A short description of Mahjong for this to make more sense: Mahjong is a tile game where each player begins with a starting hand; Players draw one tile from the remaining deck and then discard one tile from hand by turn, and they keep drawing and discarding until either a player wins or the deck runs out. To win, a player need to have a hand that matches at least one specific pattern WITH a tile added to the hand: this tile can either be the one drawn on the player’s turn or one discarded on other players turn. So, “Can I win on my next draw/few draws” and “Can the other players win on this tile if I discard it” become the daunting question for every Mahjong player, every turn. This is why a “Mahjong Assistant” program can be really helpful for players to make decisions.

TP2 Update:

Instead of simulating the opponents’ hands, the program now will ONLY analyze the Open Melds and the board of the opponents to estimate risk factor. The reason to make this change is that the simulation can have thousands of combination, and there is no good reason to use simulation anyways.

The rest of the plan doesn’t change, but to hit MVP first, the following are some simplification:

Estimating win-rate and average score now only goes 1-round deep instead of 3

Red Dora is not counted

Self\_Wind is not counted

Richi is not counted

Project Description [5 pts]: The name of the term project and a short description of what it will be.

The name of my TP is “Mahjong Assistant”. The program is meant to be run alongside with a Mahjong video game or an irl Mahjong game. For now, the program only support three-player mode and “Richi Mahjong(Japanese Mahjong)” rule set. It contains a basic feature of tracking the deck and a few advanced add-on features of calculating win-rate, average winning score, and risk factor(lose-rate, average score deduction) on each tile in the player’s hand.

Structural Plan [5 pts]: A structural plan for how the finalized project will be organized in different functions, files and/or objects.

For MVP, the program runs without 112\_graphics and takes user input through terminal. Later on, graphics would be added, and I have designed the graphics layout already.

The main features use 3 classes: game, Player, score. There will also be a large folder containing a list of [yaku](https://en.wikipedia.org/wiki/Japanese_mahjong_yaku)(winning pattern) files.

For deck tracking, the program uses the game class. For all other features, every file and class will be used.

Algorithmic Plan [5 pts]: An algorithmic plan for how you will approach the trickiest part of the project. Be sure to clearly highlight which part(s) of your project are most complex.

Calculating win-rate (second most-complex): For a hand in a Mahjong Game, there are often multiple ways to achieve one or even multiple yakus(yakus add up). To calculate win-rate, the program simulates three turns and calculate the chance of winning in one turn, two turns, and three turns. For a bad hand, there may be only 4 or 5 outcomes; for a good hand, there may be more than a dozen of outcomes.

Calculating average winning score(most complex): this function will calculate the score of each possible outcome and calculate the average point expectation based on the change of each outcome happening(which is recorded after calculating win-rate).

Note: the most complex part of the program is to simulate the next few turns. It may not seem to be super difficult, but since there are many other restrictive rules in Japanese Mahjong like “sacred discard” and “abortive laws”, not all moves can be made, and the simulation of a turn can be vastly different depending on the simulation of the previous turn. Also, calculating score is very complex because you cannot simply add up the score of yakus, and a part of the score calculation(“fu”) is independent from the main features. You can see [this Wikipedia link](https://en.wikipedia.org/wiki/Japanese_mahjong_scoring_rules" \l "Scoring_table) for a better understanding.

