University of Roehampton London

LAST UPDATED: 16 FEBRUARY 2023 BY ARTURO ARAUJO

SOFTWARE DEVELOPMENT 1 COURSEWORK SPECIFICATION

This course starts your software deployment journey by introducing the key programming concepts on which the rest of the course is built. The focus here is on a solid foundational knowledge of Python's syntax and semantics. To evaluate progress and highlight any gaps in understanding, this coursework will provide a set of challenging tasks and programming requirements, as well as ample time to reach an adequate solution to them.

Coursework 1 Submission date: 17 March 2023 via CodeGrade in Moodle by 16:59 hours GMT.

Coursework 2 Submission date: 21 April 2023 via CodeGrade in Moodle by 16:59 hours GMT.

Contribution: Each coursework contributes with 30% of the total coursework mark.

Aim: To gauge understanding on the correct use of variables and objects, data types and the ability to abstract data; as well as the develop students' fluency in programming languages and software development.

Learning Outcomes: 1: Design, implement, test, and debug a program that uses each of the following fundamental programming constructs – basic computation, simple I/O, standard conditional and iterative structures, the definition of functions, and parameter passing. 2: Analyse and explain the behaviour of simple programs involving the fundamental programming constructs – variables, expressions, assignments, I/O, control constructs, functions, parameter passing, and recursion. 3: Identify the relative strengths and weaknesses among multiple designs or implementations for a problem. 4: Use a programming language to implement, test, and debug algorithms for solving simple problems.

Grading Advice: While fully functioning programs and its documentation that meet all the requirements are the goal of the test, the programs and the rationale for their design, implementation, testing and debugging will be assessed for efficiency, simplicity, creativity and good style. Please read the Grading Rubric and bring any questions you may have to your tutor.

COURSEWORK 1

To complete this coursework, you will design the behaviour of simple programs involving the fundamental programming constructs – variables, expressions, assignments, I/O, and control constructs. **Time in class will be provided for coursework feedback.** Advice will be given in class on how to improve your work. Final mark and rubric feedback will be discussed on a short 1-1 demo after submission. ***If demos are not presented either in person or via Teams, you risk losing this part of the grade.***

A prestigious football club has hired our Software Development company. As we know, football's players transfer from one club to another every year. There are several aspects to estimating their salary, such as their skills, performance in the previous seasons, age, improvement, personality etc. The manager of the club wishes to estimate the salary of new players by rating their skills. You have been tasked to create a program that automates this calculation.

There are six main criteria to rate the skills of the player:

1- Speed
2- Shooting
3- Passing
4- Defending
5- Dribbling
6- Physicality

A score between 0 and 5 is assigned to each of these criteria, and these are used to help calculate an overall rating out of 100.

The formula to calculate the Overall Rate is:

*Overall Rate = Summation of skills rate * 100 /30*

Thus, a salary dataset example is as follows:

No	Salary	Overall rate
1	1000	80
2	700	60
3	500	45
4	400	30

Example, if you rate the skills of player as: Speed = 3, Shooting = 3, Passing = 5, Defending = 3, Dribbling = 2, and physicality = 2. Then, the overall rate should be **60.**

From the dataset above, If the overall rate is greater than or equal to 80, the output should be **1000**.

If the overall rate is between 80 and 60, the output should be **1000 700**.

If the overall rate is equal to 60 should be **700**.

If the overall rate between 60 and 45, the output should be **700 500**.

If the overall rate is equal to 45 should be **500**.

If the overall rate between 45 and 30, the output should be **500 400**.

If the overall rate less than or equal to 30, the salary output be **400**.

