Writing a Hiring Algorithm

Scenario: Moogle's Hiring Filter

Imagine you are working for *Moogle*, a well-known tech company that receives tens of thousands of job applications from graduating seniors every year.

Since the company receives too many job applications for HR to individually assess in a reasonable amount of time, you are asked to create a program that algorithmically analyzes applications and selects the ones most worth passing onto HR.

Applicant Data

It's difficult to create these first-pass cuts, so *Moogle* designs their application forms to get some numerical data about their applicants' education. Job applications must enter the grades they received in 6 core CS courses, as well as their overall GPA. For your convenience, this will be stored in a python `list` that you can access. For example, a student who received the following scores…

- **Intro to CS:** 100
- **Data Structures:** 95
- **Software Engineering:** 80
- **Algorithms:** 89
- **Computer Organization:** 91
- **Operative Systems:** 75
- **Overall GPA:** 83

… would result in the following list: `[100, 95, 80, 89, 91, 75, 83]`. You can assume that index `0` is *always* Intro to CS, `1` is *always* Data Structures, and so on.

Because you are processing many applications, your program will receive a *list of lists*. For example, this would be the information for 3 applicants:

```
[ [100, 95, 80, 89, 91, 75, 83], [75, 80, 85, 90, 85, 88, 90],
[85, 70, 99, 100, 81, 82, 91] ]
```

Your Task

Your job is to:

1. Determine how you are going to select the top applicants to pass onto HR.
2. Given a list of applicant data (a *list of lists*), write a function returns a new list of worthwhile candidates.

Your Code

Complete the following methods:

- `analyze_applicant1` accepts applicants that have an overall GPA above 80. (Does *not* need a for loop)
- `analyze_applicant2` accepts applicants that have no grade below 65.
- `analyze_applicant3` accepts applicants that have at least 4 grades above 85.
- `analyze_applicant4` accepts applicants that have an average above 85.

After writing, testing, and considering the tradeoffs of these four methods, write your own criteria in `your_analysis`.

Questions you should answer:

1. What criteria did you choose to select finalists? How did you choose that criteria?
2. Roughly what percentage of applicants does your algorithm pass on as finalists? Is that enough? If *Moogle* asked you to take a more aggressive approach with your algorithm, are there any tradeoffs?