Design:

1. Describe your strategy for guessing

The key algorithm for this program is selecting the probable correct number from the whole number list, and then building a new list to replace the old number list.

Firstly, we build a list including all the number from 100 to 999 without repeating digits. To do it, we def a function to check a number whether has repeating digits.

Only the number without repeating digits that count will be true. Then we use for loop to append all the correct numbers into the whole number list:

Then import random module, let the computer to randomly give a number from the number list:

The next step, we want to select the probable number according to the information the user gives us, there are 2 steps. The first step according to the information about the correct number. For example, for ‘123’, if the correct number is 1, then we want to find the number in the list that contains ‘1’ but not contain ‘2’ and ‘3’ or contains ‘2’ but not contain ‘1’ and ‘3’ or contains ‘3’ but not contain ‘2’ and ‘1’. The principle is the same for 2 correct digits. We use the set function, to calculate the same numbers of both the Guess number and the number picked from the number list. Only the number whose length of the same parts is equal to the given correct numbers information will be append to the next number list. We then finished the first selecting.

The next step we use the information of the exact digits to build a new number list (The numbers are all selected from the previous number list) How to do this? For example, if the Guess number is ‘123’, and the exact number is 1. Then we choose the number from the number list that the number begins with ‘1’ or end with ‘3’ or ‘2’ is in the middle. For the situation that exact number is 2, it will be a little complicated, but they are essentially the same.

Then we make all the previous steps in a infinity loop. Only when the Correct Number is equal 3 will the loop break.

When the correct digits reach 3, the thing became much easier. There are 3 circumstances. First, exact digits is 3, we can output the result directly. Second, exact digits is 1. We need to change the positions of the 2 digits inside the number. There are 3 possible situations, and from one of them we can get the correct answer. Therefore we try 3 times. Third, exact digits is 0. It’s much easier. For example, Guess Number is ‘123’ the exact digit is 0. Then there are only 2 choices, ‘231’ and ‘312’. We try twice and we are certain to get the correct number.

1. Describe the core data structures used for tracking the guess history and feedback

Firstly, we build a blank list, which is output\_list. In every loop for guessing number, the guessing situation will be stored in it. Then we use for loop to output all the guessing situation stored in the output list.

1. Functions:

check\_number(a): check whether the number has repeated digits.

check\_intersection\_1(): select numbers from the old list according to correct digits.

check\_intersection\_2(): select numbers from the old list according to correct digits.

combine(m,n,i): a function inside check\_intersection\_2() to help simplify it.

Variables:

Number\_list: the list containing all the probable numbers.

output\_list: the list containing all the guessing situations.

output\_record: the record of guessing situation in every loop.

Secret\_Number: the number you want computer to guess.

count: counter to store the guessing times.

Guess\_Number: the number randomly picked from the Number\_list that computer use to guess for the number.

Correct\_Digits: the number that the user input for the amount of correct digits.

Exact\_Digits: the number that the user input for the amount of exact digits.

Program Flow:

1. Build a Number list containing all the probable digits.
2. Ask the users to input the information of the correct digits and exact digits. If the input is not logical, let the user enter again.
3. Use 2 check function to select all the probable digits from the old number list and build a new number list.
4. Use while loop to repeat step 2, until the correct digits reach 3.
5. When correct digits reaches 3, there are 3 situations. If exact digits are 3, then output the result. If exact digit is 1, then try to change 2 number’s position. It will success at most 3 times. If exact digit is 0, then try to move every number’s position ahead (the first moves to the end). It will success at most twice.
6. When exact digits reach 3, output the result, and ask the user whether want to play again.

Test Strategy:

1. How to verify that your program works as per “Scope” – create a test plan
2. Firstly, list all the demands in the ‘Scope’, and tab them.
3. Think in mind that at what situations will these demands needed to be met.
4. Find these situations in your codes, and mark the number of the corresponding demands.
5. Use special data to get every marked code and test whether they meet the demands.
6. Debug is finished
7. Describe different plans for different scenarios
8. Whether the program can output the ‘secret number’ in 20 steps:

Try to run the program correctly 10 times, if it can all output the correct results, the program should be OK in this scenario. If sometimes it can’t output the correct results, trace back to the codes according the output list.

1. Identify whether the input of the users is correct:

Find all the codes for users to input the correct and exact digits and mark them. Then list all the ways of wrong inputs. Then design different data to get to all the corresponding codes and test all the wrong inputs, in order to find whether the program can react properly.

1. Whether the program can output feedback history and guess number with correct format and counter well:

Try to run the program correctly 5 times, if it can all output all the things mentioned before correctly, the program should be OK in this scenario. If sometimes it can’t output the correct results, trace back to the codes according the output list.

1. Whether the program can provide proper information to let the user restart or quit:

Try to run the program correctly 10 times, 10 times should all get to the secret number. 5 times enter ‘Y’, 5times enter ‘N’, to check whether the program can restart or exit successfully. If all successfully, the program should be OK in this scenario. If sometimes it fails, trace back to the codes according the output list.