You can find out more about player's ratings and estimating salary at:

https://www.fourfourtwo.com/features/fifa-22-player-ratings-explained-chemistry-fut-ultimate-team-pace-passing-speed-card

https://www.researchgate.net/publication/318655683_Computational_Estimation_of_Football_Player_Wages

http://eightyfivepoints.blogspot.com/2019/05/from-sessegnon-to-sanchez-how-to.html

https://www.capology.com/uk/premier-league/salaries/

There are three tasks you need to complete for this coursework:

Task 1. Write the pseudocode OR draw a flowchart that you will use for your program design (the detailed program description is specified in task 2). You can write it down/draw it by hand and take a picture, or choose any software you prefer, e.g., Microsoft Word for pseudocode, draw.io for flowchart.

Task 2. Following the pseudocode or flowchart you designed, craft a program in python which has the features listed below:

a. You program should use the *input()* function that includes message text to instruct the user to enter each player's data.

b. Use an appropriate data structure to create and store this dataset, which contains the players' salaries, and overall rates.
c. Your program calculates the overall rate for each player, using the six main criteria described above (speed, shooting, passing, defending, dribbling and physicality)
d. Your program should estimate the suitable salary for each player by using the salary dataset provided. For instance, if the overall rate of the player is between 60 and 45, the output should be:

    700 500

e. The computed salary value or values should be printed out as an integer (no decimals).

Task 3: Write a brief reflection on your code (~500 words), which may include but not limited to the discussions about topics below.

- How did you decompose the problem and why?
- Do you think flowchart/pseudocode helpful when you write a program and why?
- Have you met any difficulties/bugs during this coursework, and how did you tackle it?
- What did you think is the most difficult part in this coursework?
- What have you learnt from investigating this problem?
- Which part(s) in your code design is your favourite and why?
- If you had no time limit, how would you improve your program?

Resources on reflection can be found at: https://libguides.hull.ac.uk/reflectivewriting/reflection1a

Submission Requirements: Please submit your code as a **python file .py (instructions will be provided during lab sessions) or directly in codegrade's editor.** You must submit a python file (*.py) containing the code for your program (Task 2) as well as a pdf file containing your pseudocode/flowchart and personal reflection on your code (Task 1 and Task 3) to CodeGrade on Moodle. Your program should contain full documentation explaining your implementation (comment your code). **Both files should have your name, ID number and date of last update.**

Feedback: Your code will be automatically graded by CodeGrade. Time during lab/seminar will be arranged for you to demo your work, obtain a grade for the non-code submissions, and get personalized feedback from your tutor and advise on how you can improve your work. ***If demos are not presented either in person or via Teams, you risk losing this part of the grade.***

Marking Scheme: A short a video explaining the assessment criteria can be found at: https://www.youtube.com/watch?v=JrA0Dotq1p8

The following rubric will be used to assess your work:

Criteria	Excellent	Satisfactory	Not Satisfactory	Not Attempted
Solution the completeness of the code, documentation and reflection to meet the specification given.	A completed solution meeting all the specifications.	A completed solution that does not quite meet all the specifications.	A completed solution is implemented and runs but lacks much of the specification.	Solution doesn't run or does not meet the specifications defined.

