[파이썬 트랙] 1회차 월말평가 - Python

시험 과목

✔ Python

시험 목적

✔ Python 프로그래밍 기본 문법에 대한 이해

시험 유의 사항

1) 성실하게 테스트에 임할 것 **(부정 행위시 강력 조치 및 근거가 남음)**

2) 각 문제별로 스켈레톤 코드가 작성된 파일이 제공되며, 해당 코드 파일을 수정하여 정답 코드를 작성할 것 (단, 주어진 함수명은 수정불가)

3) **소스코드 유사도 판단 프로그램 기준 부정행위로 판단될 시,**
0점 처리 및 학사 기준에 의거 조치 실시 예정

4) 테스트 케이스와는 별도로 채점 케이스가 존재함

5) 일부 문제에서 **python 일부 내장 함수(예: min, max, len, sum)** 사용 불가할 수 있음
(문제에 사용 불가 함수는 명시되어 있으며, 명시되지 않은 내장 함수는 자유롭게 사용 가능)

6) 사용자의 입력을 받는 **input 함수는 절대 사용 금지**

7) 잘못된 정답을 반환할 시 오답으로 간주하므로 반드시 문제를 잘 읽어보고 풀 것

시험 코드 작성 유의 사항

최종 제출 코드가 다음 항목에 해당하는 경우, 감점 혹은 0점 처리 될 수 있음

1) Syntax Error로 인한 채점이 불가능한 경우

2) input 함수를 사용하는 경우

3) 주석 설명이 없거나 미흡한 경우

4) 출력 결과에 정답과 무관하거나 불필요한 내용이 있는 경우

5) 문제를 풀지 못하였다면 작성한 코드를 주석 처리하거나 삭제 후 pass 키워드 작성

[파이썬 트랙] 1회차 월말평가 - Python

시험 환경

- Visual studio code(이하 vscode)를 이용한다.
- 그 외 (jupyter notebook, Pycharm, …) 사용 불가

코드 실행

- vscode의 터미널창에서 python 실행 명령어로 결과 확인을 **추천**

```
Edwin@LECTURE MINGW64 /d/SSAFY
$ python problem01.py
```

정답 제출 안내

제출 안내 사항 미 준수 시, 감점 혹은 0점 처리 될 수 있음

1) 압축 및 제출 파일 이름

- 지역_0반_홍길동

  ex) 서울_1반_홍길동 / 부울경_2반_김싸피

2) 압축 폴더 구조

- 시험을 진행했던 폴더 구조 그대로 압축하여 제출 진행
- 우측 구조에서 '서울_1반_김싸피' 폴더 선택 후 압축

```
서울_1반_김싸피/
    problem01.py
    problem02.py
    problem03.py
    ...
    problem11.py
```

제출 마감시간에 서버 요청이 집중될 수 있으므로, 미리 제출하는 것을 권장함

(마감 시간 이후 제출 불가)

[파이썬 트랙] 1회차 월말평가 - Python

문제 01 (problem01.py) – 16점

❖ 킹냥은 일주일 간(7일) 매일 받은 '좋아요' 수를 기록한다.

❖ 예: **[2, 5, 3, 8, 0, 10, 4]**

❖ 리스트의 항목은 **정수**로만 구성되어 있다.

❖ 킹냥은 이 데이터를 가지고 **두 가지**를 알고 싶어 한다.

❖ **[요구사항 1] 평균 '좋아요' 수 (실수)**

❖ 7일간 좋아요 합계를 직접 구해, 7로 나눈 값을 실수(float)로 반환

❖ **[요구사항 2] 가장 많은 좋아요 수와 가장 적은 좋아요 수의 차이 (정수)**

❖ 만약 모든 날짜의 좋아요 수가 동일하다면 0

❖ 그렇지 않다면 (최대값 - 최소값)을 구한다.

❖ 한 주간의 좋아요 수를 저장한 정수 리스트(weekly_like_list)를 전달받아, **일주일의 총 '좋아요' 평균**과 **가장 많은 좋아요 수와 가장 적은 좋아요 수의 차이**를 반환하는 **analyze_likes** 함수를 완성하시오.

❖ **Python 내장함수 sum, len, min, max, sorted, 또는 리스트 sort 메서드 사용시 감점**

[파이썬 트랙] 1회차 월말평가 - Python

문제 02 (problem02.py) – 16점

- 보물 수집가 수집냥은 세상을 돌아다니며 보물 이름이 담긴 리스트를 얻는다.
- 예: **["gold", "silver", "gold", "diamond", "coin", "coin"]**

- 수집냥은 이 보물 정보를 바탕으로 **두 가지** 분석을 수행하고자 한다.

