# Final Report: Mancala Modeling

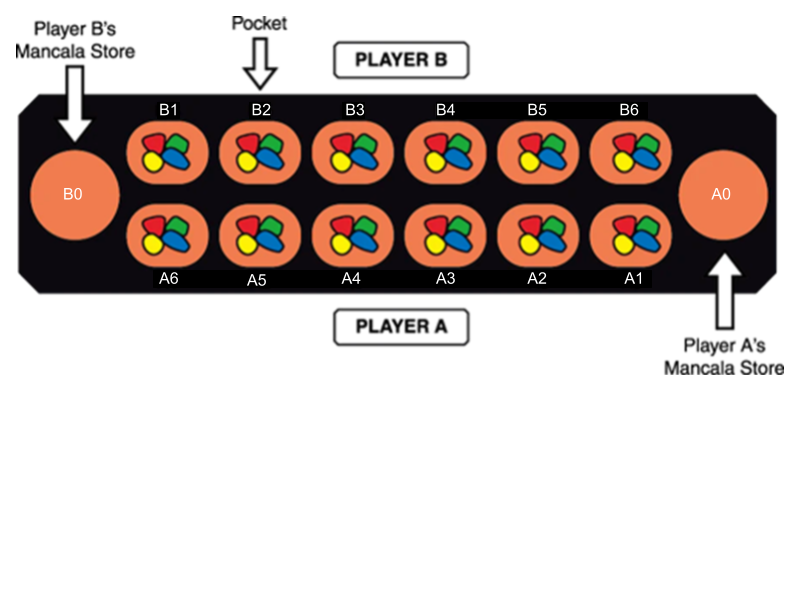
Amy Brons, Alan Teng, Dan Muntenau, Kate Edgar

Group 14 - CISC 204

Project Summary

*Mancala Description:* Mancala is an ancient boardgame where you strategically ‘sow’ and ‘capture’ until you run out of pieces. Played by two people, each player tries to get a higher score by strategically picking and placing game pieces (traditionally gems, but any small object works) around the playing area, which is made up of 12 pits or pockets (six per side) and two stores (larger pockets at the end of each board). To set the game up, four gems are placed in each of the 12 pits on the board. Once it has been decided who goes first, that player picks up all the pieces from one of the pockets on their side of the board and moving one place to the right, begins depositing a gem in each pocket until they have no more left in their hand. If the player’s last deposit lands in their store, they get another turn. Gems are not deposited in the opponent’s store. Similarly, if the player’s last deposit lands in an empty pit on their side of the board, they claim all the pieces in pit directly opposite theirs. The game ends when either of the players has six empty pits on their side, winner is determined by who has the most pieces in their respective store.

*Visualization of a Mancala board:*



For our project, we will be modelling the optimal next move for player A, given a randomized board state. For our project, we will be modelling the optimal next move for player A, given a randomized board state. The recommended move is based on the following strategy:

1. Putting player A’s final gem in their store
2. Preventing player B from depositing their final gem in their store
3. Collecting from player B’s pit
4. Move the right most non-zero pit

# Propositions

Notation: Position is denoted by the variables ‘r’ and ‘c’ where ‘r’ represents the side of the board (A or B) and ‘c’ represents the columns (0 through 6 where 0 is the player’s store and 6 is the furthest pit from the respective store). The number of gems in a selected Finally, x represents a player (A or B).

Modelling plausible game state

* The pit where the final seed gem is dropped.
  + - * Fr, c = True if the pit is where the final gem seed was dropped
* The selection of the pit
  + - * Sc = True if pit is on the same side as the player and is selected
* Position of the pit and how many gems it contains
  + - * Pg, r, c = True if there are gems in the selected pocket
* Identifying which player gets another turn
  + - * Tx = True if player x gets another turn
* Determining if the player collects gems
  + - * Cg, r, c = True if player A collects gems from the indicated pit
* Determining if two pits are opposite of each other
  + - * Oc = True if the columns in row A and B are the same

# Constraints:

Board specific constraints

* A pit can only have a fixed of gems
  + - * Ex. P0, r, c ∧ ¬ P1, r, c ∧ ¬ P2, r, c ∧ ¬ P3, r, c ∧ ¬ P4, r, c ∧ ¬ P5, r, c ...
* There can only one final gem seed position
  + - * Ex. F1,0 ∧ ¬ F1,1 ∧ ¬ F1,2 ∧ ¬ F1,3 ∧ ¬ F1,4 ∧ ¬ F1,5 ...
* A selected pocket must have gems seeds in it
  + - * Sc ∧ ¬ P0, r,c

