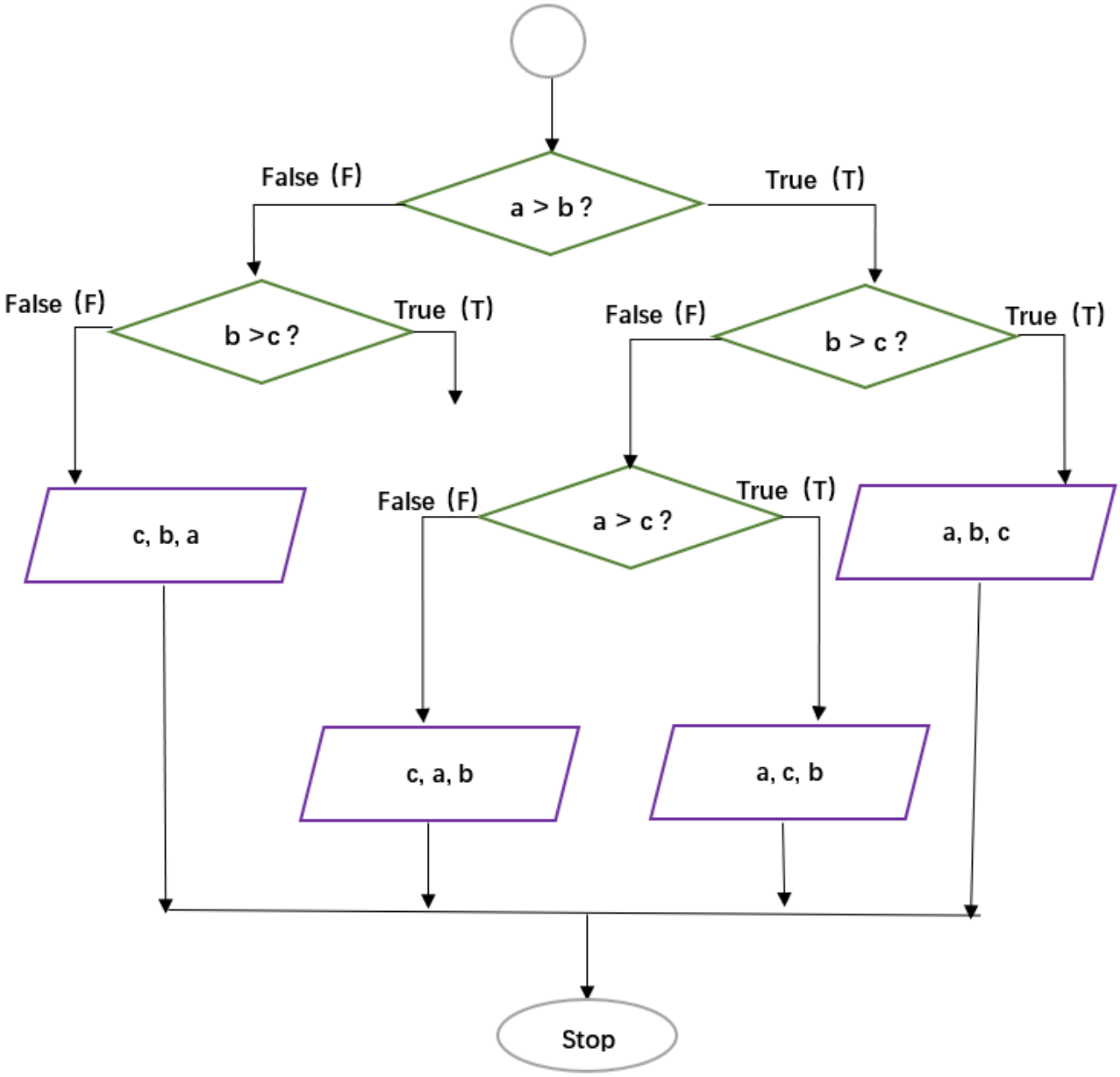
1. Flowchart

[10 points] Write a function Print_values with arguments a, b, and c to reflect the following flowchart. Here the purple parallelogram operator on a list [x, y, z] is to compute and print x+y-10z. Try your output with some random a, b, and c values. Report your output when a = 10, b = 5, c = 1.