- **[요구사항 1] 보물 개수 딕셔너리 생성**
- 리스트에 담긴 보물을 종류별로 몇 개씩 있는지 세어, {보물이름: 개수} 형태의 딕셔너리를 만든다.
- 만약 리스트가 비어 있다면, 빈 딕셔너리를 결과로 사용한다.

- **[요구사항 2] 임계값(threshold) 이상을 초과하는 보물 종류 수**
- 예를 들어, threshold=2라면, 개수가 2보다 큰 보물 종류가 몇 개인지 센다.
- 만약 threshold를 초과하는 보물 종류가 없다면 0을 반환한다.

- 보물 이름이 담긴 리스트(treasure_list)를 전달받아, 리스트에 보물이 종류별로 몇 개씩 있는지 세고, 특정 임계값을 초과하는 보물 종류가 몇 개인지 세어 반환하는 **analyze_treasures** 함수를 완성하시오.
- **Python 제공 메서드 count 사용 시 감점**

[파이썬 트랙] 1회차 월말평가 - Python

문제 03 (problem03.py) – 16점

- 동물학자 알파냥은 동물원의 여러 종(species)에 대한 개체 수를 {종: 개체수(int)} 형태의 딕셔너리로 관리한다
- 예: **animal_map = { "lion": 5, "tiger": 3, "elephant": 10, }**
- **"lion"은 5마리, "tiger"는 3마리, "elephant"는 10마리가 서식한다는 뜻이다.**
- 알파냥은 현재 동물원에서 **개체 수**가 가장 **많은 동물**이 궁금해졌다.

- **동작 과정 예시 1)**
- find_most_populated({"lion": 5, "tiger": 3, "elephant": 10})
- 가장 개체 수가 많은 동물: "elephant" (10마리)
- 결과: "elephant"

- **동작 과정 예시 2)**
- find_most_populated({})
- 빈 딕셔너리 → 동물 정보가 없으므로 None
- 결과: None

- 동물 종과 개체 수를 담은 딕셔너리 **animal_map**을 단일 인자로 전달받아, 최대 개체 수를 지닌 동물 종 이름을 반환하고, 비어 있거나 동물 정보가 없으면 **None**을 반환하는**find_most_populated** 함수를 완성하시오.
- **Python 제공 메서드 count 사용 시 감점**

[파이썬 트랙] 1회차 월말평가 - Python

문제 04 (problem04.py) – 12점

❖ 탐험가 섬냥은 섬 곳곳을 돌아다니며 다양한 아이템(정수, 문자 등)을 하나의 리스트에 모은다.

❖ 주어지는 리스트는 빈 리스트, 숫자만 있는 리스트, 문자만 있는 리스트, 숫자와 문자가 혼합된 리스트로 구성되어 있다.

❖ 예: **[2, 2, 2, "a", "a", 3, 0, -2]**

❖ 섬냥은 이 아이템 정보를 바탕으로 **두 가지** 분석을 수행하고자 한다.

❖ **[요구사항 1] 중복 제거**

❖ 리스트에 담긴 아이템들을 중복 없이 정리하고자 한다.

❖ 만약 리스트가 비어 있다면, 빈 리스트를 결과로 반환한다.

❖ **[요구사항 2] 양수와 음수의 합**

❖ 중복 제거 결과 리스트 내에서 정수만 골라 양수와 음수를 구분하여 합산한다. **(0은 제외)**

❖ 최종적으로는 (중복 제거가 끝난 리스트, (양수합, 음수합)) 형태의 튜플을 반환한다.

❖ 예: ([중복 제거된 아이템], (양수합, 음수합))

❖ 아이템이 담긴 리스트(items_list)를 전달받아, 중복 제거 리스트를 구하고, 그 리스트 내 정수들을 대상으로 양수합과 음수합을 반환하는 analyze_items 함수를 완성하시오.

