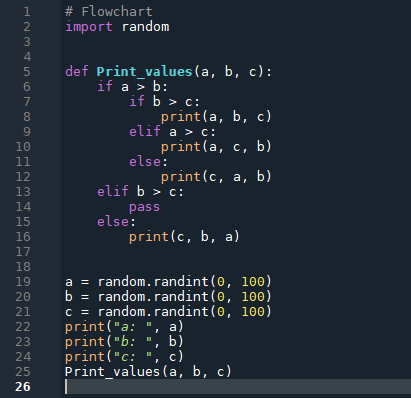
PS1 Report

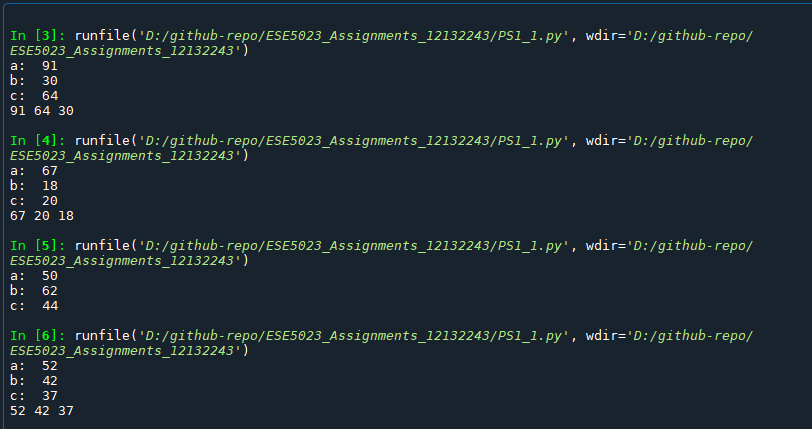
Name:左小幸 SID:12132243

**PS1\_1: Flowchart**

In this case, I created a Print\_values function with parameters a, b, and c. In the function, I use nesting of **if** statements to implement the judgment shown in the flowchart.

Here is the code and some results:

