CP1401/CP5639 Assignment 2

Task - Risky Business:

For this assessment, you are to plan and then code a medium-sized console-based program in Python 3. This assignment is designed to help you build skills using:

- As in assignment 1: Input, Processing and Output; Decision structures; Repetition structures
- New in assignment 2: Functions; Random numbers; Lists; Files

The assignment also includes a **Developer's Journal** – a document you complete as you plan and code your program to help you focus on and improve your process and become a better developer.

Incredibly Important: This assignment must not be where you first learn about these topics. The subject is designed to systematically teach you these principles through the lectures (first), then the practicals. You should have practised each programming concept/construct many times before you start using it in your assignment. If you get to a point in your assignment where you're not sure about something, go back and learn from the subject teaching (not from the Internet). Remember:
100% of what you need to know to complete this assignment is taught in the subject.

Problem Description:

Risky Business is a game that's totally not about gambling… but rather about risking digital "cash" to win or lose based on random chance. The program starts with a welcome and a main menu with the following options:

- (P)lay
  - Pressing P (or p) plays a turn of the game where you can choose an amount to risk up to your current balance. You can then choose from 3 risk-reward levels, as shown in the **sample output** below. Notice the error checking looping on both the risk amount and the risk option. Notice also that we have explicitly taught how to handle errors like this using the error checking pattern: https://github.com/CP1404/Starter/wiki/Programming-Patterns#error-checking
    You may also remember that we have written useful functions for getting valid inputs before, e.g., https://github.com/CP1401/Practicals/tree/master/prac_06#example
    So… follow what you've been taught and feel confident that you're on the right track! ☺
- (I)nstructions
  - This prints the instructions for the program. Notice that the instructions reference values like the percentages for chance and reward, and remember that we've taught you about using CONSTANTS in places where values appear more than once – like in print statements and in calculations.
- (D)isplay report
  - This displays a report of all the results from the game so far, unless there are no results yet. Use a **list** to store the results of each turn. Notice how the dollar values are formatted and line up neatly. Only store each result in the list. Do not store anything else like the balance, just the result.
- (S)how statistics
  - This displays statistics about the turns as in the sample output: minimum, maximum, and win/loss percentages, unless there are no results yet.
- (Q)uit
  - Choosing this option will end the main menu and then display the report again. Notice that you have been taught how to write menus and you know that (Q) should not be a separate option within the menu, but rather the quit condition with final actions coded outside the main menu loop: https://github.com/CP1404/Starter/wiki/Programming-Patterns#menus
  - When the user quits, they are presented with a summary including the standard report. They are then offered the choice of saving their results to the file results.txt. If they don't say no, the values in the results list are written to the file, one value per line.

Make sure you understand how the program should work (that's the "analysis" step in program development) before you plan it ("design" step), then code it ("implementation"). Don't forget to test your program thoroughly, comparing it to these requirements.

Coding Requirements and Expectations:

1. Make use of named **constants** as appropriate, e.g., for things that would otherwise be "magic numbers", like the chance and reward percentages. Remember the guidelines for constants: https://github.com/CP1404/Starter/wiki/Programming-Patterns#constants
    a. A good way to test that you have used constants properly is that you should be able to change ONE value in ONE place to make the game have a 99% chance of getting the crazy reward… and the instructions should correctly show this value.
2. You are expected to include two kinds of useful **comments** in each of your program:
    a. *Every* function should have a `"""docstring"""`.
    b. Use `# block` comments for things that might reasonably need a comment.