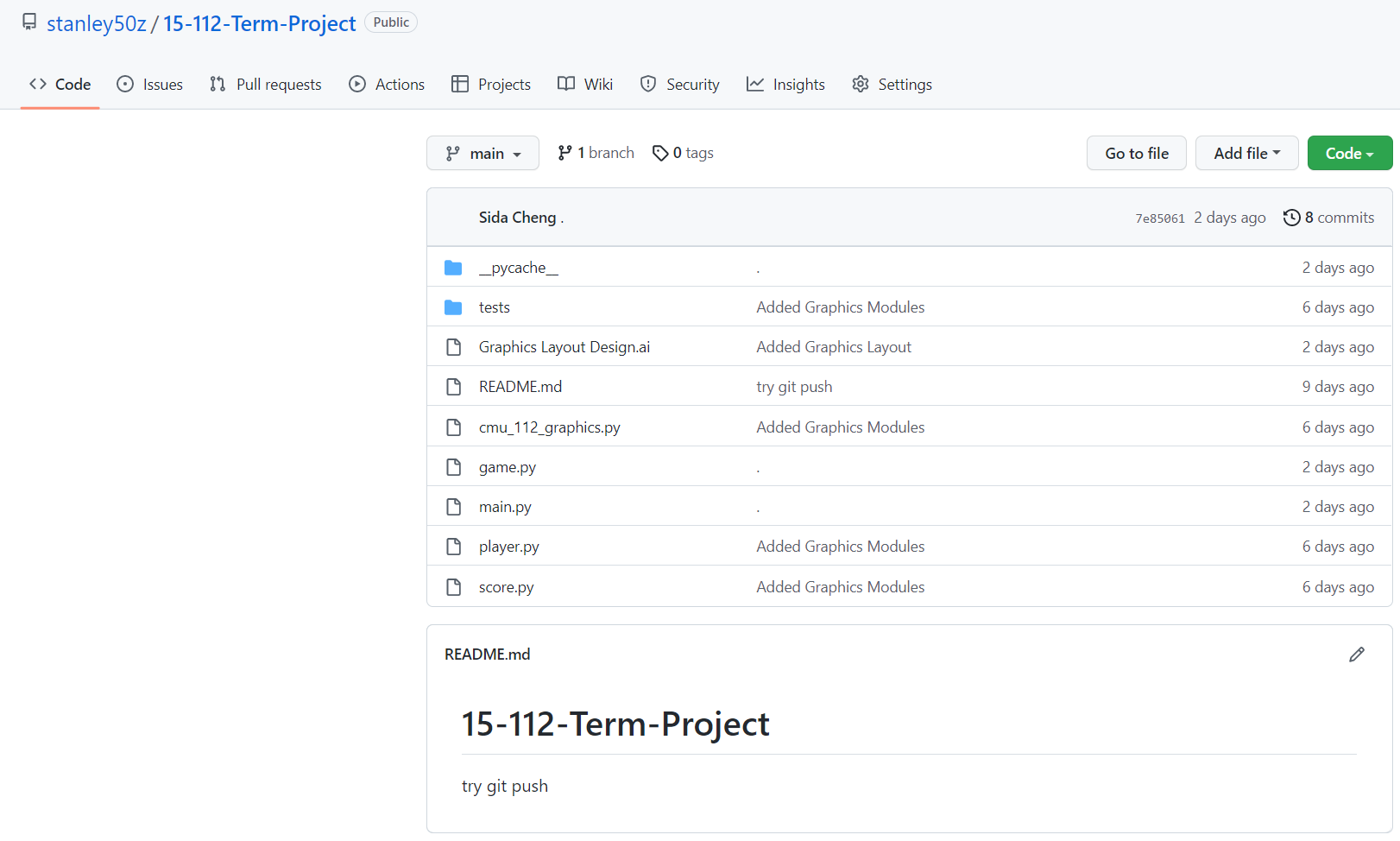
Version Control Plan [5 pts]: A short description and image demonstrating how you are using version control to back up your code. Notes:

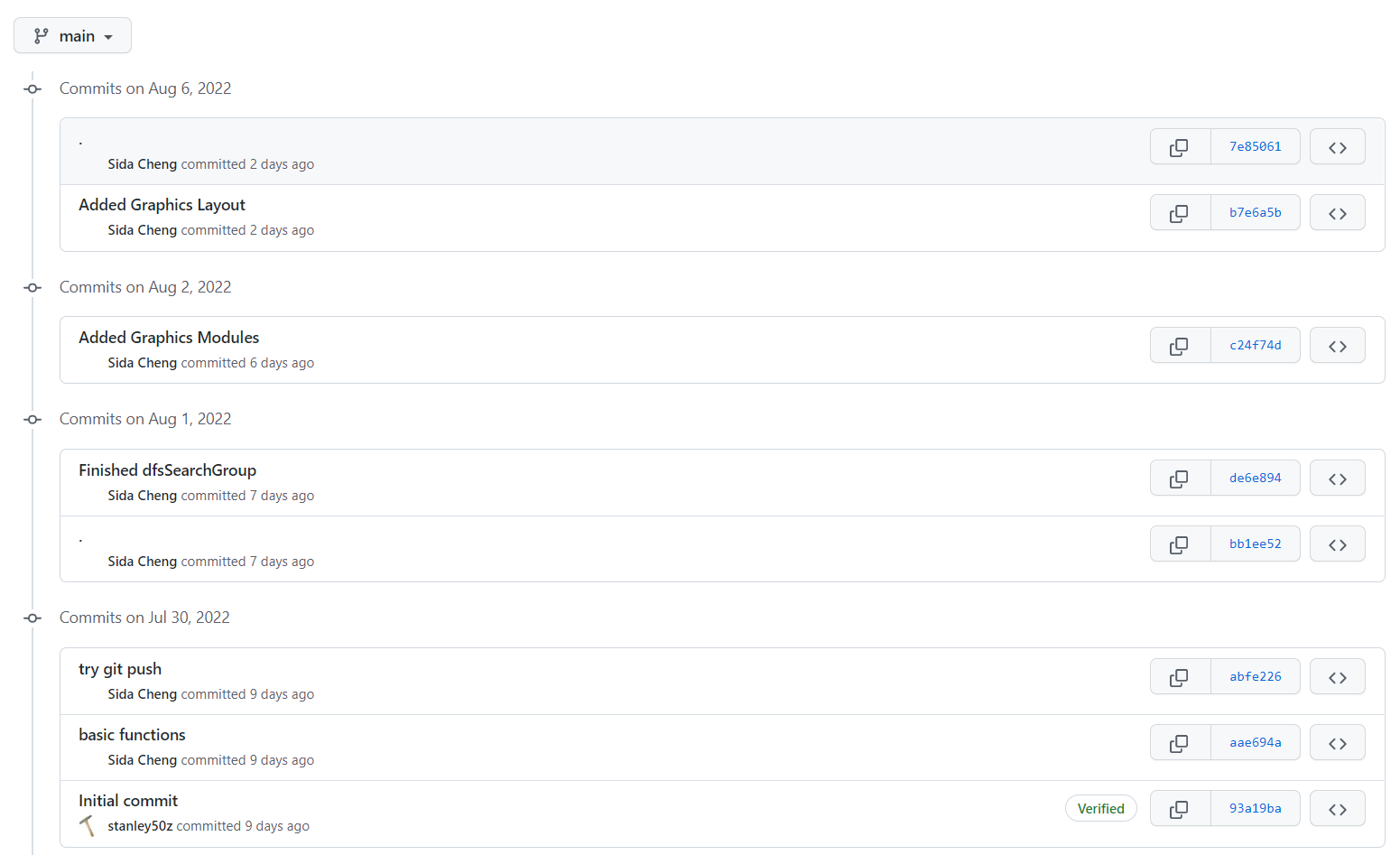
You must back up your code somehow!!!