Design ability to decompose a problem into coherent and reusable parts.	The solution design decomposes the problem by using appropriate structures. The overall design is appropriate, and with a consideration of reusability.	The solution design generally uses appropriate structures. Program elements and documentation both exhibit good design.	Not all of the selected structures are appropriate. Some of the elements are appropriately designed.	Few if any of the selected structures are appropriate, or the problem is decomposed in ways that make little sense.
Correctness ability to create solution that reliably produces correct answers or appropriate results.	Solution produces correct answers or appropriate results for all inputs tested.	Solution produces correct answers or appropriate results for most inputs.	Solution approaches correct answers or appropriate results for most inputs but can contain miscalculations.	Solution does not produce correct answers or appropriate results for most inputs.
Logic ability to use correct program structures appropriate to the problem domain.	Solution logic is correct with no known boundary errors, and no redundant or contradictory conditions.	Solution logic is mostly correct but may contain an occasional boundary error or redundant or contradictory condition.	Solution logic is on the right track but shows no recognition of boundary conditions (such as < vs. <=).	Solution contains some conditions that specify the opposite of what is required (less than vs. greater than), confuse Boolean AND/OR operators, or lead to infinite loops.
Clarity ability to format and document code for human consumption (Good Style) and reflection.	Program contains appropriate documentation for all major functions, variables, or non-trivial algorithms. Formatting, indentation, and other white space aids readability.	Program contains some documentation on major functions, variables, or non-trivial algorithms. Indentation and other formatting are appropriate.	Program contains some documentation (at least the student's name and program's purpose) but has occasionally misleading indentation.	Program contains no documentation, or grossly misleading indentation.
Robustness ability of the solution to handle unexpected input and error conditions correctly as evidenced via testing.	Solution handles erroneous or unexpected input gracefully; action is taken without surprising the user. Boundary cases are considered and tested.	All obvious error conditions are checked for and appropriate action is taken. Nearly all boundary cases are considered and tested.	Some obvious error conditions are checked for and some sort of action is taken. Most boundary cases are considered and tested.	Solution often fails or fails completely. Boundary conditions are not tested for.

COURSEWORK 2

To complete this coursework, you will design the behaviour of complex programs involving the fundamental programming constructs – functions, operations, and file reading/writing. A template Jupiter notebook and a text file are provided to complete the tasks. Time in class can be used to troubleshoot and get feedback on your program.

The prestigious football club who originally hired our Software Development company is very pleased with the outcome and has requested an extension. The program that you have crafted for coursework 1 takes in rates of 6 skills of new players in the form of a score between 0 and 5 (shouldn't take inputs outside of this range). The six main criteria to rate the skills of the player should be:

1. Speed
2. Shooting
3. Passing
4. Defending
5. Dribbling
6. Physicality

The program uses these individual skill rates to calculate an overall rating out of 100 using the formula:

*overall rate = summation of skills rate * 100 / 30*

The resulting number (should be between 0 and 100) is then used to estimate the salary using the salary dataset:

No	Salary	Overall rate
1	1000	80
2	700	60
3	500	45
4	400	30

From the dataset above,

- If the overall rate is greater than or equal 80, the salary should be: 1000
- If the overall rate less than or equal to 30, the salary should be: 400
- If the overall rate is in between two numbers, like 45 and 30, the output should be a range of the maximum salary and the minimum salary: 500 400

You are asked to add the following functionality (up to 78 points, depending on scoring an "Excellent" score in each rubric component):

1. Functions (8 points): Split and pack up your coursework 1 code into functions. The program must contain a main() function and at least two other functions called "calculate_rating()" and "calculate_salary()".
   Use the main() function to call the other functions to receive the 6 player's skill ratings, calculate the player's score and their salary range.

2. Input expansion (8 points): Adjust your program to ask users to enter a player ID (2-digit number), name and date-of-birth (D.o.B) by sequence, before asking for the 6 skills from coursework 1 (speed, shooting, passing, defending, dribbling, physicality). D.o.B should be in ISO format YEAR-MONTH-DATE.

3. Loop your code (8 points): Your program should use a loop to ask for the information of three players, or until the input for user ID is "end".

4. Calculate the players' overall ratings and store it for later use (2 points).

5. Calculate the players' salary ranges and store it for later use (2 points).

6. Calculate players' ages (8 points): According to each user's year of birth, your program should calculate the age of the player and store it for later use. Hint: the library datetime can be used here to make your job easier.