    Do not include unnecessary comments as these are just "noise" and make your program harder to read. See the subject teaching for how to properly write good comments: https://github.com/CP1404/Starter/wiki/Styles-and-Conventions#commenting
3. **Error checking** should follow our standard pattern: https://github.com/CP1404/Starter/wiki/Programming-Patterns#error-checking
4. Follow what you've been taught about how to write **menus**, and know that (Q) should not be a separate option within the menu: https://github.com/CP1404/Starter/wiki/Programming-Patterns#menus
5. **Functions** should be used for sections of the program and repeated tasks as you have been taught. Follow the DRY (Don't Repeat Yourself) principle and consider turning repeated code into functions. **Do not use global variables.** Any use of global variables will result in a maximum mark of 1/10 for the functions criterion. Here are some possibilities for functions:
    a. displaying the report is done the same way in multiple places
    b. playing a turn is a significant section
    c. getting a valid risk amount and risk option could each be functions
    d. displaying the instructions is large enough that turning this into a function makes the main function easier to read
    e. saving to a file contains low-level detail that should be abstracted away into a function

    The main menu and one-off program behaviour (like the start and end) should all be part of the main function – like in our examples and teaching.
6. **Sample output** from the program is provided. Follow this to meet requirements.
7. The way computers work with floating point numbers means that your results may end up with problematic precisions, so **round** your results and balance to 2 decimal places when needed:

```
balance = round(balance, 2)
```

We strongly suggest you work incrementally on this task: focus on completing small parts rather than trying to get everything working at once. This is called "iterative development" and is a common way of working, even for professionals. A suggested approach to this is described below:

1. Start with planning and pseudocode – this is important for your process as well as your journal (see below for details about recording your process in your journal).
2. Start coding with the main function and creating a working menu using the standard pattern. For each menu item, just print something (like "… play…").
3. Choose one function/section at a time to implement. E.g., start with the function to play a turn and get this working, then call the function from your main menu.
4. When you do a more complex section, keep it simple first, then add complexity. E.g., when getting a risk amount, ignore the error-checking to start with. Get it working properly, then add error-checking.

5. When writing the function to play a turn, start with just generating and displaying a random number between 1 and 100… but notice this is never printed… so print it as a helpful debugging output until your function is working correctly, then remove the print statement.
6. When writing and testing code with randomness, you can encourage it towards what you want to test by modifying your constants or starting values. E.g. when you want to test what happens the user is successful with a crazy risk, change the constant from 5 to 99 to make sure they get it. Then change it back. Don't waste time running your program many times to hopefully get the random scenario you want to test.
7. The file saving can be done last as it is optional for the user.

Journal:

A significant desired outcome for this subject is you learning to develop solutions to problems systematically and thoughtfully. That is, we are not only interested in the final product but in your *process* and the *lessons* you have learned through your experience. To encourage you in learning the systematic problem-solving process, you will record your work experiences and insights in a simple journal for this assignment.
Use the template file you have been provided with.

There are three parts to your journal:

1. Assignment 1 Reflections and Lessons
2. Work Entries
3. Summary

Assignment 1 Reflections and Lessons:

Before you begin working on this assignment, take some time to reflect on what you learned about your process through assignment 1, including how you will make changes this time based on any feedback you received. You could consider your previous reflection exercise from the practicals: https://github.com/CP1401/Practicals/tree/master/prac_07#reflection

Work Entries:

Each time you work on the assignment, record an entry in your journal that includes:

- Date and time you worked, including duration
- What you worked on with simple details, enough that someone reading it would understand
- Any difficulties you faced and how you overcame them

Please do not include multiple entries for short work sessions. If you did 7 minutes, took a break then came back and did 37.5 minutes… we don't need this level of detail – just include a single entry of about 45 minutes. The entry detail can be quite short, but make sure your focus is on your ***process***, not your actual code/work.

Summary:

Summarise the lessons you learned about the problem-solving process (not about Python code) through doing this assignment. **This is the most important part** – where you **reflect** on your process and show what you have learned. Did anything you considered or changed based on assignment 1 reflection prove important or useful? How are you improving in terms of following a systematic development process? What would you differently next time?

Please note that the only reasonable way to write a journal is regularly, ***as*** you develop your solution. A journal that is completed at the end of the assignment after you've finished everything is not a journal and will not aid your learning experience as much.

