Cardiff School of Computer Science and Informatics

Module Code:	CM1103
Module Title:	Problem Solving with Python
Lecturer:	Stuart Allen
Assessment Title:	Problem solving exercise
Assessment Number:	1 of 2
Assessment Weighting	40% of a 20-credit level 4 module
Assessment Limits	Python code for Questions 1 and 2 as necessary. Report for Question 2 of at most 750 words.

The Assessment Calendar can be found under 'Assessment & Feedback' in the COMSC-ORG-SCHOOL organisation on Learning Central. This is the single point of truth for (a) the hand out date and time, (b) the hand in date and time, and (c) the feedback return date for all assessments.

Learning Outcomes

The learning outcomes for this assessment are as follows:

- Use Python and common modules to implement simple algorithms expressed in pseudocode, and understand fundamental programming concepts
- Develop informal algorithms and apply recursion to solve simple problems
- Write scientific reports describing the analysis of a problem

---

Submission Instructions

Your submission should consist of multiple files:

Description	Type	Name
Cover sheet ***Compulsory:*** *Code for Q1, Q2 and report*	One pdf file One Jupyter notebook file	`[Student number].pdf` `[Student number].ipynb`

The template files for this coursework can be accessed by accepting this invitation to a GitHub classroom:

`https://classroom.github.com/a/YJ_U7vmY`

(or by downloading individual files from the assignment on Learning Central)

The coversheet is included in the GitHub Classroom repository (or can be found under 'Assessment & Feedback' in the COMSC-ORG-SCHOOL organisation on Learning Central).

All files should be submitted via committing and pushing to your GitHub classroom repository. (Alternatively submissions can be made via the assignment page The submission page can be found under 'Assessment & Feedback' in the CM1103 module on Learning Central).

Any code submitted will be run using the CodeSpaces environment included in the GitHub classroom on a system equivalent to those available in the Windows/Linux laboratory and must be submitted as stipulated in the instructions above.

Any deviation from the submission instructions above (including the number and types of files submitted) may result in a reduction in marks for the assessment. Any deviation from the use of the provided template may result in a reduction in marks for the assessment.

If you are unable to submit your work due to technical difficulties, please submit your work via e-mail to comsc-submissions@cardiff.ac.uk and notify the module leader.

Assessment description

This coursework asks you to compare different scoring methods for the sport of squash by simulating matches between players of different ability.

Rules of squash

This coursework uses a simplified summary of the rules of squash. Squash is a racquet game played by two players, and consists of a number of rallies. In each rally, the player who starts is the *server*, and the receiving player is the *returner*. A player wins a rally if the other player is unable to make a legal shot. *Matches* are usually played over a number of games, with each the winner being the player who reaches a certain score first. Professional matches are played over 5 games, with the winner being the first player to win 3.

There are two scoring systems commonly used in squash:

Point-a-rally scoring (PARS)

- The winner of each rally always receives a point (regardless of whether they were the server or returner).
- The first player to reach at least 11 points **and** be ahead by at least 2 points wins the game.
- If the server wins a rally, they continue as server.
- If the returner wins a rally, they become the server.

English scoring

- *Only the server is awarded a point if they win a rally.*
- If the server wins a rally, they receive a point and continue as server.
- If the returner wins a rally, they become the server but don't receive a point.
- The first player to reach 9 points wins the game **unless** the score has reached 8-8.
- If the score reaches 8-8, the player who reached 8 first decides whether to play to 9 or to 10.