7. **Tabulate (8 points):** Display a table that summarises the input data (ascending order with user ID) using the tabulate library. It should look like:

```
  UID  Name               D.o.B         Age    Score  Salary Range
-----  -----------------  ----------  -----  -------  --------------
   13  Lieke Martens      1992-12-16     29  86.6667  1000
   02  Cristiano Ronaldo  1985-02-05     37  73.3333  1000 700
   24  Lucy Bronze        1991-10-28     31  50       700 500
```

8. **File writing (8 points):** Save the table into a new local file named "players.txt"
9. **Make sure that you:**
    a. **Implement robust input validation for a player's ID, D.o.B., and ratings.** Your program should give a warning that says ***"The rating you entered was invalid"*** for any input errors **(6 points)**
    b. **Display use of appropriate data structures for storing player information (5 points)**
    c. **Display use of appropriate code structures for implementing the functionality (5 points)**

10. **Distinction Advanced Function:** Once everything in requirements 1-9 has been completed, for a distinction grade, extend your program with one more function called advanced(). This function should:
    a. be a complimentary main() method that, instead of taking user input, uses the provided file as input.
    b. accept in its argument (filename) the name of the text file to read.
    c. read the user information record from the provided file "PlayerData.txt", which contains player IDs, names, and date-of-births. The function should then use this data as the input for the program.
    d. call all the other functions used and produces a text file output (requirement 8).

Submission Requirements: Submit your code via Codegrade as a Jupyter notebook named:

footballer_skills.ipynb

The program file should contain full documentation explaining your implementation (comment your code). Your file should have your name, ID number and date of last update.

Feedback: Feedback will be given automatically through Codegrade via Moodle.

Marking Scheme: Grades will be based on number of components completed. You will be assessed on

- **Solution:** the completeness of the code, documentation, and reflection to meet the specification given.
- **Design:** ability to decompose a problem into coherent and reusable parts.
- **Correctness:** ability to create solution that reliably produces correct answers or appropriate results.
- **Logic:** ability to use correct program structures appropriate to the problem domain.
- **Clarity:** ability to format and document code for human consumption (PEP-8) and reflection.
- **Robustness:** ability of the solution to handle unexpected input and error conditions correctly as evidenced via testing.

The following rubric will be used to assess your work:

Basic Requirements

In order to get full marks for this rubric category your code must:

- Be encapsulated into functions including `*main()*`, `*calculate_rating()*`, and `*calculate_salary()*`. (8)
- Prompt users for an ID, name, D.o.B, and 6 player ratings, in that order. (8)
- Accept data for exactly 3 players before stopping or if "end" was entered instead of an ID. (8)
- Calculate the players' overall ratings. (2)
- Calculate the players' salary ranges. (2)
- Calculate the players' ages. (8)
- Display a table of all the data passed to the program using `tabulate` (8)
- Save this table to `players.txt` (8)

Intermediate requirements

In order to get full marks for the rubric category your code must:

- implement robust input validation for a player's ID, D.o.B., and ratings (6)
- Display use of appropriate data structures for storing player information (5)
- Display use of appropriate code structures for implementing the functionality (5)

Distinction Requirements:

Distinction work in the range of 72-78 should demonstrate:

- exemplary attainment of learning outcomes
- a high level of insight and critical evaluation of the material
- a comprehensive and up-to-date account of relevant theoretical and empirical material
- a thorough understanding and integration of material supporting a cogent argument
- excellent writing of a high academic standard

Distinction work in the range of 82-100 should demonstrate:

- attainment beyond the intended learning outcomes
- an outstanding level of originality and creativity, providing a significant new perspective on the question or topic
- a clear, elegant and well supported argument, based on the integration and sophisticated critical evaluation of a substantial body of knowledge
- suitability for publication in high quality journal

Academic Misconduct: All submissions will be processed through a code plagiarism tool. If signs of misconduct are found, all students involved will be contacted to discuss further steps. Please see here for information on academic integrity at the university https://portal.roehampton.ac.uk/information/Pages/Academic-Integrity.aspx.

Our guiding principle is that academic integrity and honesty are fundamental to the academic work you produce at the University of Roehampton. You are expected to complete coursework which is your own and which is referenced appropriately. The university has in place measures to detect academic dishonesty in all its forms. If you are found to be cheating or attempting to gain an unfair advantage over other students in any way, this is considered academic misconduct and you will be penalised accordingly. Please don't do it.