❖ **Python 내장 함수 set, sum 사용시 감점**

[파이썬 트랙] 1회차 월말평가 - Python

문제 05 (problem05.py) – 12점

- 도도새는 자기 전에 매일 일기를 쓰는 습관이 있다.
- 최근 도도새는 기분이 좋았던 날에만 "happy"라는 단어를 자주 사용한다는 사실을 알아차렸다.
- 도도새는 자신의 일기 내용을 바탕으로 "happy"라는 단어가 몇 번 등장했는지 확인하고 싶어 한다.
- **대소문자는 구분하며, 오직 정확히 'happy'와 일치하는 부분 문자열만 센다.**
- 공백, 문장 부호 등은 따로 처리할 필요가 없다.
- 부분적으로 겹치는 경우(예: "happyhappy")는 문자열 순서대로 각각 카운트한다.

- **동작 과정 예시 1)**
- count_happy("I feel happy. HAPPY! so happy!")
- # 대소문자 구분 -> "happy" 1회, "HAPPY" 무시
- # "so happy!" -> 1회
- # 결과: 2

- **동작 과정 예시 1)**
- count_happy("happyhappy")
- # "happyhappy" -> "happy" + "happy"
- # 겹치는 부분이 없으므로 2회

- 주어진 **문자열(일기 내용)을 인자로 받아, 문자열에 등장하는 "happy" 단어의 총 횟수를 반환**하는 **count_happy 함수**를 완성하시오.

- **Python 제공 메서드 count 사용 시 감점**

[파이썬 트랙] 1회차 월말평가 - Python

문제 06 (problem06.py) – 12점

- 크기가 N인 2차원 정사각 행렬이 주어졌을 때, **가장 큰 값을 가지는 행, 열 좌표를 반환**하는 **find_max_position** 함수를 완성하시오.
- 행렬의 크기 N은 2 이상 10 이하이다.
- 가장 큰 값이 여러 개 있을 경우 행의 인덱스가 가장 작은 좌표를 리스트로 반환한다.
- 가장 큰 값이 여러 개가 같은 행에 위치한 경우에는 열의 인덱스가 가장 작은 좌표를 리스트로 반환한다.

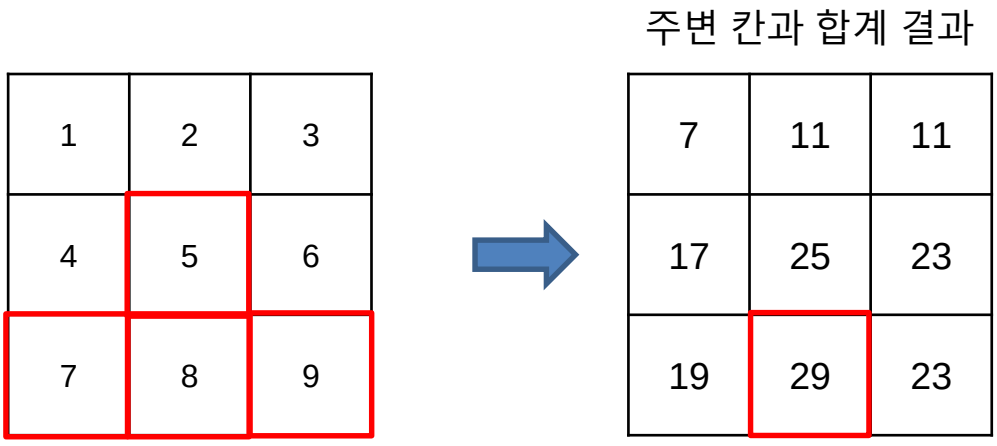
문제 07 (problem07.py) – 8점

- 주어진 **문자열을 뒤집는 재귀 함수**를 완성하시오.
- 함수 **reverse_string**은 문자열을 인자로 받아, 그 문자열을 뒤집은 결과를 반환한다.
- 반드시 재귀 함수 구조로 작성해야 한다.
- **반드시 재귀 함수를 이용해야 하며 재귀함수가 아니면 감점**

[파이썬 트랙] 1회차 월말평가 - Python

문제 08 (problem08.py) – 8점

- 크기가 N인 2차원 정사각 행렬이 주어졌을 때, 현재 위치를 포함한 각 칸의 위, 아래, 좌, 우 방향의 칸의 모든 합을 구한다.
- 이렇게 모든 칸을 순회하여 **주변 칸과의 합을 계산했을 때 합이 가장 큰 값을 반환**하는 **max_adjacent_sum** 함수를 완성하시오.
- 이때 주변 영역이 행열의 범위를 벗어나면 해당 방향은 제외하고 남은 칸의 값을 더한다.
- 행렬의 크기 N은 2 이상 10 이하이다.



붉은 색 : (2, 1) 좌표일 때 합계 예시