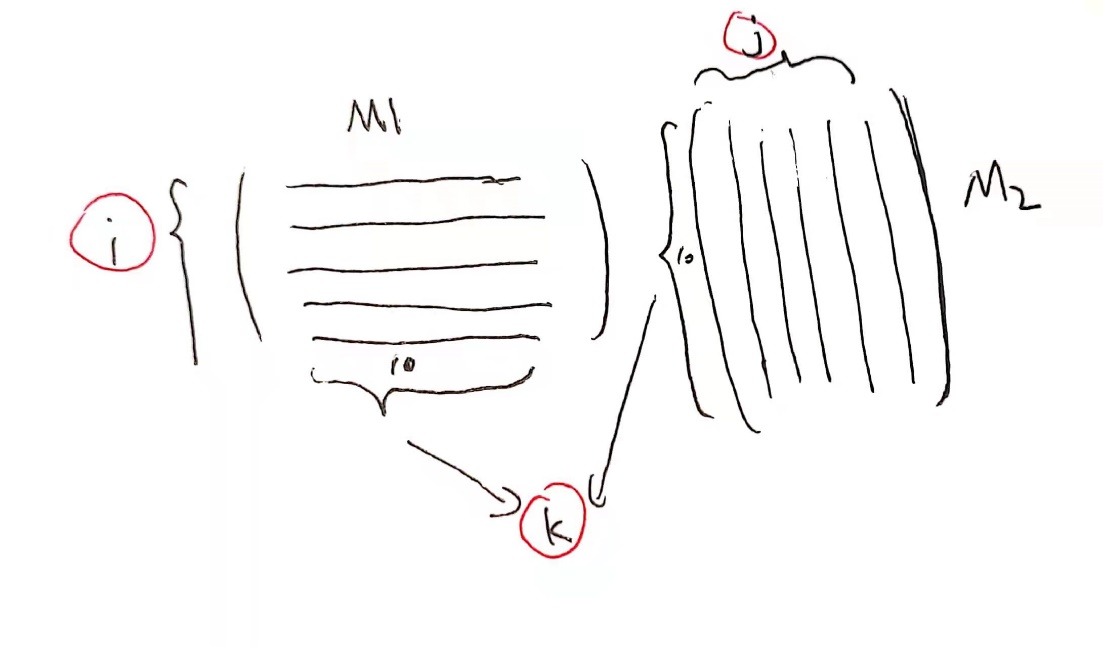




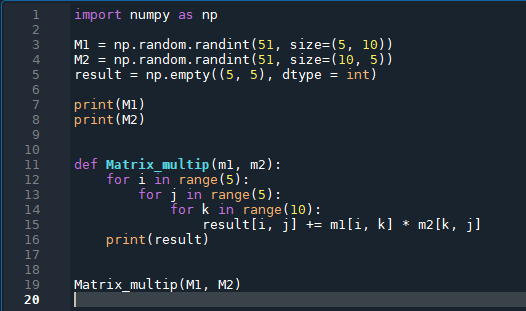
**PS2:** **Matrix multiplication**

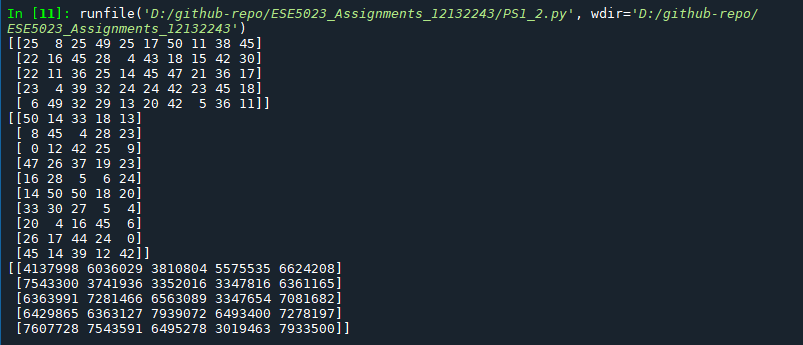
2.1: The M1 and M2 matrices are created using **np.random.randint()** function, both are filled with random integers from 0 and 50. The first parameter in the function specifies the range of values in the matrix, and the second parameter specifies the size of the matrix.

2.2: I've created a Matrix\_multip() function that uses a for loop to multiply matrices. Where **i** represents the 5 rows of matrix M1, **j** represents the 5 columns of matrix M2, and **k** represents the 10 elements in each row of M1 multiplied by the 10 corresponding elements in each column of M2.



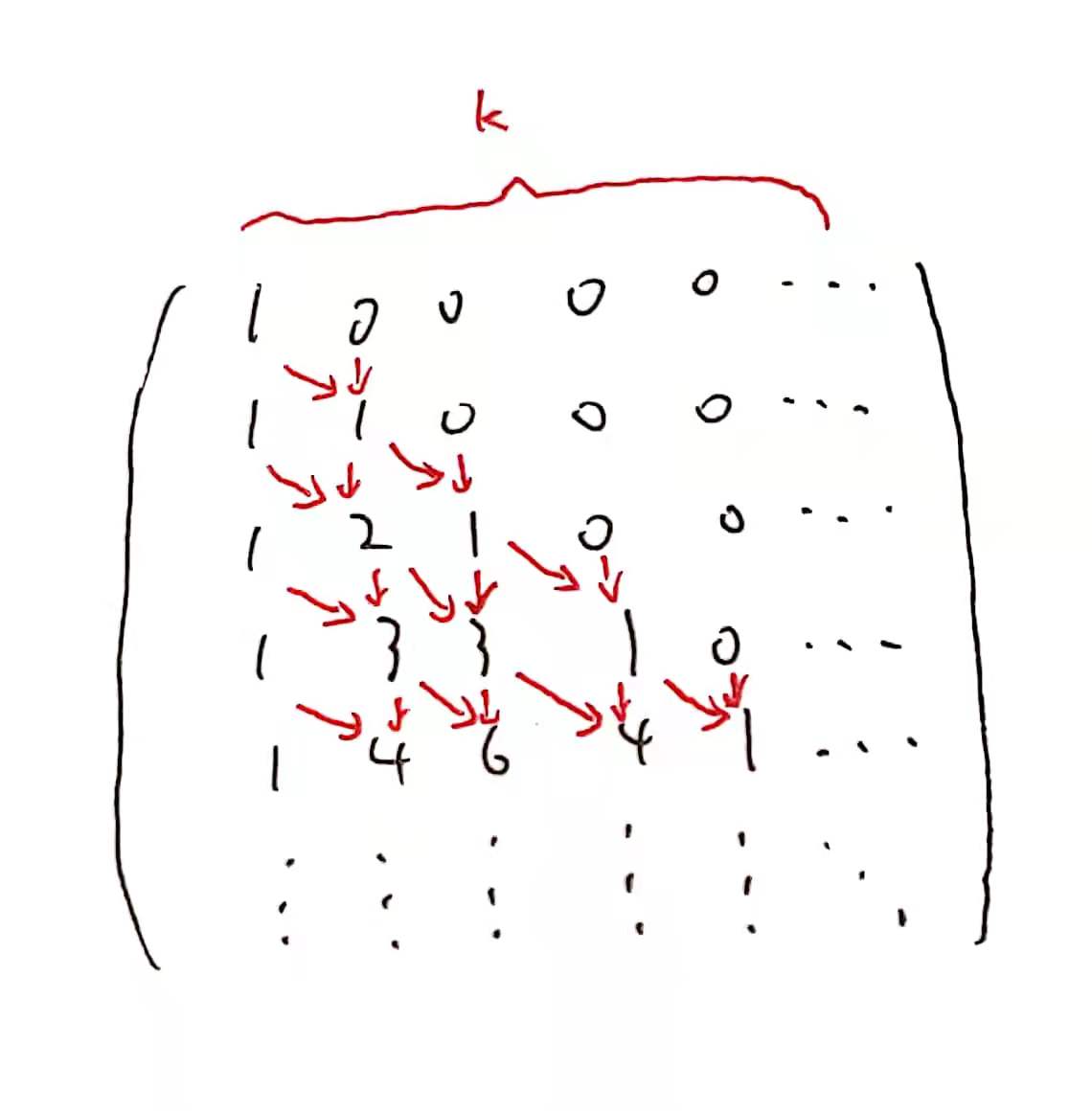
Here is the code and a result:





