Team 27
Yoonsang Chong & Pranav Dhingra

We decided to use the codebase of Team 12 (Pranav's previous team). The two codebases shared similarities, but ultimately it proved to be more thorough and a more suitable foundation from which to continue work on the Santorini project.

Both of the projects use Python. However, Team 12's codebase (henceforth referred to as the chosen codebase, as opposed to the dropped codebase) employs the pytest library for convenient unit testing. This proved to be a great asset when moving into more complex testing regimes, as required by assignment 6, so it is one of the advantages that the chosen codebase had concerning the use of Python libraries.

Additionally, it was determined that both codebases had well-implemented functions that behaved as expected. However, the chosen codebase offered additional functionality that the dropped codebase did not. First, the Board class of the chosen codebase employed the use of unique functions set_board(), undo_move(), and undo_build(), which all work in unison to avoid the use of deepcopy when needing to manipulate a Board object, like when looking several moves ahead in the Strategy class. We believe that this implementation would lead to better memory management by the program (since heavy usage of deepcopy() was proving to be a bottleneck), and so this was another factor that led to us choosing the designated codebase. It also dropped the use of the Cell class, which proved unnecessary. As a plus, the JSON parser of the chosen codebase is more comprehensive, which while not necessary, seemed like a good asset to have in the back pocket.

Finally, design played the largest role in our decision. The chosen codebase is more modular, and effectively encapsulates its components and uses imports to both have less repetition, and keep parts of the program that should be separated, separated. Also, the code was better commented and more Pythonic, limiting the verbosity of the code itself and making it more accessible to outside viewers. It also had a contracts implemented wherever necessary in the form of custom exceptions which makes debugging easy and helps catch errors. We believe all this will come in handy during our codewalk, during which we will be able to refer to our functions' inputs, outputs, and contracts directly on the screen.