**Assessment type (****):**

☒ Questioning (Oral/Written)

☐ Practical Demonstration

☐ 3rd Party Report

☒ Portfolio

**Assessment Resources:**

Python3 interpreter.

Python IDE, like PyCharm, or VSCode (the latter is not supported by the college), with the ability to use Python Virtual Environments.

Access to Office 365 and Microsoft Word.

Git and access to GitHub.

*Use of some of these items may not occur in this part of the assessment task.*

**Assessment Due**

|  |
| --- |
| This item is due:   **Week 7, 17:00 (5pm) on the day of the scheduled lecture**  Refer to Blackboard for most accurate dates, which may alter due to unforeseen circumstances. We also endeavour to update these documents at the same time.  *It is advantageous for you to attempt to meet this deadline.* |

**Assessment Instructions:**

# Scenario

You are employed as a junior software developer at a boutique software house called *Softwares-RUs* in Perth.

You have been assigned to the Games team and will provide some of the code required for various games that the company builds and releases.

You will be updating the existing code from the previous task to store player passwords in a secure manner. You will be using hashing to accomplish this.

# Instructions

Your code must adhere to certain style guides, the most important being PEP-8 – Style Guide for Python Code. You should familiarise yourself with [PEP-8](https://pep8.org/) before continuing as SRU have adopted PEP-8 as their code style.

Use of a Git repository is standard practice at *Softwares-R-Us* and most of the steps require the use of Git and GitHub.

There are multiple steps in this task, and you must perform each step to a satisfactory level. Please note that you’ll need the results of the tasks outlined in this assessment tool for future tasks as well.

You may answer any questions in the provided template (if available) and the use of screenshots is encouraged. However, you must also provide the actual source code as a ZIP-file of the project for any programming tasks. *Make sure to remove the Virtual Environment folder (*venv *or* .venv*) from the ZIP-file before uploading.*

You must document your code properly. Use docstrings where needed including but not limited to entire classes and methods. Use inline comments to clarify certain parts of code.

If you use any external resources, you should provide references.

**Assessment Instrument:**

|  |
| --- |
| **Step 1 – Knowledge Question (40-70 words)**  In your own words, describe what hashing is in general.  Hashing is the process of converting input data into a fixed-size string of characters, using a hash function. It ensures quick data retrieval, integrity checks, and security by making it difficult to reverse-engineer the original input.  **Step 3 – Knowledge Question (50-90 words)**  Provide a stepwise description (algorithmic) of a) how you can store a password safely using hashing techniques and b) how you can verify that some string is the right password?  Argon2 is considered one of the most secure password hashing algorithms available today. It works by combining the password with a unique salt and then applying a hash function that takes into account memory, time, and parallelism costs. These costs make it resistant to attacks, especially those from powerful hardware like GPUs. When verifying a password, the system re-hashes the user input using the same salt and parameters stored in the database, and if the resulting hash matches the stored one, the password is correct.  **Step 4 – Knowledge Question (20-40 words)**  What is the purpose of a “salt” when hashing a password? What are the two most important properties of a “salt”?  The purpose of a "salt" when hashing a password is to add randomness to the hashing process, ensuring that even if two users have the same password, their hashed values will be different. This prevents attackers from using precomputed tables (like rainbow tables) to reverse-engineer passwords.  **Step 5 – Add password to Player class**  In this step, you will be adding functionality to the Player class to store a password in a safe manner. This step has multiple parts to it.   1. Install the external package argon2-cffi from pypi using pip or your IDE. *Document how you installed this package.*      1. Read the documentation for this package to understand what this package has to offer. 2. Add a method add\_password to the Player class. It should accept a single argument (a string), which is the plaintext password. Determine which function to use from the argon2 package and implement the function to calculate a *hashed* version of the password. Store this value in a private instance variable. Do *not* create a property for this value. You may have to initialise this instance variable in the initialiser method. 3. Add a method verify\_password to the Player class. It should also accept a single argument (a string), which is the plaintext password which should be checked, and return a Boolean indicating whether there is a match. Implement this method to verify that the provided password matches the stored password. 4. Create at least two unit tests to check whether your implementations work correctly. 5. Commit your changes and push to your remote GitHub repository. 6. Answer the question “How does the argon2-cffi package handle salt?”   The argon2-cffi package automatically generates a unique salt for each password when it is hashed. This salt is stored within the hash itself and is used during the verification process, ensuring that even identical passwords will produce different hashes. |

# Submitting your work

ZIP your entire project into a single file.

Make sure to *remove the Python Virtual Environment folder* *(called* venv *or* .venv*) from the ZIP-file* before uploading it into the Blackboard assessments area.