1、Flowchart

```
def print_values(a, b, c):
    if a > b:
        if b > c:
            print(a+","+b+","+c)
        else:
            if a > c:
                print(a+","+c+","+b)
            else:
                print(c+","+a+","+b)
    else:
        if b > c:
            print()
        else:
            print(c+","+b+","+a)


a_value = input("Please enter the a value: ")
b_value = input("Please enter the b value: ")
c_value = input("Please enter the c value: ")

print_values(a_value, b_value, c_value)
```

2、Matrix multiplication

```
import numpy as np


def Matrix_multip(M1, M2):
    new_matrix = []
    for i in range(len(M1)):
        row = []
        for j in range(M2.shape[1]):
            result = 0
            for k in range(len(M1[i])):
                result += M1[i][k]*M2[k][j]
            row.append(result)
        new_matrix.append(row)
    print(new_matrix)

m1 = np.array(range(0, 50)).reshape(5, 10)
m2 = np.array(range(0, 50)).reshape(10, 5)
```

```
Matrix_multip(m1,m2)
```

3、Pascal triangle

```
def pascal_triangle(k):
    line = [1]
    for i in range(k):
        line.append(int(line[i]*(k-i)/(i+1)))
    return line

k = int(input("Please enter the k value: "))
print(pascal_triangle(k))
```

4. Add or double

```
def Least_moves(n):
    if n%2 == 0:
        print(int(n/2))
    else:
        print(int((n+1)/2))

k=int(input("Please select a number from 1 to 100: "))
Least_moves(k)
```

5. Dynamic programming

```
from functools import reduce

operators = {
    1: '+',
    2: '-',
    0: ''
}

bases = ['1', '2', '3', '4', '5', '6', '7', '8', '9']


def find_expression(num):
    arr = []
    for i in range(8):
        i = 7-i
        arr.append(num//(3**i))
        num -= (num//(3**i))*(3**i)
```

```
    arr = map(lambda x: operators[x], arr)

    formula = reduce(lambda x, y: x+y, zip(bases, arr))
    formula = list(formula)
    formula.append('9')
    formula = ''.join(formula)

    result = eval(formula)
    return result, formula


if __name__ == '__main__':
    total = 3**8
    n = 0
    Total_solutions = []
    for j in range(total):
        result, formula = find_expression(j)
        if result == 50:
            print(formula+" = 50")

    for k in range(1, 100):
        for l in range(total):
            result, formula = find_expression(l)
            if result == k:
                n += 1
        Total_solutions.append(n)
        n = 0

print(Total_solutions)
print(Total_solutions.index(max(Total_solutions))+1)
print(Total_solutions.index(min(Total_solutions))+1)
```