Game constraints

* Selecting a pit will lead to a certain fixed board state
  + - * Ex. Sc →  Pg1, r, c ∧ Pg2, r, c-1 ∧ Pg3, r, c-2 ∧ Pg4, r, c-3 ∧ Pg5, r, c-4 ∧ Pg6, r, c-5... ∧ Fr, c
* If the final gem is dropped in player A’s store, then player A gets another turn
  + - * Fr,0  → A
* Player A collects gems seeds from two opposite pit if Player A’s last gem seed is done in an empty pit that is not the player’s bank
  + - * Fr,c ∧ P1, r, c → Cg, B, c

Model Exploration:

Mancala is a game that requires you to anticipate your opponent’s moves and set up future plays for yourself accordingly. Since we limited the scope of our model to a single step play, it became difficult for us to figure out how to add in this anticipation element. To combat this, we’ve designed our model to analyse the following four strategies (listed in order of importance):

1. Putting the final gem in the player’s store
2. Block the opponent from putting a gem in their store
3. Put the player’s gem into the opponent’s empty pit
4. Empty the player’s pit
5. If all else fails, play the right most pit

Given a randomized board state, the model will run through those four propositions. The proposition with the highest truth value will be selected as the optimal next move.

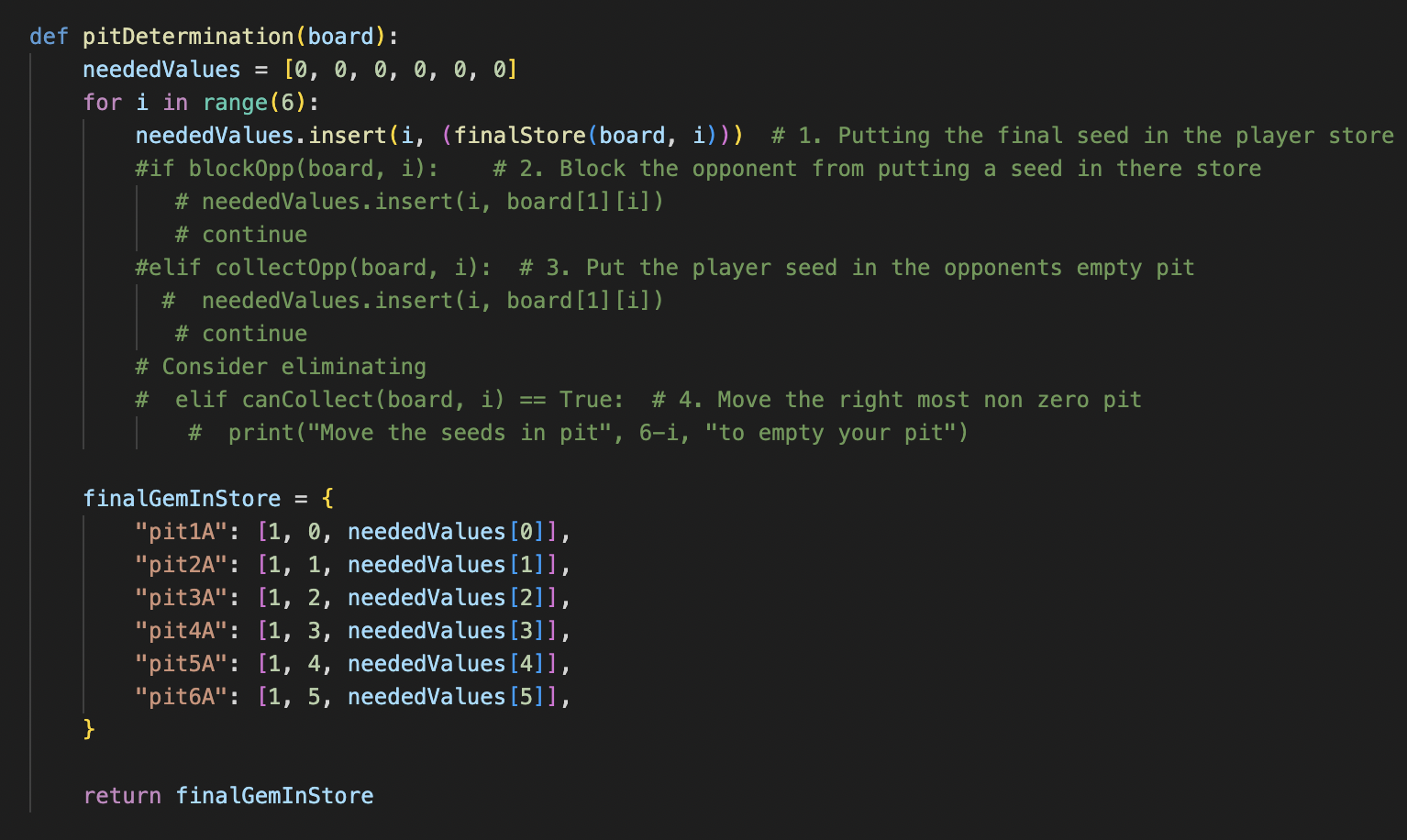
To being with our project we first constructed our logic using Bauhaus. In this file, we initially cam across some difficulty in the way of determining how to translate our constructed logic into the python file. After much deliberation, we were able to construct a solid logic run file.