**PS3: Pascal triangle**

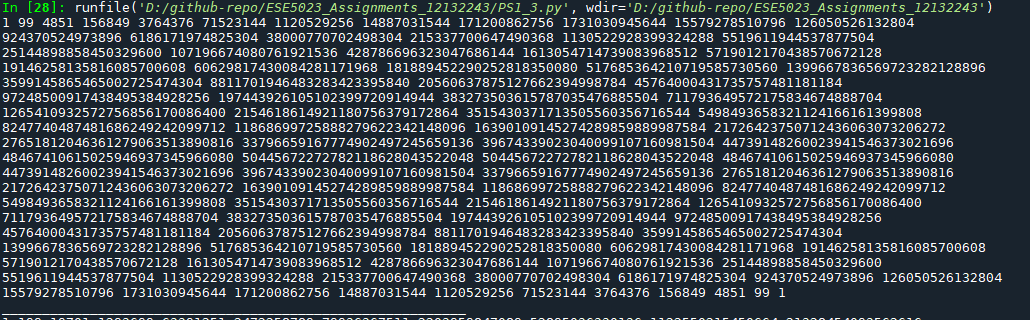
In this case, I've created a function Pascal\_triangle(k) that takes an argument k. In the function, a square matrix of size k will be created first, and then the values of PASCAL triangles will be generated line by line and stored in the square matrix according to the rule, as shown in the following figure.



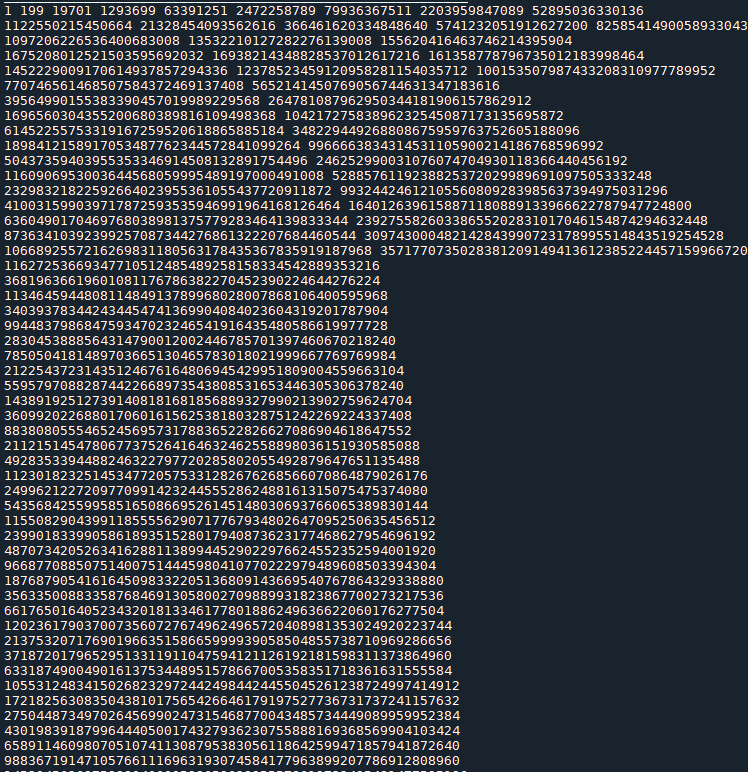
Here is the code and a result:



