Computing Foundations for Data Science

HW 1

Due date: 2022/10/06 11:59 PM

주의사항

- 코드를 Jupyter Notebook 에서 작성하였더라도 **python 파일(.py)로 제출할 것**.
- 함수가 의도한 값을 return 하는지를 확인할 것. (print 와 혼동하지 말 것)
- 파일명은 P1.py, P2.py, P3.py 를 유지하고, 해당 파일들을 HW1_학번_이름.zip 으로 압축하여 제출할 것. 예를 들면 학번이 2022-12345 이고, 이름이 Hyunwoo Park 이라면 **HW1_2022_12345_HyunwooPark.zip** 로 압축하여 제출.
- 제출 형식 오류로 인한 채점 누락은 책임지지 않음.
- 예시로 제시한 입력 값 외에도 조교가 임의로 생성한 입력 값으로도 코드가 잘 실행되는지 테스트할 예정.
- 뼈대 코드의 함수 이름 및 매개변수(parameter)는 변경하지 말 것.
- 기본 모듈 외 사용 금지 (ex. Collections)
- 자세한 사항은 파일의 함수 설명 참고.

Instructions

- **Submit the file with format of python file(.py)** NOT the Jupyter Notebook file(.ipynb).
- Check the function returns the right results. (Do NOT be confused with print().)
- The names of python files should remain P1.py, P2.py, P3.py and compress your file as file named “HW1_your student ID_your name.zip”. ex) **HW1_2022_12345_HyunwooPark.zip**
- Submission type error will cause grading missing.
- TA will test your codes by applying other test samples to your codes.
- Do NOT change the name of the functions and parameters given.
- Stay in basic modules. (Collections module is not allowed.)
- Further details are given in the assignment file.

문제 1

작성된 bank class를 참조하여, 아래에 account class 를 완성하시오.
Finish the class Account thoroughly by referring to class Bank.

1-1) account의 __init__ 함수를 완성하시오.
Complete __init__ () method in Account class.

1-2) deposit method를 완성하시오.
Complete the form of deposit () method in Account class.

Deposit method 안에는 다음과 같은 조건이 있다.
Amount 가 0이하인 경우 : "Your request was {amount}. Please check your request." 와 같은 경고문을 출력한다.
Amount 가 0 이상인 경우 : account 의 balance를 증가시키고 bank의 Balance 정보를 업데이트한다. 그 결과 customers와 total_balance 가 변경되어야 한다. 그리고 “Save your money in the bank : {amount}” 와 같은 문구를 출력하도록 한다.

Conditions in Deposit method () are:
If the Amount is under 0 or the same, print a warning like "Your request was {amount}. Please check your request.".
If the Amount is more than 0, amount will increase, the information of Balance will be updated. Therefore, customers and total_balance will eventually change. Last, print “Save your money in the bank: {amount}”.

1-3) withdraw method를 완성하시오.
Complete the form of withdraw () method in Account class.

Withdraw method 안에는 다음과 같은 조건이 있다.
Amount 가 0이하로 들어오면 "Your request was {amount}. Please check your request." 와 같은 경고문을 출력한다.
Amount 가 0 이상인 경우 account 의 balance를 감소시키고 bank의 Balance 정보를 업데이트한다. 그 결과 customers와 total_balance 가 변경되어야 한다. 그리고 "Withdraw your money from the
bank : {amount}"를 최종적으로 출력한다.

Conditions in withdraw () method are:
If the Amount is under 0 or the same, print a warning like "Your request was {amount}. Please check your request.".
If the Amount is more than 0, amount will decrease, the information of Balance will be updated. Therefore, customers and total_balance will eventually change. Last, print “Save your money in the bank: {amount}”.

Output Example

```
gsdsBank = Bank()
myAccount = Account('holideez', gsdsBank, 'saving', 5000)
print(myAccount.owner)
# >>> holideez
print(myAccount.accountType)
# >>> saving
print(gsdsBank.total_accounts)
# >>> {'checking': 0, 'saving': 1}
print(gsdsBank.total_balance)
# >>> 5000
print(gsdsBank.customers)
# >>> {'holideez': 5000}
myAccount.deposit(-1000)
# >>> Your request was -1000. Please check your request.
myAccount.deposit(1000)
# >>> Save your money in the bank : 1000
myAccount.withdraw(10000)
# >>> your request exceeds your balance.
myAccount.withdraw(2000)
# >>> Withdraw your money from the bank : 2000
```

문제 2

MBTI는 각 성격 요소(예를 들어 외향성, 내향성, 직관, 감각 등)에 대한 점수 결과를 더하여 둘 중 우세한 성격요소에 해당하는 알파벳으로 결정된다. 현우는 MBTI 검사를 했다. 그리고 그 결과가 파일 안에 들어있다. 먼저 파일을 열고 읽는다. 그리고 현우의 MBTI 결과를 알아내기 위한 코드를 작성하여라.

