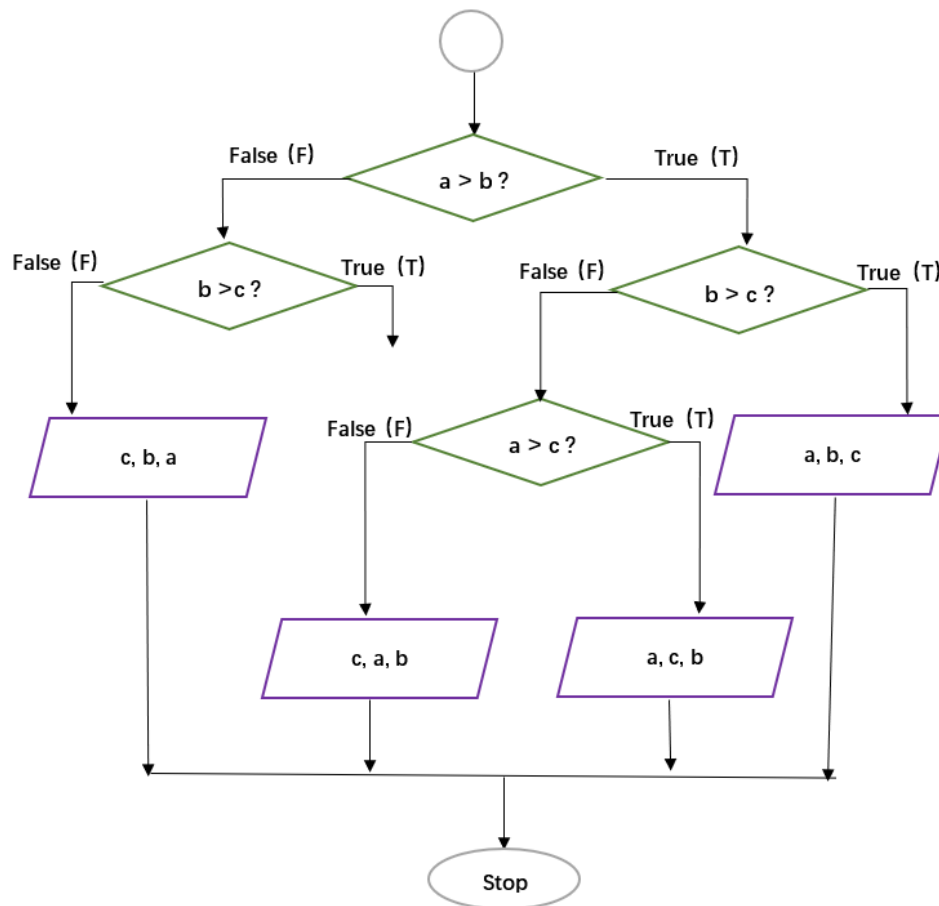


1. Flowchart

[10 points] Write a function `Print_values` with arguments `a`, `b`, and `c` to reflect the following flowchart. Here the purple parallelogram operator is to print values in the given order. Report your output with some random `a`, `b`, and `c` values.



```
PS1_1.py x PS1_2.py x PS1_3.py x PS1_4.py x PS1_5.py x
13         print(a,c,b)
14     else:
15         print(c,a,b)
16     elif b > c:
17         print(c,a,b)
18     else:
19         print(c,b,a)
20
21 Print_values(1,3,2)
22
23
```

```
PS1_1 x
D:\anaconda3\python.exe D:/ESE5023/PS1_1.py
2 1 3
Process finished with exit code 0
```