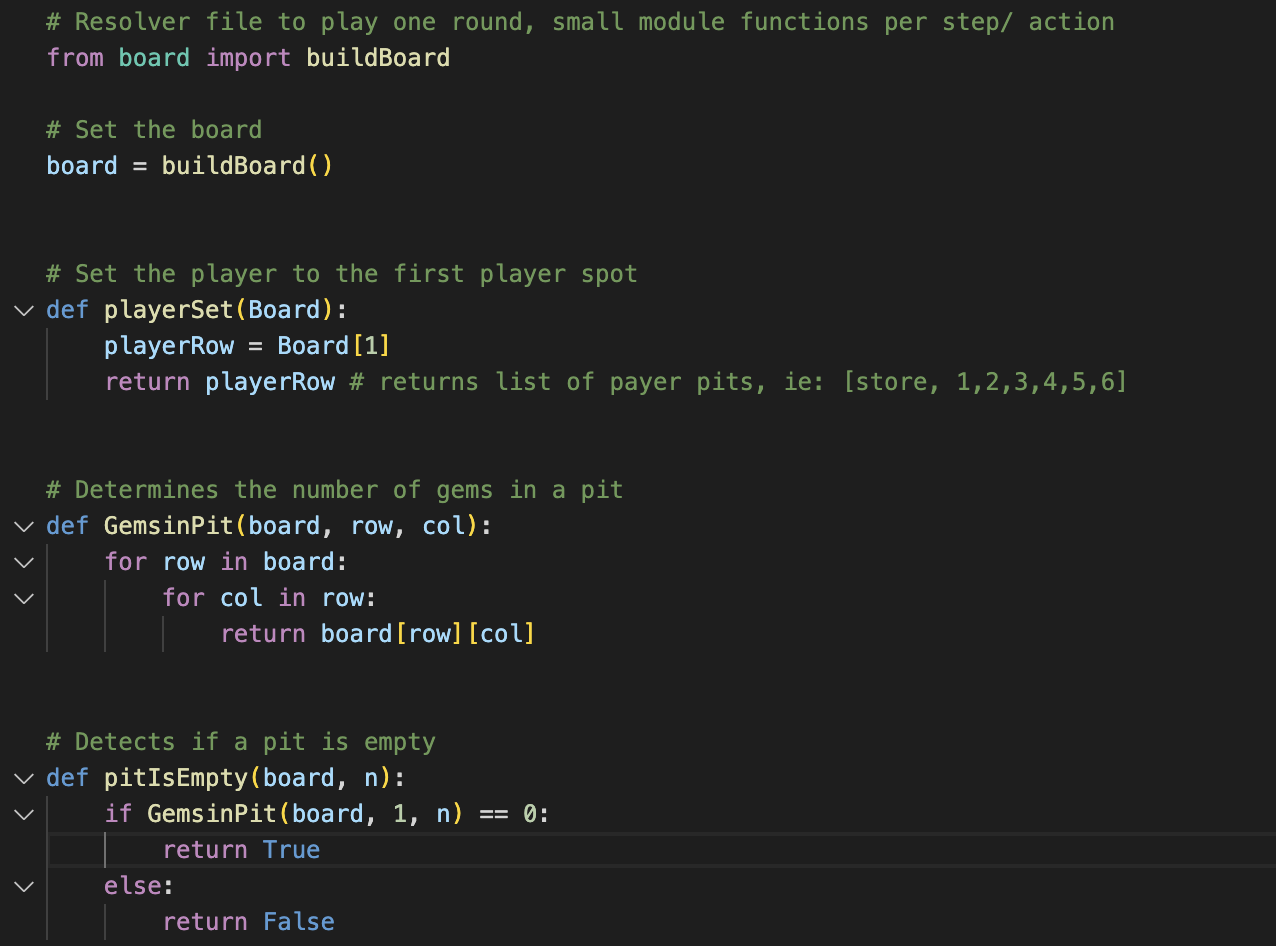
*## ATTACH SCREEN SHOTS*

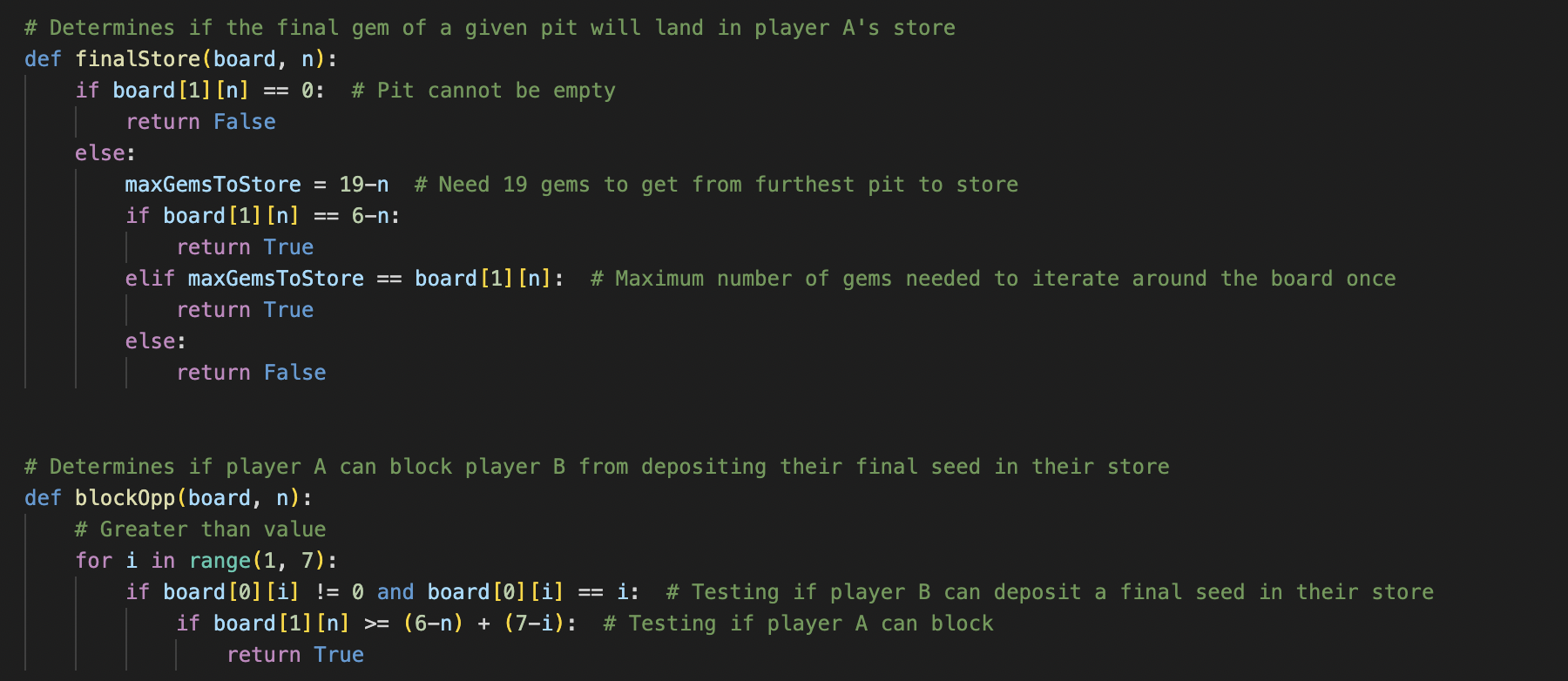
