General information

For each question, paste your solution in the text field under the description.

For programming questions:

- All answers should be provided as working Python code
- You should perform your own tests to verify that the code is correct. However test cases need not be provided with your answer. If you do provide test cases, you should identify those clearly (for example with comments).
- For maximum credits, the provided answer should be in the best possible complexity class. Less efficient answers will be given partial credits.
- It is not necessary to comment on your code, it is more important that the code is clean and easy to read. Comments are good to document assumptions that you make, if any.

The maximum possible total number of points is either 60 or 35. For 60-point exam: 24 points correspond to grade 3, 36 points correspond to grade 4, and 48 points correspond to grade 5. For 35-point exam: 14 points correspond to grade 3, 21 points correspond to grade 4, and 28 points correspond to grade 5.

Allowed aids:

- You can refer to your 1-page of handwritten notes.
- You can use the Self-Practice website in sandbox mode to test your code. From the Safe Exam Browser, you should click the "Python Sandbox" button on the bottom left of your screen to open it. This will open the self-practice website that we used in the course in another window. Note that your code won't be saved if you switch exercise in the self-practice website, or if you close or reload its window. Therefore it is advised to have one Self-Practice window open for each question and switch between windows using Alt+Tab or Option+Tab. Every time you click on "Python Sandbox", a new window will appear.
- You can refer to the "python reference" link at the top of the Self-Practice. Again, it is advised to open it in a separate window.
- Questions are provided in Swedish for convenience, but in case of discrepancy the English version applies. You can use the (so-called "hamburger") menu to choose the language of questions.

Disallowed aids: anything else. In particular, it is not allowed to use the following electronic tools.

- AI tools (Copilot, ChatGPT, Llama, ...)
- Communication tools (phone, chat, mail, social media, ...)
- Electronic documentation (Google, stack overflow, python guides or manuals, ...)
- Books
- Notes in printed or electronic format

Q1: Elo calculator

In a lot of games, such as Chess, players are assigned an Elo rating. A usual Elo rating is between 1000 and 2000.

If the player playing the white pieces has rating A and the player having the black pieces has rating B, the expected score of a matchup is given by the following formula:

$$\frac{1}{10^{\frac{B-A}{400}}+1}$$

A score of 1 corresponds to certainty for A to win the matchup, and a score of 0 corresponds to B winning. Write a function `guess_winner(white_rating, black_rating)` which prints one of the following messages:

- "White is expected to win"
- "Black is expected to win"
- "A draw is expected"

Print the last message if the expected score is between 0.45 and 0.55, and either of the other messages accordingly to the prediction.

Q2: Molecular Mass

Write a function `m_mass(molecule)` which computes an approximation of the molecular mass of `molecule`, as follows.

The molecule is provided as a string, with a series of atom names possibly followed by a digit. For example: "H2O", "CH4", "H2SO3", etc.

The mass of the molecule should be computed by an appropriate weighted sum of the given atoms. For instance, the estimated weight of H2O is 2 times the weight of atom H plus the weight of atom O.

You should assume a file `atoms.txt` which contains the atomic weight data. The file has one atom data per line. On each line, the atom letter occurs first, then a space, then its weight.

- You can assume that atoms are identified by a single letter. For instance, LiOH is not a valid input.
- You can assume that each atom does not occur more than 9 than times in a molecule. For instance, C10H22 is not a valid input.
- You can assume that data for every atom in the molecule is present in `atoms.txt` .
- Within the **Exam Sandbox, Question 2**, you can open an example `atoms.txt` file.

Q3: Player Pool Manager

Write a class `EloPool` which tracks the Elo rating of a pool of players. It should have the following methods:

- `__init__(self)`
  This function initialises an empty pool of player. For each attribute initialised here, add one line of comment explaining its role.
- `register_player(self,player_name)`
  This function adds a player to the pool. Every player starts as an active player with a rating of 1000 Elo. If the player is already registered, an error message should be printed instead of registering the player. A retired (non-active) player cannot re-enter the pool of active players. Print an error message if this is attempted.
- `match_players(self,player1_name,player2_name,result)` This function should update the rating of the players when a match between them is played.
  `result` is 1 if player1 wins, 0 if player2 wins, and 0.5 if it is a draw.
  Let *e* be the expected score for the match (see Q1).

Then, both players ratings should be adjusted. One of them is increased or decreased by *K×(result-e)* and the other by *K×(e-result)*. To determine which is which, consider that if there is a winner, then the score of the winner will be increased and the loser score will be decreased. In the above formula, let `K=40` .

If either of the players are not registered active players, an error message should be printed.

Note: you should copy your implementation of the formula for *e* from Q1 in your answer to this question. If you could not implement the formula for *e* in Q1, you can let *e* be 0.8.

- `retire_player(self,player_name)`
  This method removes a player from the pool of active players. The player's Elo points (rating) should be returned to the pool of players. When doing so you must *maintain the invariant that the sum of Elo ratings in the pool is equal to 1000 times the number of active players*. The points should be distributed evenly among all the remaining active players. If the player isn't registered, or was already retired, print an error message instead.
- `print_players(self)`
  This method should print a table of all players who have ever entered the pool. For each player, print the following information:
  i. the player's name
  ii. their current rating or their rating at time of retirement.
  The table should be sorted by player name.