2. Matrix multiplication

2.1 [5 points] Make two matrices M1 (5 rows and 10 columns) and M2 (10 rows and 5 columns); both are filled with random integers from 0 and 50.

```
PS1_1.py x PS1_2.py x PS1_3.py x PS1_4.py x PS1_5.py x
4
5 @author: TianYF
6 """
7 # 2.1
8 import numpy as np
9
10 M1 = np.mat(np.random.randint(50, size=(5, 10)))
11 M2 = np.mat(np.random.randint(50, size=(10, 5)))
12 print(M1)
13 print(M2)
14
```

```
PS1_2 x
D:\anaconda3\python.exe D:/ESE5023/PS1_2.py
[[ 1 46 36  1 21 35 25  0  5  3]
 [ 4  9 34 44 28  1 28  4 23  2]
 [28 29 15 20 19 26 20 29  4  1]
 [35 21 13 11 39 47 43 20 15 17]
 [11  2 40 19 19 26 32 13 40 15]]
[[18 36 23  4  4]
 [43 35 22  1 26]
 [32  5 45 43 42]
 [19 28 36 23 12]
 [37 40 26 13 31]
 [39 14 45 48 39]
 [38 41 14 17 30]
 [18 34 20 27 25]
 [39 45 46 21 10]
 [11 36 29 41 29]]

Process finished with exit code 0
|
```

2.2 [10 points] Write a function `Matrix_multip` to do matrix multiplication, i.e., $M1 * M2$. Here you are **ONLY** allowed to use `for` loop, `*` operator, and `+` operator.

```
# 2.2
def Matrix_multip(M1,M2):
    r1, c1 = M1.shape
    r2, c2 = M2.shape
    result = np.zeros((r1, c2))
    for i in range(r1):
        for j in range(c2):
            for k in range(c1):
                result[i][j] += M1[i][k] * M2[k][j]
    print(result)
    return result

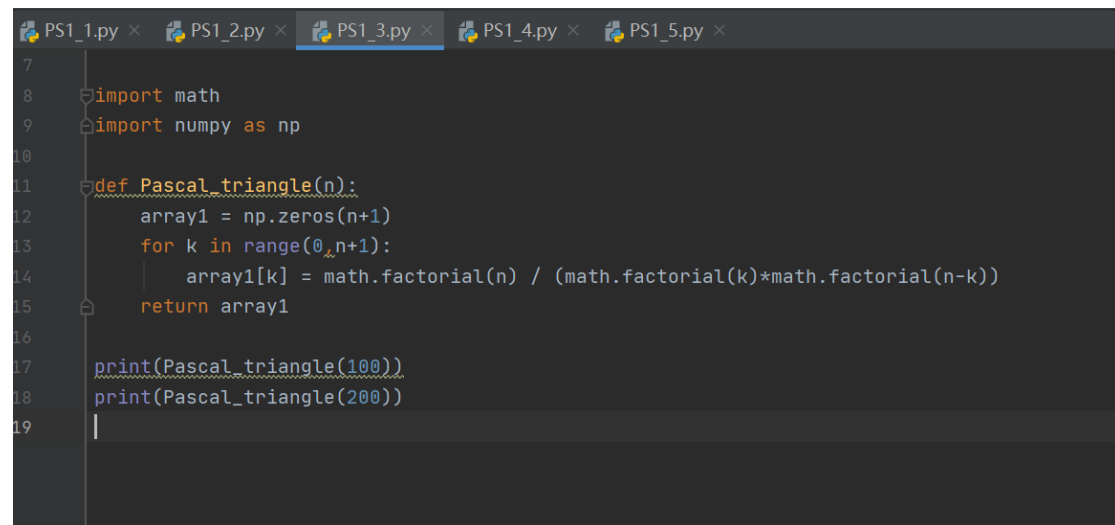
Matrix_multip(M1,M2)
```

```
[[5560. 4958. 4754. 5144. 5203.]
 [9067. 6148. 5914. 7416. 6440.]
 [7503. 5783. 5600. 6709. 6907.]
 [5898. 6880. 6207. 7345. 7391.]
 [4928. 3837. 2681. 4807. 3629.]]

Process finished with exit code 0
```

