 한번에 출력되도록 한다. (실행예시 참조)

[문제 2] 주어진 score1.txt 파일에는 학생들의 이름, 숙제, 중간고사, 기말고사 점수가 차례로 적혀있다. 첫째줄은 자료의 설명이고 둘째줄부터 학생별 점수가 한 줄당 한 명씩 적혀있다. 코드 구현 전에 txt 파일을 오픈하여 자료의 구성을 확인하시오. 채점시 다른 파일들로도 test 를 할 것이며, 각 파일의 구성은 동일하나 학생의 수는 다를 것이다. 아래와 같이 make_dictionary 함수와 sort_by_total 함수를 구현하시오.

1) make_dictionary 함수 구현: 파일을 읽어서 점수로 dictionary 를 만들어 반환하는 함수

함수의 입력 파라미터는 파일이름(string 형)이다. 이 함수는 해당 파일을 열어 dictionary 를 만들어 반환한다. 반환하는 dictionary 는 학생 이름을 key 로 하고, 숙제, 중간고사, 기말고사 점수를 하나의 list 로 만들어 이를 value 로 한다. 이 list 의 길이는 항상 3 이다. 단, list 안의 점수들은 반드시 int 형이어야 한다. (실행예시 참조)

2) sort_by_total 함수 구현: 총점이 높은 순서대로 (학생이름, 총점) tuple 을 list 로 반환하는 함수

함수의 입력 파라미터는 make_dictionary 함수로 만든 dictionary 이다. 해당 dictionary 에는 학생 이름과 학생의 숙제, 중간고사, 기말고사 점수가 하나의 list 로 만들어져 담겨있다. 이 함수는 학생당 총점을 구하여 **(학생이름, 총점)** 형태의 tuple 로 만들고, 총점이 높은 순서대로 이 tuple 들을 하나의 list 에 담아 반환한다. 이때, 학생이름은 string 형, 총점은 int 형이어야 한다. 총점이 같은 학생은 없다고 가정한다. 동명이인도 없다고 가정한다. (실행예시 참조)

========== 끝 ==========

아래는 영문 번역본입니다. 위의 내용과 동일합니다. Below is the English version.

We do not take questions regarding the final exam problems during the online test. This file contains the problems, while the answer sheet is also provided as .ipynb file. You should open the answer sheet file, read what's inside and write your code in that file. You should complete the python code to satisfy all the conditions mentioned in the problem. The execution example code inside the answer sheet must run and print the same result with your implementation. All names of functions and variables mentioned in the problem must be the same as required. Otherwise, your point will be deducted.

[Problem 1] Implement class City and class Country as instructions below.

1) class City

- Class City has only three methods: __init__ , __str__ , and density. No other methods are allowed.
- __init__ method initializes three instance variables: name, population, and area. These variables are initialized as the input parameters when the object is created. Refer to the execution example. No other instance variables allowed.
- Implement the __str__ method so that the object is printed as shown in the execution example.
- density method returns population density. It is calculated by dividing population by area. Round the result to only two decimals.

2) class Country

- Class Country has only six methods explained below. No other methods are allowed.
- __init__ method initializes two instance variables: name and cities. These variables are initialized as the input parameters when the object is created, but cities may or may not be given. Cities is initialized as an empty list if not given. Refer to the execution example. No other instance variables are allowed.
- num_cities method returns the length of the cities list.
- total_population method returns the sum of populations of the cities.
- total_area method returns the sum of areas of the cities.
- density method returns the population density calculated by dividing the total_population by the total_area. Round the result to only two decimals.
- Implement the __str__ method so that the object is printed as shown in the execution example. All the sentences of the cities are attached together.

[Problem 2] A text file named score1.txt is as input data. The text file contains names and scores of homework, midterm, and final exams of several students. The first line of the file is the content explanation, and the actual data

starts from the second line. Each line contains information of one student. Open and check how the data is written in the text file before you begin writing your code. Your code will be tested using other files with different number of students.

1) Implement make_dictionary function

This function reads a text file, make a dictionary, and returns it. The function takes a filename in string type as input parameter. The function opens the file of that filename, and make a dictionary whose keys are student names, and the values are lists of each student's scores. Each value of dictionary should be a list of length 3, which contains the scores of homework, midterm, and final of each student. The score should be integer type. Refer to the execution example.

2) Implement sort_by_total function

This function receives a dictionary created by make_dictionary function as input parameter. The dictionary contains student names as keys and lists of scores as value. Sort_by_total function calculates the total score for each student and make a tuple as **(name, total score)** for each student. Then, the function makes a list of these tuples sorted by the descending order of the total score. The function returns this list. Refer to the execution example.