Python3 Programming Project Report
Peruke Game – Game of Chance

STUDENT ID: 2187182

GITHUB REPOSITORY LINK: https://github.com/mionitsa/PerukeGame

INTRODUCTION:

Peruke Game is a game of chance and a dice game of strategy and luck. Two players have 6 numbered disks (from 1 to 6). Players roll the dice one at a time and can choose either to attack an opponent's disk or protect her or his own disk. Players can attack or protect only a disk whose number fell on the cube. If a player chooses to attack an unprotected disk, it becomes the killed one and there are no actions with this disk anymore. If a player chooses to attack a protected disk, it becomes an unprotected one. If a player chooses to protect their own disk, it becomes a protected one. In this game implementation, users are asked to set the number of rounds (number of dice rolls) after which the game ends. The more the non-killed disks player has, the more the points player will have.
Protected disk – 3 points, Unprotected disk – 1 point, Killed disk – 0 point.

REQUIREMENTS:

This Peruke Game Python Program gives an opportunity to play the game alone and with friends but there are a couple of things to do before running the program. First of all, users should be aware of how to download and open ZIP files. Secondly, users should know how to open the Python file with a particular code IDE (Integrated development environment) such as Pycharm, Atom or Node.js. Thirdly, it is important to know how to run the python program within the IDE. It is not needed to know how to do programming or coding because all functional capacity of the program is already provided. Even though all program testing were taken place on MacOS, there are no limitations in terms of operational systems.

Program requirements:

1. A registration and login form.
2. Disks dashboard with current round information
3. Keeping correct count of rounds, players' turns and random dice numbers.
4. Successful interaction with a user. If input errors occur, no bags and crashes.
5. Efficient data storing with an appropriate method of organising information.

DESIGN AND IMPLEMENTATION:

It was decided to divide the code into two files: the gameplay code with the brain of the game and the file with users' initialisation code (registration and login) as well as the code that provides creating 3 JSON files and storing information there. Both files are organised as a bunch of functions that are connected with each other. Importantly, two files are intersected and returning values from one to another. The game is implemented in a terminal without any visual extensions.

Implementation plan:

- Creating a simple general program with two players with attacking and protecting functions.
- Improving the effectiveness of the program by adding exceptions, deleting unnecessary parts of code, and changing the approach where needed.
- Creating the “userInitialisation.py” file with two main functions of login and registration as well as storing all collected information into JSON files:
  - usersLoginDetails.json -> a file with personal details of users (login, password)
  - usersGamesHistory.json -> a file with games history that includes a number of rounds, usernames, a result of the game, final disk dashboards
  - leaderBoard.json -> personal statistics of users (number of games, winnings, losing and draws)

  It was important to create 3 different individual files because of the nature of the information. For example, password information should not be in the same file with game logs due to security reasons.
- Adding a scoring system that contributes to finding the leader of the game after completing a certain number of rounds.
- Creating another mode game that provides to play with the virtual player (computer).
- Creating the design of the program by adding the description, colours, fonts changings, and loading animation to make the program look more user-friendly and easier to follow.
- Creating a GitHub repository and pushing all files using the terminal by committing changes.

Throughout the process, the program has been debugged and tested by finding errors manually after the python interpreter messages of interruptions are revealed. Mainly, computer science and programming knowledge was used as well as the basics of a mathematical problem-solving approach, information security and the ability to find the information.

PROGRAMMING TECHNIQUES USED:

Programming techniques:

- Suitable variable and function names, comments and annotations (*gameType, playersStatisticsRequest, numberOfRounds*).
- Effective Python libraries and modules (*json, random, sys, time, os.path*).
- An efficient approach to connect two parts of code (*from userInitialisation import initialisationOptions, creatingFiles*).
- Reading data from the file and writing data to the file.
- Checks if some files exist in a directory (function "creatingFiles").
- Complex data types such as dictionaries, lists and tuples are used.
- Object-oriented programming (set of functions with mutual/shared objects and attributes using "return" and embedded objects that are identified immediately - *statisticsQuestion(username1, username2 = False)*).
- Comprehensive JSON files actions (load, dump, open with, indent).
- Transforming between data types where necessary *(dictionary elements into coloured string in function "diskBoardForPrint").*
- Coloured terminal text (*"\u001b[41mSomething went wrong. Try again.\u001b[0m"*).
- IF statements conditions using AND/OR/NOT for logical comparisons.
- Variety of implementations of FOR and WHILE loops.
- Recursion.
- Boolean expressions for some variables.
- Exception handling by "try-except" technique.
- Storing current time as an object of JSON file (*datetime.now()*)
- Input handling and analysis – text or numbers.
- Many string and integer methods (*lower(), isdigit(), strip()*)
- Global variables (only 2: *player1, player2*)

INSTRUCTIONS:

In order to successfully run the program it is needed to follow these steps:

1) Download a ZIP file that consists of two main files called "PerukeMainGame.py" and "usersInitialisation.py" as well as supporting files from Tabula or GitHub repository.
2) Run "PerukeMainGame.py" program via Pycharm. The program will start running in Python Terminal at the bottom of the screen. A welcome description of the purple program should appear. Please, make sure everything is working correctly before proceeding.
3) Follow the instructions on the screen to register your user and play the game.

TESTING AND MAINTENANCE:

It is recommended to register many users in order to test the program and store information methods appropriately. Moreover, as a suggestion, you can have a look at JSON files that are automatically created which are called “leaderBoard.json”, “usersGamesHistory.json”, “usersLoginDetails.json” to find out in what format and arrangement data is stored.

For example, user details can be like that: Username - john | Password - John_1995. These user details are absolutely not mandatory to put, it is possible to create your own username and password.

Testing the program can be performed by following the instructions on the screen and interacting with the game itself. It is provided to input any values because all exceptions are considered and an input section will be restarted.

REFLECTION AND NEXT STEPS:

I am satisfied how the working process took place. More specifically, time for the programming project was evenly distributed over two weeks, so the code development was given calmly and step by step. The idea of using OOP in the code was very successfully implemented since some of the functionality of the program was similar and OOP prevents the repetition of parts of the code. Comments and variables names that were put throughout the process were extremely useful as there was no need to memorise which function and value is responsible for what. However, speaking about my previous projects, this difference played a significant role this time as I did not pay attention to naming variables and putting comments appropriately before this project. In the future, I would like to work with Python classes as I am aware that it is really helpful and practical, especially, in terms of programming apps and games.

REFERENCES / OPEN SOURCE DECLARATIONS:

1. https://stackoverflow.com
2. https://www.w3schools.com
3. https://www.geeksforgeeks.org
4. https://docs.python.org/3/

These resources only helped to find solutions to the problems through open methods, and not by the code that was inserted into the project. All the code was written by the student.

TOTAL WORDS COUNT: 1274