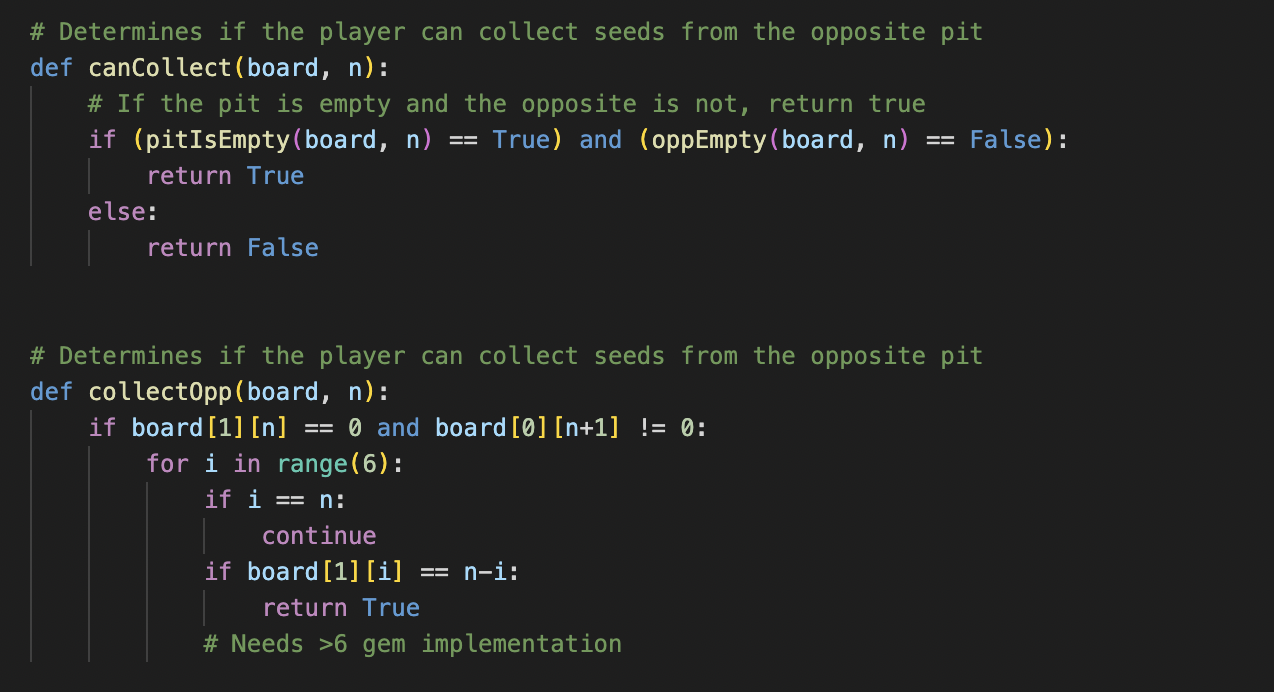
We can started to get good modelling feedback from this code, so we could be sure that all the possible modelling possibilities could be determined. Here is the output from our *run.py* file:

