**Objective**

Create a Python program (autograder.py) that grades Python files based on **structure, comments, function usage, variable usage, and program output**. Your grader should produce a **rubric table with scores and feedback**.

**You are NOT allowed to use unittest, doctest, or assert statements on this assignment.**

**Programs to Grade:**

* **Program 1:** find *program1.zip* linked on assignment instructions on canvas. The zipped folder contains the solution file plus correct and incorrect submissions , examples shown below:
  + program1\_solution.py
  + program1\_stu\_incorrect1.py
  + program1\_stu\_incorrect2.py
  + program1\_stu\_incorrect3.py
  + program1\_stu\_correct.py
* **Program 2:** find *program2.zip* linked on assignment instructions on canvas. The zipped folder contains the solution file plus correct and incorrect submissions , examples shown below:
  + Program2\_solution.py
  + Program2\_stu\_incorrect1.py
  + Program2\_stu\_incorrect2.py
  + Program2\_stu\_incorrect3.py
  + Program2\_stu\_correct.py

Below are instructions that will help with the first program (*program1.py*)

**Step-by-Step Instructions**

**Step 1: Use the starter template**

* You are provided with autograder\_template.py starter template.
* Your task is to **implement each TODO section**:
  1. check\_loops()
  2. check\_comments()
  3. check\_variables()
  4. compare\_outputs()

**Step 2: Run student programs**

* Use subprocess to run Python files (for example program1\_stu\_incorrect1.py).
* Capture stdout, stderr, and exit code.
* Strip leading/trailing whitespace for output comparison.

**Step 3: Check code structure**

* Use ast to analyze the code.
* Validate loops:
  + Reject while True or while <variable>
  + Accept boolean expressions like while x < 10
* Detect function definitions.

**Step 4: Check comments**

* Ensure comments exist in key sections.
* Provide feedback if comments are missing or insufficient.

**Step 5: Check variable usage**

* Detect variables in loops and functions.
* Encourage readable and descriptive variable names.

**Step 6: Compare outputs**

* Run program1\_solution.py against one of the program1\_stu files
* Compare outputs, ignoring extra whitespace
* Provide detailed feedback when output differs

**Step 7: Generate rubric table**

* Each criterion should show:
  + Pass/Fail
  + Weight
  + Points earned
  + Feedback messages
* Use the grade\_student() function in the template to combine scores.

**Step 8: Calculate final grade**

* Apply weights defined in CRITERIA dictionary.
* Sum the points from all criteria for the final score.

**Step 9: Testing**

* Test your grader on the other program1\_stu student files.
* Ensure scoring, feedback, and rubric table are correct.

**Step 10: Make the program MENU DRIVEN**

Remember the program is to grade both program 1 and program2. You’ll therefore want to make it menu-driven so that the instructor can choose which assignment to grade.

The program is to display the following menu once it’s first executed.

-------------Menu----------------  
1) Grade program1.py submission  
2) Grade program2.py submission  
3) Exit

**Make sure you have a proper decision structure and loop**

**Step 11: Enhance the autograder**

* **If option 1 is chosen**. The program is to run the completed in steps 1 through 9 above.
* If option 2 is chosen. Enhance the program so that it grades program2.py submission based on following criteria.
  + Assignment instructions for program2 is linked in M2Pro2 instruction post on canvas.
  + Grading Criteria for this assignment is below  
      
    Criteria :   
    CRITERIA = {

"structure(loops and decision structures)": 20, # % weight for loop/structure

"output ": 50, # % weight for correct output

* + - Option 1 : 10 points
    - Option 2: 5 points
    - Option 3: 10 points (**evaluate use of String methods**)
    - Option 4: 10 points (**evaluate use of String methods**)
    - Option 5: 10 points (**evaluate use of String methods**)
    - Option 6: exit() not used , program properly terminates ( 5 points)

"functions": 10, # % weight for using functions

"pseudocode": 5, # % weight for comments

"variables": 5 # % weight for variable usage

}

**Step 12 Submission**

* Submit:
  + autograder.py (with your implementations filled in)
  + program1.zip folder
  + program2.zip folder