MBTI is determined by adding all points of each personality type: Extraversion, Introversion, Sense, Insight, and so on. Suppose that Hyunwoo did MBTI test. In a file, there is a Hyunwoo's MBTI data which will become his MBTI result. First, open a file and read data. Then, design a code that will result in what the MBTI type Hyunwoo has.

Step1 먼저 파일을 열고 읽어온다. 파일에는 'e', 'i', 's', 'n', 't', 'f', 'p', 'j' 라는 MBTI를 구성하는 글자들과 그에 해당하는 점수들이 담겨있다.
First open the file, data of which is consisted of 8 letters: 'e', 'i', 's', 'n', 't', 'f', 'p', 'j' with their scores.

Step2 'e', 'i', 's', 'n' ..와 같은 각 성격 요소들은 모두 딕셔너리에 담겨 key가 되고 그 글자의 점수는 value가 된다. 그리고 각 key에 해당하는 value들의 값을 더하는 방식으로 해당 알파벳의 총합을 구한다.
The personality components like 'e', 'i', 's', 'n' ⋯ will become keys of the dictionary and the scores of each component will become a value of that dictionary. Summing up the values of each key should be done.

Step3 점수의 비교 : 'e'의 점수와 'i'의 점수가 비교되고 그 중 큰 점수의 알파벳이 산출된다. 's'의 점수와 'n'의 점수가 비교되고 그 중 큰 점수의 알파벳이 산출된다. 'f'의 점수와 't'의 점수가 비교되고 그 중 큰 점수의 알파벳이 산출된다. 'p'의 점수와 'j'의 점수가 비교되고 그 중 큰 점수의 알파벳이 산출된다.
The pairs of components are 'e' and 'i', 's' and 'n', 'f' and 't', and 'p' and 'j'. The letter among these two letters with the bigger sum will be selected.

최종적으로 MBTI 알파벳을 리턴하는 함수를 작성하시오.
Return the alphabets of MBTI.

Output Example

```
print(checkMBTI("Q2_testcase.txt"))
# >> isfj
```

문제 3

현우는 테니스 동호회를 운영하고 있다. 매주 새로 들어오는 사람들이 있기에 회원들을 누적 득점 수를 기준으로 줄을 세우고, 4명씩 잘라서 수준별로 팀을 이루게 한다.

최근 알고리즘 공부를 하게 된 현우는 merge sort 알고리즘에 따라 1분에 한 명 씩 이동하여 득점왕이 가장 오른쪽에, 초보자가 가장 왼쪽에 서도록 부탁한다. 하지만 m분 째에 회원들은 줄만 바꾸다 시간이 다 간다며 화를 냈다. 이때 이동하고 있던 사람의 득점 수를 구하라.

Arguments
(1) players: 서로 다른 player의 득점 list. 양의 정수 list.
(2) M: 화를 낸 시점(분). 양의 정수.

Condition
- 만약 M분 이전에 merge sort 완료된다면 -1 출력
- 특정 시점에 이동해야 할 player가 서있는 지점이 이미 올바른 자리라도 이동중으로 간주한다.

Hyunwoo runs a tennis club. There are new people coming in every week, so members line up based on their cumulative score, and four people are cut into teams by level.

He asks one person per minute to move according to the merge sort algorithm, which he recently studied. The player with the highest score stand at the rightmost position and the lowest points stand at the leftmost position. However, in the middle of M minutes, people got angry saying they don't want to waste their time on changing the line. Return the score of the person who was moving at this moment.

Arguments
(1) players: List of players' scores. List of positive integer.
(2) M: the minute people got angry. Positive integer.

Condition
- If sorting is completed before m minutes, return -1.
- Even if the player who is supposed to move at a certain time is standing at the appropriate position already, we still say that he/she is moving.

Example 1:
>>> score ([4,5,1,3,2],7)
3
[Explanation]
4 5 1 3 2 -> 4 5 1 3 2 -> 4 5 1 3 2 -> 1 5 1 3 2 -> 1 4 1 3 2 -> 1 4 5 3 2 -> 1 4 5 2 2 -> 1 4 5 2 3 -> 1 4 5 2 3 -> 1 2 5 2 3 -> 1 2 3 2 3 -> 1 2 3 4 3 -> 1 2 3 4 5.
총 12회 이동이 발생하고 7분에 움직이고 있는 사람의 점수는 3이다.
Total of 12 moves occur, and the person moving in the middle of 7 minutes has score of 3.

Example 2:
>>> score ([4,5,1,3,2],13)
-1
[Explanation]
총 이동횟수는 12이므로 13분 째에는 이동이 일어나지 않는다. 따라서 -1을 반환한다.
Total of 12 moves occur, but it's smaller than 13. Merge sort is already finished when 13 minutes has passed, so return -1.

Output Example

```
$ python Q3_solution.py
players: [4, 5, 1, 3, 2]
M: 7
answer: 3
$ python Q3_solution.py
players: [4, 5, 1, 3, 2]
M: 13
answer: -1
$ python Q3_solution.py
players: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
M: 30
answer: 5
$ python Q3_solution.py
players: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
M: 300
answer: -1
```