Pascal\_triangle(100)



Pascal\_triangle(200)

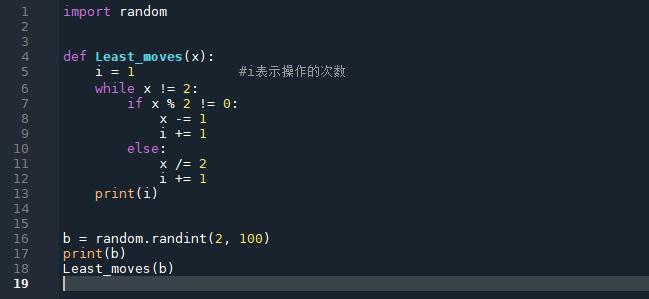


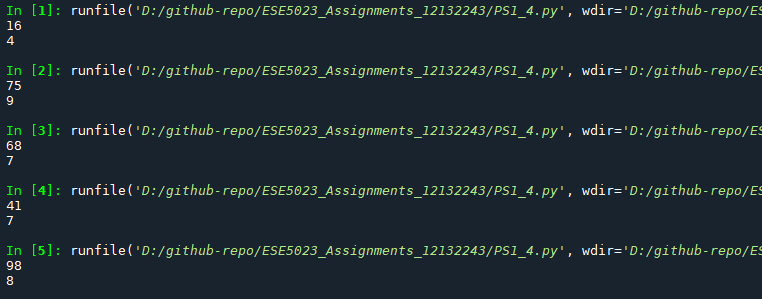
(The complete results can be viewed by running the code.)

**PS4: Add or double**

In this case, I did the reverse. That is, if I have an integer between 1 and 100, I can divide by 2(remainder must be 0) or subtract by 1 each time to find the smallest number of operations so that the final result is 1.

Here is the code and some results:

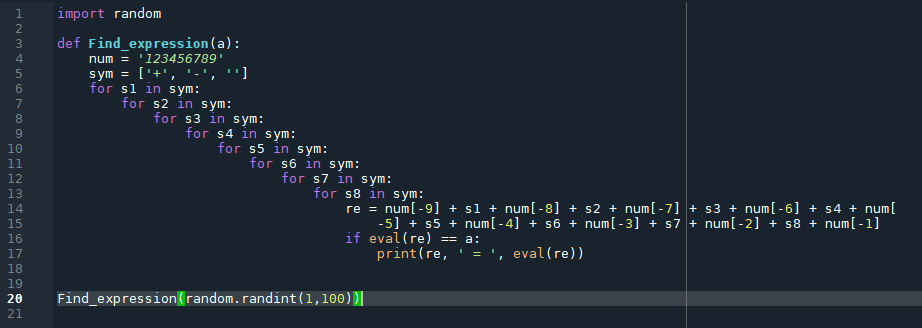


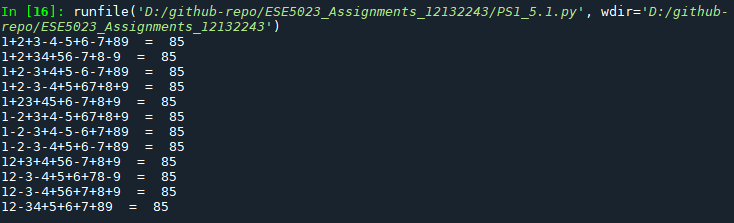
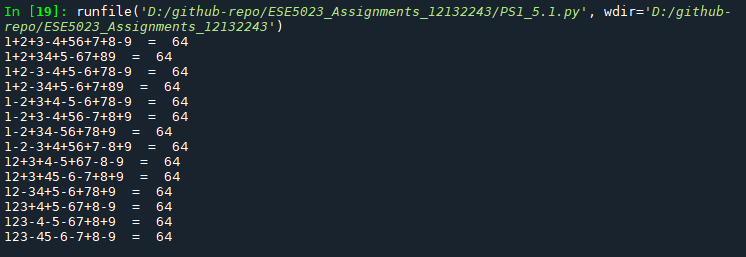


**PS5: Dynamic programming**

5.1: In this case, I've created a function Find\_expression(a) that takes argument “a”. The nine integers from 1 to 9 are sequentially stored in a string "num", and the list "sym" is used to store three possible symbols between each two numbers. A for loop is then used to list all possible cases, and an eval function is used to evaluate the result of the string operator re, and finally to print the result equal to a.