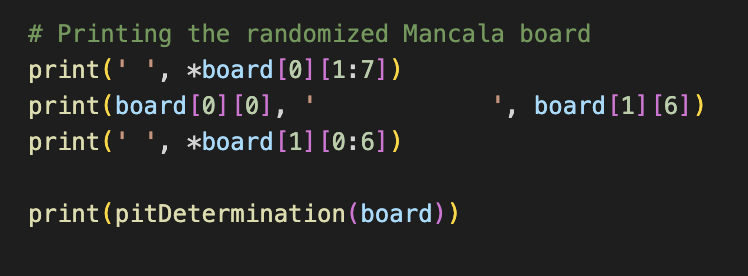
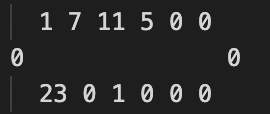
*#ATTACH SCREEN SHOTS*

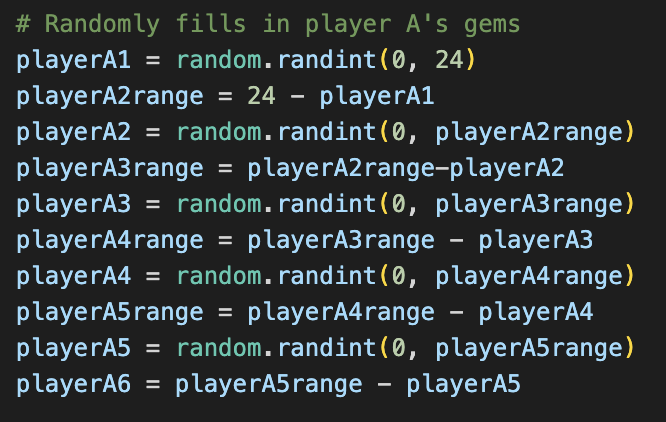
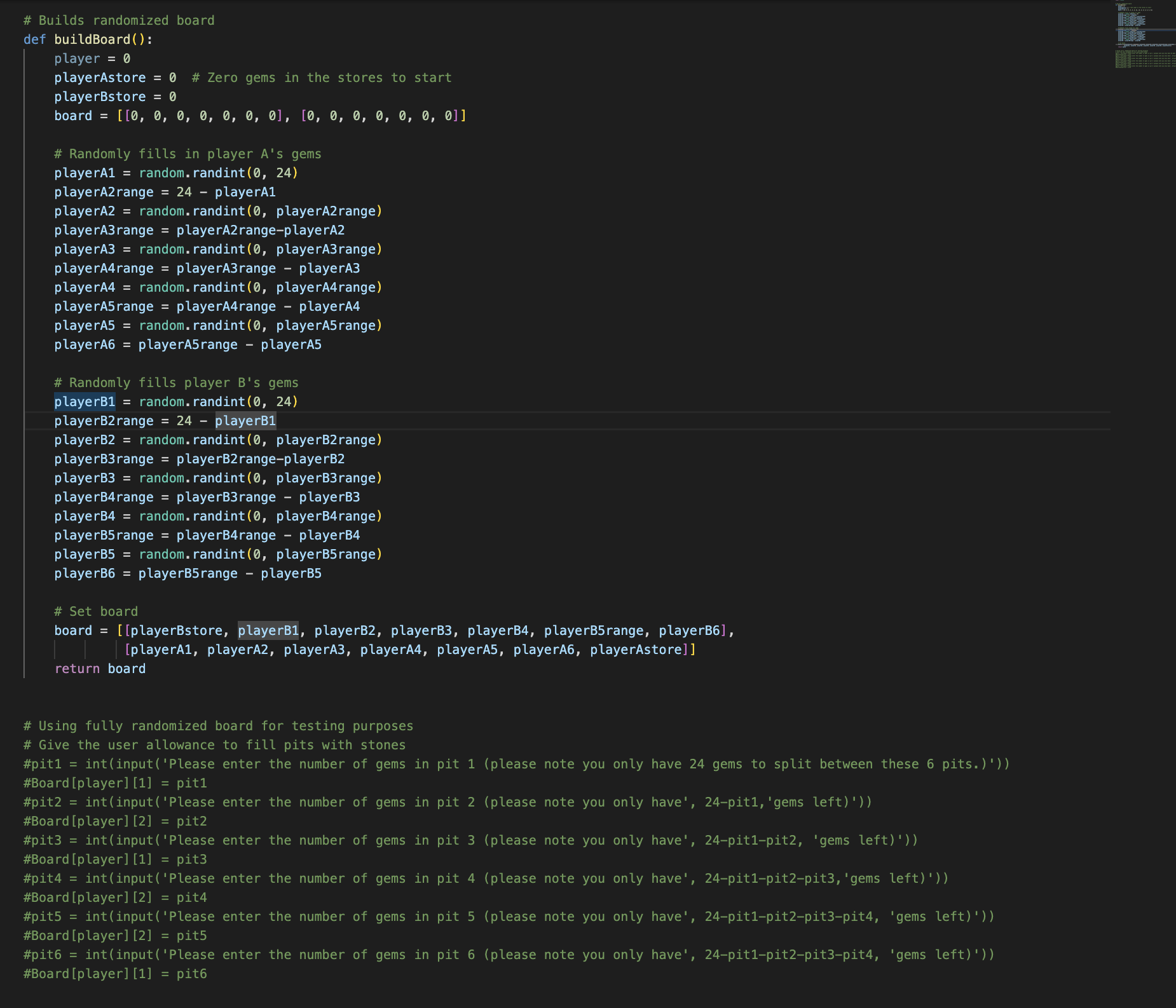
The second part of our modelling was creating a resolver file that modelled the board on the same logic, which we could use for output, visualization, testing and debugging. The integration of these two files created many issues, but created a more stable and well rounded model.