You are not allowed to use GenAI (like ChatGPT, Copilot, etc.) for this assignment at all. Please understand that journal writing done by AI is usually very obviously written by AI as it is far too general, saying unfounded things like, “I have learned the value of breaking a larger problem into smaller parts”. We will give very low marks to journals that are not specific and personal, whether we think they are written by AI or not. So, be clear, specific, personal, and use examples.

Here is a sample journal entry that shows a 'satisfactory' level:

> **25/07/2025, 8:30 – 9:30am**
>
> **Work**: Wrote pseudocode for main function; nearly completed start and menu section.
>
> **Challenges**: It took a while to remember how to deal with lists and functions in pseudocode. I checked the "Pseudocode Guide" and followed the examples, which helped.
> I am so grateful for the excellent guides and high-quality teaching in this subject :)

Sample Output:

It should be clear which parts below are user input (not printed, but entered by the user).

```
Welcome to Risky Business
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: why?
Invalid choice
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: i
Risky Business
Each turn, you can risk some of your cash to try and win a reward.
You can choose a risk level:
- conservative (68% chance for a +24% reward)
- aggressive (42% chance for a +64% reward)
- crazy (7% chance for a +144% reward)
If your risk-taking doesn't pay off, you lose the amount you choose to risk.
Risky Business. You win some. You lose more.
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: d
No risks taken yet. Get started...
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: s
No risks taken, so there are no statistics.
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: quit
Invalid choice
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: p
Enter amount to risk, up to $1248.48: $9999
You can't risk that amount!
Enter amount to risk, up to $1248.48: $0
You can't risk that amount!
Enter amount to risk, up to $1248.48: $248.24
(C)onservative, (A)ggressive, Cra(Z)y: success!
Please choose from the available options.
(C)onservative, (A)ggressive, Cra(Z)y:
Please choose from the available options.
(C)onservative, (A)ggressive, Cra(Z)y: z
You lost $248.24
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: s
Best result : $-248.24
Worst result: $-248.24
  0.0% (0/1) of your turns were gains
100.0% (1/1) of your turns were losses
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: p
Enter amount to risk, up to $1000.24: $500
(C)onservative, (A)ggressive, Cra(Z)y: c
Congratulations! You earned $120.00
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: p
Enter amount to risk, up to $1120.24: $1000
(C)onservative, (A)ggressive, Cra(Z)y: A
Congratulations! You earned $640.00
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: d
Risky Results Report:
Starting balance: $1248.48
$  -248.24  ->  $  1000.24
```

```
$   120.00  ->  $  1120.24
$   640.00  ->  $  1760.24
Current balance:$  1760.24
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: s
Best result : $640.00
Worst result: $-248.24
 66.7% (2/3) of your turns were gains
 33.3% (1/3) of your turns were losses
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: p
Enter amount to risk, up to $1760.24: $1400.24
(C)onservative, (A)ggressive, Cra(Z)y: a
Congratulations! You earned $896.15
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: p
Enter amount to risk, up to $2656.39: $2650
(C)onservative, (A)ggressive, Cra(Z)y: z
You lost $2650.00
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: oh, that's sad
Invalid choice
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: s
Best result : $896.15
Worst result: $-2650.00
 60.0% (3/5) of your turns were gains
 40.0% (2/5) of your turns were losses
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: p
Enter amount to risk, up to $6.39: $6.39
(C)onservative, (A)ggressive, Cra(Z)y: c
Congratulations! You earned $1.53
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: p
Enter amount to risk, up to $7.92: $7.92
(C)onservative, (A)ggressive, Cra(Z)y: c
Congratulations! You earned $1.90
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: p
Enter amount to risk, up to $9.82: $9.82
(C)onservative, (A)ggressive, Cra(Z)y: a
You lost $9.82
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: p
You have no $$$... time to quit!
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: d
Risky Results Report:
Starting balance: $1248.48
$  -248.24  ->  $  1000.24
$   120.00  ->  $  1120.24
$   640.00  ->  $  1760.24
$   896.15  ->  $  2656.39
$ -2650.00  ->  $     6.39
$     1.53  ->  $     7.92
$     1.90  ->  $     9.82
$    -9.82  ->  $    -0.00
Current balance:$    -0.00
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: s
Best result : $896.15
Worst result: $-2650.00
 62.5% (5/8) of your turns were gains
 37.5% (3/8) of your turns were losses
(P)lay (I)nstructions (D)isplay Report      (S)tatistics   (Q)uit
Choose: q
Summary of your 8 turns:
Risky Results Report:
Starting balance: $1248.48
$  -248.24  ->  $  1000.24
$   120.00  ->  $  1120.24
```

```
$   640.00  ->  $  1760.24
$   896.15  ->  $  2656.39
$ -2650.00  ->  $     6.39
$     1.53  ->  $     7.92
$     1.90  ->  $     9.82
$    -9.82  ->  $    -0.00
Current balance:$    -0.00
Save your results to a file? (Y)es or (N)o? y
Results saved.
Thank you for playing Risky Business.
```