3. Pascal triangle

[20 points] One of the most interesting number patterns is [Pascal's triangle](#) (named after Blaise Pascal). Write a function `Pascal_triangle` with an argument `k` to print the k^{th} line of the Pascal triangle. Report `Pascal_triangle(100)` and `Pascal_triangle(200)`.



```
PS1_1.py × PS1_2.py × PS1_3.py × PS1_4.py × PS1_5.py ×
7
8 import math
9 import numpy as np
10
11 def Pascal_triangle(n):
12     array1 = np.zeros(n+1)
13     for k in range(0, n+1):
14         array1[k] = math.factorial(n) / (math.factorial(k)*math.factorial(n-k))
15     return array1
16
17 print(Pascal_triangle(100))
18 print(Pascal_triangle(200))
19 |
```

```
PS1_3 x
D:\anaconda3\python.exe D:/ESE5023/PS1_3.py
[1.00000000e+00 1.00000000e+02 4.95000000e+03 1.61700000e+05
 3.92122500e+06 7.52875200e+07 1.19205240e+09 1.60075608e+10
 1.86087894e+11 1.90223181e+12 1.73103095e+13 1.41629805e+14
 1.05042105e+15 7.11054250e+15 4.41869427e+16 2.53338471e+17
 1.34586063e+18 6.65013487e+18 3.06645108e+19 1.32341573e+20
 5.35983370e+20 2.04184141e+21 7.33206689e+21 2.48652703e+22
 7.97760756e+22 2.42519270e+23 6.99574817e+23 1.91735320e+24
 4.99881370e+24 1.24108478e+25 2.93723398e+25 6.63246383e+25
 1.43012501e+26 2.94692427e+26 5.80717430e+26 1.09506715e+27
 1.97720458e+27 3.42002955e+27 5.67004899e+27 9.01392403e+27
 1.37462341e+28 2.01164402e+28 2.82588089e+28 3.81165329e+28
 4.93782358e+28 6.14484712e+28 7.34709982e+28 8.44134873e+28
 9.32065589e+28 9.89130829e+28 1.00891345e+29 9.89130829e+28
 9.32065589e+28 8.44134873e+28 7.34709982e+28 6.14484712e+28
 4.93782358e+28 3.81165329e+28 2.82588089e+28 2.01164402e+28
 1.37462341e+28 9.01392403e+27 5.67004899e+27 3.42002955e+27
 1.97720458e+27 1.09506715e+27 5.80717430e+26 2.94692427e+26
 1.43012501e+26 6.63246383e+25 2.93723398e+25 1.24108478e+25
 4.99881370e+24 1.91735320e+24 6.99574817e+23 2.42519270e+23
 7.97760756e+22 2.48652703e+22 7.33206689e+21 2.04184141e+21
 5.35983370e+20 1.32341573e+20 3.06645108e+19 6.65013487e+18
 1.34586063e+18 2.53338471e+17 4.41869427e+16 7.11054250e+15
 1.05042105e+15 1.41629805e+14 1.73103095e+13 1.90223181e+12
 1.86087894e+11 1.60075608e+10 1.19205240e+09 7.52875200e+07]
```

4. Add or double

[20 points] If you start with 1 RMB and, with each move, you can either double your money or add another 1 RMB, what is the smallest number of moves you have to make to get to exactly x RMB? Here x is an integer randomly selected from 1 to 100. Write a function `Least_moves` to print your results. For example, `Least_moves(2)` should print 1, and `Least_moves(5)` should print 3.

```
PS1_1.py × PS1_2.py × PS1_3.py × PS1_4.py × PS1_5.py ×
14         list_i = list_i_double + list_i_add
15         L.append(list_i)
16         for i, nested in enumerate(L):
17             if x in nested:
18                 print(i)
19                 break
20             else:
21                 continue
22             break
23     elif x == 2:
24         print('1')
25     else:
26         print('0')
27
28
29     Least_moves(5)
```

```
PS1_4 ×
D:\anaconda3\python.exe D:/ESE5023/PS1_4.py
3
Process finished with exit code 0
```

5. Dynamic programming

Insert + or - operation anywhere between the digits 123456789 in a way that the expression evaluates to an integer number. You may join digits together to form a bigger number. However, the digits must stay in the original order.

5.1 [30 points] Write a function `Find_expression`, which should be able to print every possible solution that makes the expression evaluate to a random integer from 1 to 100. For example, `Find_expression(50)` should print lines include:

$$1-2+34+5+6+7+8-9=50 \quad 1-2+34+5+6+7+8-9=50$$

and

$$1+2+34-56+78-9=50 \quad 1+2+34-56+78-9=50$$

5.2 [5 points] Count the total number of suitable solutions for any integer i from 1 to 100, assign the count to a list called `Total_solutions`. Plot the list `Total_solutions`, so which number(s) yields the maximum and minimum of `Total_solutions`?

[illegible]