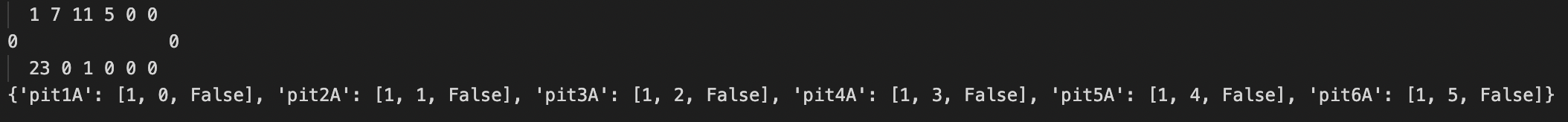
Here is some of the code that we created as functions, equal to the logic in *run.py.*

**

**

As mentioned before, a small visualization of the board is produced in this file, as seen here for example:

This visualization is created in-part due to a file called *board.py* which randomizes the gems in each pit, off of the initial 24 gems per player.

Finally, here is our full resolver output.

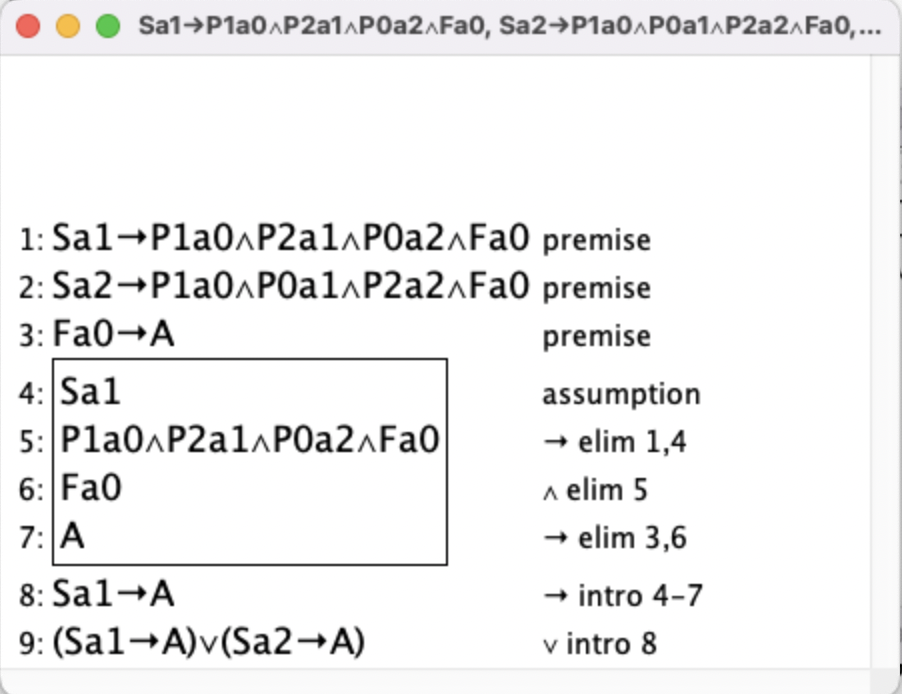
Here is the final logic check and test on the resolver. Returned is the initial board. Under the board is a list with the following: Pit name: [pit row, pit column, boolean: Can this pit return one of our objectives?].

If True is returned, the pit can be chosen to solve one of the strategies. If all is false, strategy 5 can be chosen. For example, in the return above, all are false. Therefore, the rightmost pit will be chosen. In the lower return, pit1 will solve for a strategy, therefore pit1 will be chosen.Image