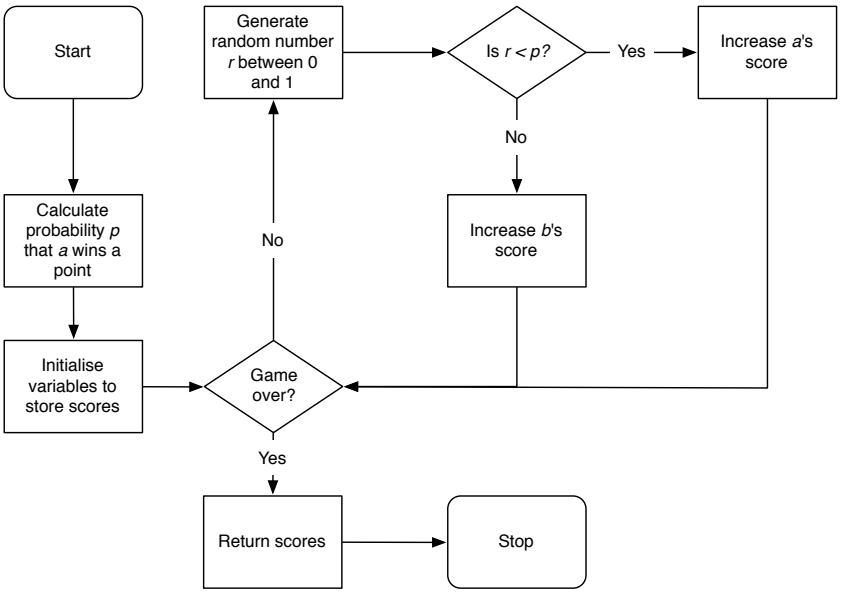
Modelling playing ability

- Assume a player A's ability is represented by an *integer* value r_A such that $0 < r_A \leq 100$.
- In a game between player A and player B, the probability that A wins any given point is:

$$P(A \text{ wins}) = \frac{r_A}{r_A + r_B}$$

Algorithm

The algorithm to simulate a single game of squash using the PARS system is shown in the following flowchart.



Instuctions

Download the Jupyter notebook file `CM1103_coursework_template_2023.ipynb.ipynb` from Learning Central, rename it to `[your_student_number].ipynb`, and complete the questions contained within it. You should execute all cells before submission, so that your file contains output for all cells.

Report sections

Question 2 requires you to write a scientific report (within the Jupyter notebook) with the following sections:

Problem Briefly describe the problem you are investigating. How are you measuring "better" for the scoring schemes? What do you expect to see?

Report marking criteria – is the problem stated clearly and concisely?

Method Summarize the method you are using to investigate the problem. What simulations will you run? What range of parameters will you use? What will you measure and report?

Report marking criteria – is the method stated with sufficient detail to allow someone else to replicate the results?

Assumptions Are there any assumptions that either you or the coursework specification have made that may affect your confidence in the results or their applicability?

Report marking criteria – are the key assumptions listed?

Results Show the results of your simulations – including one figure that allows easy comparison. Explain what the results show.

Report marking criteria – are the results clear and well-presented? Are appropriate figures used? Are figures labelled and titled?

Conclusions Briefly summarize your progress towards solving the problem, highlight any limitations and potential future extensions.

Report marking criteria – are the conclusions stated clearly and concisely, and supported by the rest of the report?

Criteria for assessment

Functionality:

PASS (all marks awarded): The code displays an understanding of basic Python programming; performs the required tasks correctly for the given test data and most reasonable inputs; code may contain minor or easily corrected errors.

FAIL (no marks awarded): The code displays a lack of understanding of basic Python programming; only performs the required task for a very limited subset of inputs; contains significant errors; consists of example code with minimal changes.

Report:

PASS (all marks awarded): Relevant content covering many of the required elements; good evidence; few errors/omissions; generally clear; some omissions in figure.

FAIL (no marks awarded): Little or no relevant content; lack of evidence; extensive errors/omissions; unclear description; no figure.

Achievement:

PASS (all marks awarded): Broadly addresses scope, with some omissions or errors.

FAIL (no marks awarded): Little or no achievement; many significant errors.

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria.

You will also receive feedback covering the quality and style of your code – this will not be included in your mark for this coursework, but is provided to help you improve your programming skills for future modules.

Code quality: Is the code elegant and well-written; easy to run; simplified by the use of built-in languages features where appropriate; readable and easy to follow? Are appropriate functions defined and written to enable reuse between questions?

Individual feedback and marks will be returned by email before the start of the Spring Semester, with the opportunity for 1-1 or group feedback sessions.

Group feedback will be provided via model solutions directly after all student submissions have been made.

Feedback from this assignment will be useful for all modules involving programming, including CM1301: Principles, Tools and Techniques for Secure Software Engineering, CM1210: Object Oriented Java Programming, CM1208: Maths for Computer Science, CM2105: Data Processing and Visualisation.

Help and Support

Questions about the assessment and support for programming can be asked on

https://stackoverflow.com/c/comsc/

and tagged with 'CM1103', at the beginning/end of lectures, drop-in sessions or timetabled lab sessions.