At the end of this first run, the file results.txt would contain:

```
-248.24
120.0
640.0
896.15
-2650.0
1.53
1.9
-9.82
```

Now here is a second run.

```
Welcome to Risky Business
(P)lay (I)nstructions (D)isplay Report     (S)tatistics  (Q)uit
Choose: p
Enter amount to risk, up to $1248.48: $1248
(C)onservative, (A)ggressive, Cra(Z)y: z
You lost $1248.00
(P)lay (I)nstructions (D)isplay Report     (S)tatistics  (Q)uit
Choose: p
Enter amount to risk, up to $0.48: $0.48
(C)onservative, (A)ggressive, Cra(Z)y: a
You lost $0.48
(P)lay (I)nstructions (D)isplay Report     (S)tatistics  (Q)uit
Choose: p
You have no $$$... time to quit!
(P)lay (I)nstructions (D)isplay Report     (S)tatistics  (Q)uit
Choose: q
Summary of your 2 turns:
Risky Results Report:
Starting balance: $1248.48
$ -1248.00  ->  $     0.48
$    -0.48  ->  $     0.00
Current balance:$     0.00
Save your results to a file? (Y)es or (N)o? n
Thank you for playing Risky Business.
```

At the end of this second run, the file results.txt would be unchanged.

And here is a third run where the user just selected quit as the first menu choice.

```
Welcome to Risky Business
(P)lay (I)nstructions (D)isplay Report     (S)tatistics  (Q)uit
Choose: q
Summary of your 0 turns:
Risky Results Report:
Starting balance: $1248.48
Current balance:$  1248.48
Save your results to a file? (Y)es or (N)o? why would I?
Results saved.
Thank you for playing Risky Business.
```

At the end of this third run, the file results.txt would be empty.

Submission:

Write your pseudocode and code in a single Python file called **`a2_risky.py`**.

Note: You are only required to write **pseudocode** for your main function and the play function. You are welcome and encouraged to do it for the whole program, but we will only mark these two functions' pseudocode. However, your main function must be substantial and appropriately designed (e.g., do not use a "menu" function). Remember, "main should look like the whole program", with the details in the functions:
https://github.com/CP1404/Starter/wiki/Programming-Patterns#main-program-structure

Copy the module docstring below and update this with your own details and your pseudocode, then your solution code below.

```
"""
CP1401 2025-2 Assignment 2
Risky Business
Student Name: XX
Date started: XX

Pseudocode:


"""
```

Use the provided template file for your journal: **`CP1401-A2-Journal.docx`**.

DO NOT zip/compress your files together. Submit two separate files (.py and .pdf) as this makes it considerably easier for markers to access your work (and it's easier for you!).

Submit your two files, by attaching them in LearnJCU under Assessments by the date and time specified on LearnJCU. Submissions received after this date will incur late penalties as described in the subject outline. Note that the assignment allows multiple submissions, but we will only look at and assess the most recent submission. If you need to resubmit, then submit both files again.

Integrity:

The work you submit for this assignment must be your own. Submissions that are detected to be too similar to that of another student or other work (e.g., code found online) will be dealt with according to the College procedures for handling plagiarism and may result in serious penalties.