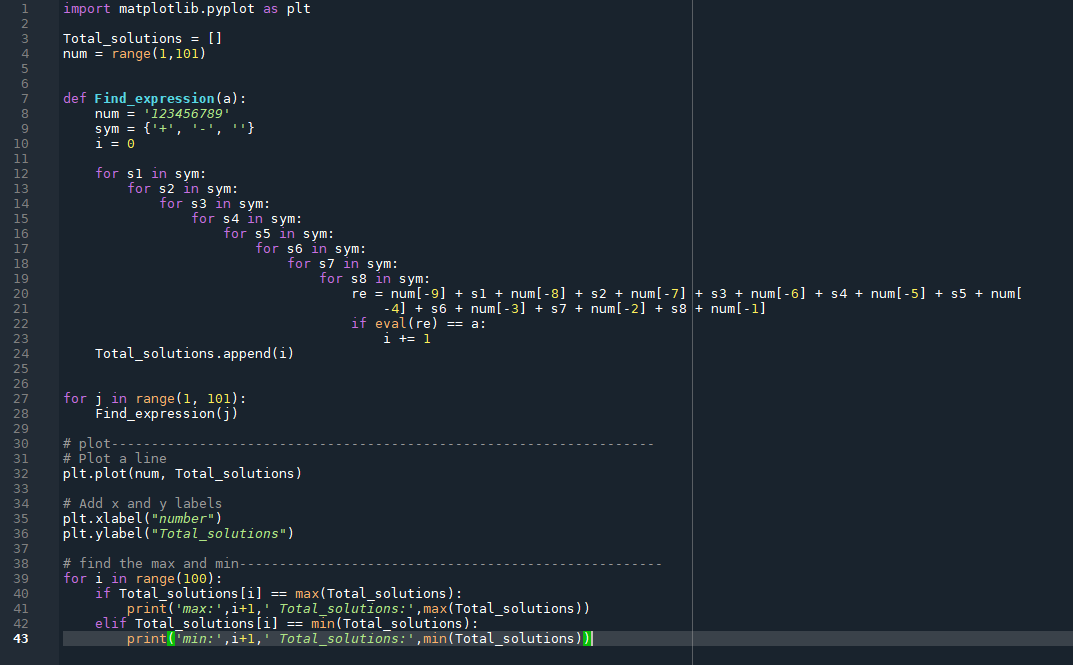
Here is the code and some results:

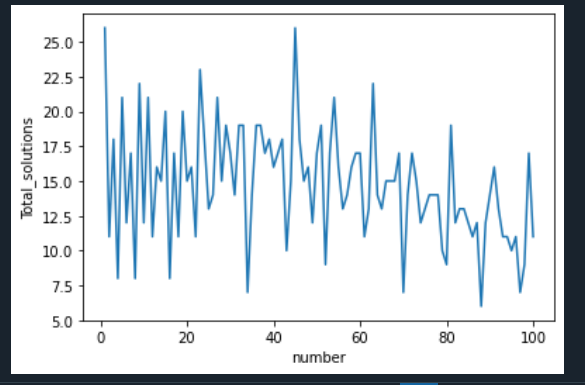


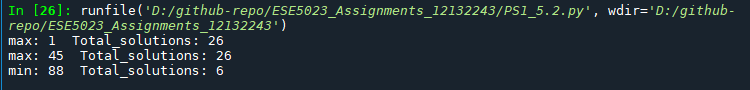
 

5.2: In this case, we only need to change the 5.1 code slightly. Use a list to store solutions with integers 1-100, chart the results, and find the best value.

Here is the code and some results:







**Reference:**

1.<https://www.cnblogs.com/oNull/p/13472480.html#random_1> help me know how to create random numbers in python. **In in problem set 1.**

2.<https://www.runoob.com/numpy/numpy-array-creation.html> and <https://www.cnblogs.com/pipiyan/p/10445948.html> helped me understand how to create matrices using the Numpy library, **in problem set2**.

3.https://blog.csdn.net/weixin\_34273481/article/details/92068609 tells me how do I create a 0 matrix, **in problem set3.**

4. This is where I found the use of the eval function(<https://www.runoob.com/python/python-func-eval.html>) , which helps me evaluate string expressions, **in problem set5**.

5. What have I learned from this website(https://zhu-group.github.io/ese5023/Lab\_02.html) about how to chart data, **in problem set5**.