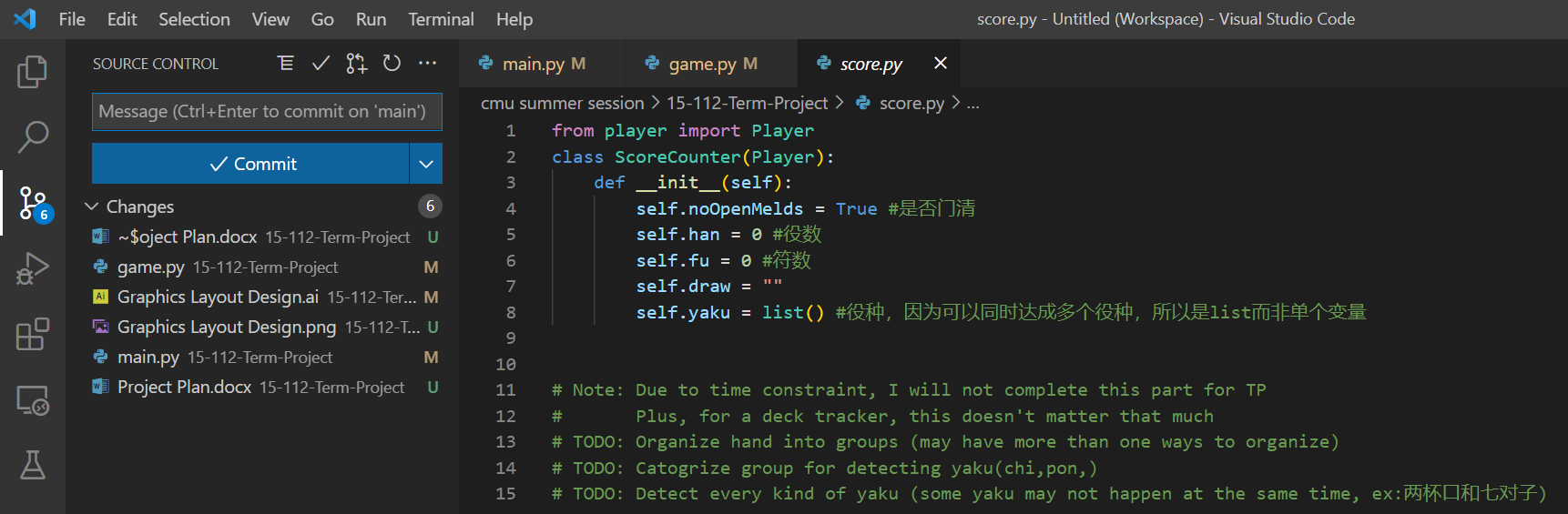
Your backups must not be on your computer (ideally, store them in the cloud)

Google Drive is a common option, or GitHub if you are comfortable with it.

I use GitHub. Here’s the [link](https://github.com/stanley50z/15-112-Term-Project). Here are some screenshot to prove that I am using it well.







Another note on the Algorithm Complexity of my TP:

I have seen all 8 other Mahjong Games presented in the Gallery; none of them do “Richi” rules(which has many additional rules); and none of them do score calculation(which is also more complex in “Richi” rules). My TP would be far more algorithmically complex than any other Mahjong TPs in the past. After my initial TP meeting, I realize that I may have set the standard of MVP too high. It is very likely that even if I hit the MVP on TP2, some of the features may not be statistically accurate, but it should still have a decent complexity.