The goals of this assignment include helping you gain understanding of fundamental programming concepts and skills, and future subjects will build on this learning. Therefore, it is important that you develop these skills to a high level by completing the work and gaining the understanding yourself. You may discuss the assignment with other students and get assistance from your peers, but you may not do any part of anyone else's work for them and you may not get anyone else to do any part of your work. Note that this means **you should never give a copy of your work to anyone or accept a copy of anyone else's work, including looking at another student's work or having a classmate look at your work**. If you require assistance with the assignment, please ask **general** questions in #cp1401 in Slack, or get **specific** assistance with your own work by talking with your lecturer or tutor.

The subject materials (lecture notes, practicals, textbook and other guides provided in the subject) contain all of the information you need for this particular assignment. You should not use online resources (e.g., Google, Stack Overflow, ChatGPT, etc.) to find resources or assistance as this would limit your learning and would mean that you would not achieve the goals of the assignment - mastering fundamental programming concepts and skills.

Marking Scheme:

Ensure that you follow the processes and guidelines taught in the subject to produce high quality work. Do not just focus on getting your code working. This assessment rubric provides you with the characteristics of exemplary to very limited work in relation to task criteria, covering the outcomes:

- SLO1 - apply problem-solving techniques to develop algorithms in the IT context
- SLO2 - apply basic programming concepts to develop solutions

Criteria	Exemplary (9, 10)	Good (7, 8)	Satisfactory (5, 6)	Limited (2, 3, 4)	Very Limited (0, 1)
Journal SLO1 20%	Good first reflection showing learning; significant number of entries showing good problem-solving process including starting with planning; summary highlights useful lessons; specific, personal and contains good examples.	Exhibits aspects of exemplary (left) and satisfactory (right)	Some first reflection; reasonable number of entries; process is not exemplary (e.g., planning or testing are missing or not in appropriate order); summary lacks insight and clear lessons; not specific or personal enough, lacking good examples.	Exhibits aspects of satisfactory (left) and very limited (right)	No journal or trivial effort.
Algorithms SLO1 10%	Clear, well-formatted, consistent and accurate pseudocode that completely and correctly solves the problem.		Some but not many problems with algorithm (e.g., incomplete solution, inconsistent use of terms, inaccurate formatting).		Many problems or algorithm not done.
Correctness SLO2 20%	Program works correctly for all functionality required.		Program mostly works correctly for most functionality, but some required aspects are missing or have problems.		Program works incorrectly for all functionality required.
Identifier naming SLO2 10%	All variable, constant and function names are appropriate, meaningful and consistent.		Multiple variable, constant or function names are not appropriate, meaningful or consistent.		Many names are not appropriate, meaningful or consistent.
Use of code constructs SLO1, SLO2 15%	Appropriate and efficient code use following the standard patterns, including good choices for variables, constants, processing, decision and repetition.		Mostly good code use but with problems, e.g., not following patterns, unnecessary or repeated code (DRY), missing constants, poor choice of decision or repetition.		Many significant problems with code use. **Any use of while True will result in < 5.**
Use of functions SLO2 15%	Functions and parameters are appropriately used, functions are well reused to avoid code duplication.		Functions used but not well, e.g., breaching SRP, poor use of parameters, unnecessary duplication, main code outside main function.		No functions used or functions used very poorly. **Any use of global variables will result in < 5.**
Commenting SLO2 5%	Every function has a docstring, some helpful block/inline comments, module docstring contains all details, no 'noise' comments.		Some noise (too many/unhelpful comments) or missing some function docstrings or incomplete module docstring.		Commenting is very poor either through having too many comments (noise) or too few comments.
Formatting SLO2 5%	All formatting meets PEP8 standard, including indentation and spacing. PyCharm shows no formatting warnings.		Multiple problems with formatting reduce readability of code. PyCharm shows formatting warnings.		Readability is poor. PyCharm shows many formatting warnings.