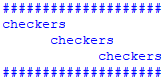
Pre-release tasks – part 1

# Welcome message

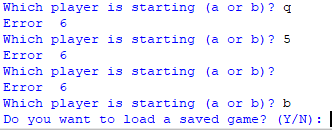
Useful link: <https://pyformat.info/>

Define a procedure called **welcome** that prints out the following message within **20 characters using .format** and call it at the start of the **game procedure.**



# Choose starting player

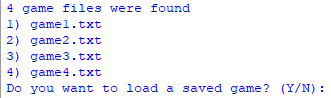
Define a function called **StartPlayer** that asks the user to enter the letter of the first player. The function should return the validated character to the **game procedure** and save the result in the variable **NextPlayer**.



# Directory scraping

Useful link: <https://docs.python.org/3/library/filesys.html>

Define a procedure called **scrape** that prints out all of the game text files in the same directory as the python program and call it before **SetUpBoard** in the **game procedure.** If no games are found use **DisplayErrorCode** to print error 5.



Pre-release tasks - part 2

# Taking opposition pieces

Update the ValidJumpfunction to allow jumping over the OppositePiecePlayer.

Update the MakeMove function to remove the opponents piece from Board and set its row and column positions to -1.

Test the functionality by loading **game3.txt**

# Score points when taking opponent pieces

Update the MakeMove function so that when a piece is taken the player gains one point in index position [0][2] of their statistics list.

Test the functionality by loading **game3.txt**

# Taking opposition pieces and jump own pieces

Update the ValidJump function so that you can jump over the players own pieces and not remove them from the board or score a point. It still must be possible to jump over the opponent’s pieces and score a point.

Test the functionality by loading **game3.txt**

# Dames can move backwards

Update ListPossibleMoves so that if the piece is a dame, the player is allowed to select a backwards diagonal movement.

# Automated game saving

Create a new procedure called Save that takes four parameters Board, A, B and filename. The procedure should save the current state of the game in a text file so that it can be loaded correctly in future.

Update the Game procedure so that straight after the SetUpBoard function has run the user is asked to enter the filename they wish to save their game as. The filename must not be left blank.

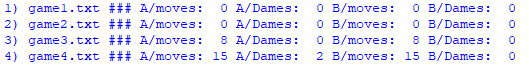
Update the Game procedure so that the Save procedure is called before the PrintPlayerPieces procedure is run.

Pre-release tasks – part 3

# Save analyser

This is an extension of **Directory scraping** from part 1 and refers to the procedure **scrape**.

When scraping the save files, print out the statistics of each player for that game in the following format:



Each number that is printed is aligned to the right of two spaces so that every statistic is neatly in line.

# Real checkers part 1

This is an extension of **Taking opposition pieces** from part 2.

One of the rules of checkers is that if a piece can be taken, it must be taken.

There are multiple possible methods of completing this however you may wish to extend the **class MoveRecord** to include self.CanTake = False and then update **ClearList** and **ListPossibleMoves.**

# Real checkers part 2

This is an extension of **Real checkers part 1**.

One of the rules of checkers is that when taking pieces, you can keep taking pieces as long as they are in line of each other.

You have two options to complete this challenge:

1. Allow the manual selection of jumping over each piece (player must not be able to move another piece than what they moved originally)
2. Automatically take both pieces and move to the correct position on the board following both takes

**Game 3 is setup to make a6 double take from the first move.**

Pre-release tasks – part 4

# Capturing backwards

In English draughts pieces can jump only forwards; in international draughts and Russian draughts men can jump both forwards and backwards.

Update the relevant functions and procedures to allow opposition pieces to be taken both forwards and backwards by a normal piece.

# Double capturing backwards and forwards

This is similar to **Real checkers part 2** from **part 3** and your code from that may already function correctly.

Update the relevant functions and procedures to allow opposition pieces to be taken both forwards and backwards by a normal piece if there are further moves available.

# Flying Dames part 1

In international draughts, kings (also called flying kings) **move any distance along unblocked diagonals**, and may capture an opposing man any distance away by jumping to any of the unoccupied squares immediately beyond it.

Example of usage: <https://goo.gl/2yJN7D>

Update the relevant functions and procedures to allow this to happen.

# Flying Dames part 2

Because jumped pieces remain on the board until the turn is complete, it is possible to reach a position in a multi-jump move where the flying king is blocked from capturing further by a piece already jumped.

Update the relevant functions and procedures to allow this to happen.

Pre-release tasks – part 5

# Make Player skip turn

Allow player 'a' and 'b' to be able to skip their turn

# Input file ends with ".txt"

The user has to enter a file which ends with ".txt", else it will add it for them, or it will iterate and ask again.

# Display number of moves taken

Displays the number of moves taken after each turn.

# Displays error value meanings

Error values 1-4 don't have any notation as to what each error means. Add a description for each one when displayed.

# Random player starts

Allow player 'a' and 'b' to be randomly chosen at the beginning of the game.

# Player B is controlled by the computer

Have an AI control B instead of having another player do it. It isn't too difficult to set up. I would recommend replacing the function call to SelectMove() in Game() with a new function to generate a valid number for PieceIndex. The way the AI works can be up to you, whether it is just random or actually has a strategy to it.