Project 1: Ethical Automation
COSC423/523, Fall 2021

Due: Sept 1, 11:59pm

1 Overview and Instructions

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 the Human Resources Department (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. Your job is to build an algorithm that helps determine the order which job applicants will be called in for an interview. This is a real scenario playing out every day and will likely impact you as an applicant someday.

2 Applicant Data

To make it easier for their algorithms to process, *Moogle* designs their application forms to get some numerical data about their applicants education. Applicants must enter the grades they receive from their 5 core CS courses, as well as their overall non-CS GPA at the university. 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
- **Algorithms:** 89
- **Computer Organization:** 91
- **Operating Systems:** 75
- **Non-CS GPA:** 83

... would result in the following list:

```
[100, 95, 89, 91, 75, 83]
```

3 Implementation Requirements

This project requires that you write a series of predefined `analyze_applicant` methods which apply different criteria to applicants. The implementation requirements for this project are as follows:

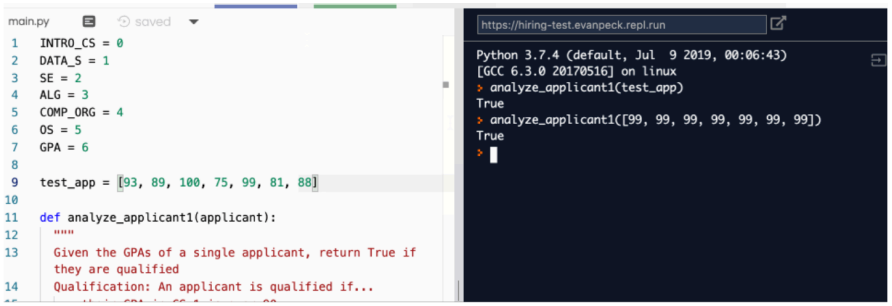

Figure 1: Examples of `analyze_applicant1`'s usage.

1. You must implement four different functions that accepts applicants (i.e., returns True) if a particular condition is met:

   (a) **`analyze_applicant1`:** Accepts applicants that have an average above 85.

   (b) **`analyze_applicant2`:** Accepts applicants that have no grade below 65. This includes all grades (including their non-CS GPA).

   (c) **`analyze_applicant3`:** Accepts applicants that have at least 4 grades above 85 (their non-CS GPA counts as a grade in this case)

   (d) **`analyze_applicant4`:** Accepts applicants that have an average above 85 in their 5 CS courses. **(You cannot use list slicing.)**

2. Your implementation must be in Python. For this assignment, you CANNOT use pythons built-in functions like `sum()`, `min()`, `max()` or `count()`. Python libraries for additional data structures (e.g., NumPy) are not permitted.

3. All source code should be implemented in a file named `main.py`. This file should read in a datafile of applicant records, determine if an applicant meets all four criterias for being accepted, and produce an output file named `results.csv`.

   (a) **applicants.csv:** A datafile of applicant records where each row represents one applicant's data. The file is available on Canvas.

   (b) **results.csv:** A file containing the program's results where each row represents the program's output based on the supplied input file. The output should for accepted applicants should be "ACCEPT" while the output for unaccepted applicants should be "REJECT".

4. Your code should be adequately documented. Each function should have a supporting docstring followed the PEP 257 convention. The top of `main.py` should include your name as an author and provide a brief description of your source code.

5. You should include a "README" file that provides an overview of your code. You should provide instructions for running your code locally.

   (a) **Ethical Statement:** Your README must include a statement on the ethical nature of the code that you've written. Should this software be permissible for real-world use? Why or why not? Your statement can be up to four sentences in length.

```python
def analyze_applicant1(applicant):
  """
  Given the GPAs of a single applicant, return True if they are qualified
  Qualification: An applicant is qualified if...
    - The average of all their grades is above 85

  @param applicant: a list of GPAs (integers)
  @return True if the applicant qualifies
  """

  return True # Simulate an assessment

# define a list of dummy scores
applicant = [1, 2, 3, 4, 5, 6]

# call the function
if analyze_applicant1(applicant):
    print("They passed! :)")
else:
    print("They didn't pass! :(")
```

4 Starter Code

Above, you will find a sample of starter code given to you that clearly (1) defines a function, (2) demonstrates the format of a Python function with a PEP-257 compliant docstring, and (3) calls the defined function.

5 Grading and Submission

The assignment is worth 100 points. Create a ZIP file that includes all of your Project 1 files. Upload the ZIP file to the Project 1 submission folder on Canvas by the due date. For questions regarding the late policy for assignment submission, please consult the syllabus.

5.1 Grading Scheme

The following grading scheme will be used for undergraduate students:

Requirement	**Point Value**
README follows specification	10 points
Ethical Statement follows specification	10 points
Code documentation follows specification	10 points
Specification of `analyze_applicant` is followed	40 points
Specification of `main.py` is followed	30 points
Total	100