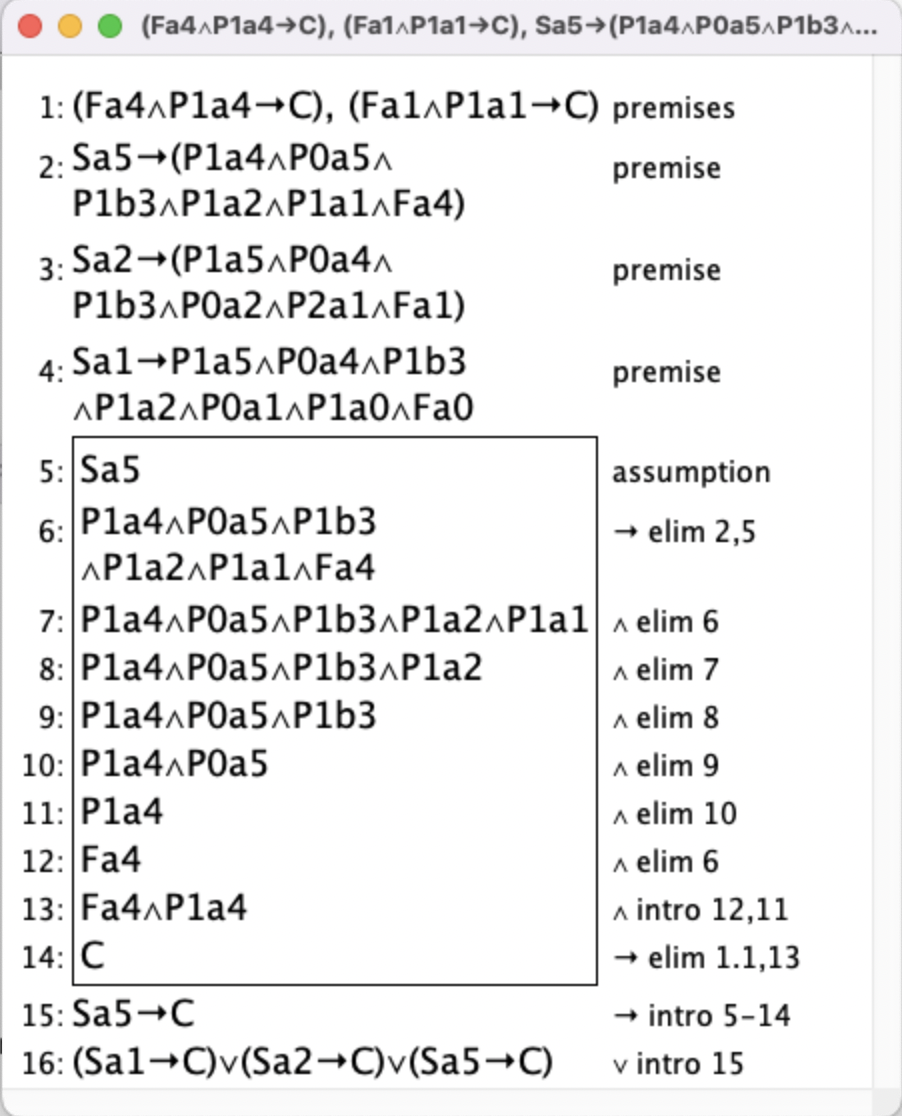
Ideally, the logic file would be able to import this, and return the ideal pit in the case that two pits return true. In the case of this project, that will not be solved for, as this would land outside of our determined project scope.

# Jape Proofs: Our first proof is to check if we can get an extra turn by looking at all pit. In this scenario, we have 2 gems on a2 and 1 gem on a1 and every other pit has no gem, We model using this formula:

# Sa1 → P1a0∧P1a1∧P0a2∧Fa0, Sa2 → P1a0∧P1a1∧P0a2∧Fa0, Fa0 → A⊢(Sa1 → A)∨(Sa2 → A)



Our second proof is the same as before, but we check to see if there are any pits that leads to us collecting gem directly from two opposing pits. In this scenario, there is a gem in A5, B3, A2, A1 and non in the other pits. We model it using this formula:

**(Fa4**∧**P1a4 →C),(Fa1**∧**P1a1 →C), Sa5→(P1a4**∧**P0a5**∧**P1b3**∧**P1a2**∧**P1a1**∧**Fa4), Sa2→(P1a5**∧**P0a4**∧**P1b3**∧**P0a2**∧**P2a1**∧**Fa1),   
Sa1→P1a5**∧**P0a4**∧**P1b3**∧**P1a2**∧**P0a1**∧**P1a0**∧**Fa0**⊢**(Sa1 → C)**∨**(Sa2 → C)**∨**(Sa5 → C)**

Our final proof is proving that selecting pit a1 (which would have 5 gems) would block the collection of a3 and b4. This won’t be applicable as we would need 2 instances of the model. So we model it using a revised formula.

# Sa1 → (P1a0∧P2b6∧P1b5∧P1b4∧P0a1∧P1b3∧¬P0b4) ⊢Sa1 →(P0b4∧Fb4→¬C)Image

# First-Order Extension:

The inclusion of predicate logic settings would be used to show the ideal move that would be achieved, as above. This would be by stating that there is a seed that exists that fulfills the most important strategy. For example, we could extend the first-order to say: there is a pocket that will result in the final seed being dropped in the player store, therefore satisfying strategy one. Or, all pockets would be ineffective in satisfying strategy one, and strategy two, strategy three, strategy four, therefore the rightmost pit would be played. This would not require big changes in the constraints, but the propositions would have to include the strategies as listed above.