In [1]:

```
def Print_values(a,b,c):
    if(a>b):
        if(b>c):
            arg=[a,b,c]
        else:
            if(a>c):
                arg=[a,c,b]
            else:.
                arg=[c,a,b]
    else:
        if(b>c):
            print("NaN")        #Next there are no branches and the NAN is printed.
            return
        else:
            arg=[c,b,a]
    res=arg[0]+arg[1]-10*arg[2]
    return res

x=int(input("please input x :"))
y=int(input("please input y :"))
z=int(input("please input z :"))
Print_values(x,y,z)
```

```
please input x :10
please input y :5
please input z :1
```

Out[1]:

```
5
```

2. Continuous celing function

[10 points] Given a list with N positive integers. For every element x of the list, find the value of continuous ceiling function defined as F(x) = F(ceil(x/3)) + 2x, where F(1) = 1

In [32]:

```
import math
def func(x) :
    if x==1 :
        res=1
    else:
        res=func(math.ceil(x/3))+2*x
    return res

def show(lis,N):                    #For every element x of the list, find the value of continuous cei
    for i in range(n):
        print(func(lis[i]),end=" ")

lis=[1,2,3,4,5,6]
N=len(lis)
print("input list [1,2,3,4,5,6]")
print("result: ")
show(lis,N)
```

```
input list [1,2,3,4,5,6]
result:
1 5 7 13 15 17
```

3. Dice rolling

3.1 [15 points] Given 10 dice each with 6 faces, numbered from 1 to 6. Write a function Find_number_of_ways to find the number of ways to get sum x, defined as the sum of values on each face when all the dice are thrown.

```
Analyse:The problem can be expressed mathematically,f(10,x)=f(9,x-1)+f(9,x-2)+f(9,x-3)+f(9,x-
4)+f(9,x-5)+f(9,x-6).Solve the problem by setting up a recursive loop.
```

In [33]:

```
N=10
def Find_number_of_ways(N,x):
    res = 0
    if x>6*N or x<N :
        return 0
    elif N==1 :                     #A dice for any x has only 1 and 0 possibilities.
        return 1
    else:
        for i in range(1, 7):       #Recursive loop.
            res = int((res + Find_number_of_ways(N-1, x-i)))
    return res
x=int(input("Please input x: "))    #Input the sum of values on each face when all the dice are thro
Find_number_of_ways(N,x)
```

```
Please input x: 50
```

Out[33]:

```
85228
```

3.2 [5 points] Count the number of ways for any x from 10 to 60, assign the number of ways to a list called Number_of_ways, so which x yields the maximum of Number_of_ways?

In [6]:

```
Number_of_ways=[]
for x in range(10,61):
    Number_of_ways.append(Find_number_of_ways(10,x)) #The running time is about 20 seconds
print(Number_of_ways)
res=10+Number_of_ways.index(max(Number_of_ways))     #Find the maximum index and find the correspond
res
```

```
[1, 10, 55, 220, 715, 2002, 4995, 11340, 23760, 46420, 85228, 147940, 243925, 38347
0, 576565, 831204, 1151370, 1535040, 1972630, 2446300, 2930455, 3393610, 3801535, 41
21260, 4325310, 4395456, 4325310, 4121260, 3801535, 3393610, 2930455, 2446300, 19726
30, 1535040, 1151370, 831204, 576565, 383470, 243925, 147940, 85228, 46420, 23760, 1
1340, 4995, 2002, 715, 220, 55, 10, 1]
```

Out[6]:

```
35
```

4. Dynamic programming

4.1 [5 points] Write a function Random_integer to fill an array of N elements by randomly selecting integers from 0 to 10.

In [36]:

```
import numpy as np
def Random_integer(N):
    arr=np.random.randint(0,10,N) #Generates N random integers ranging from 0 to 10
    return arr
N=int(input("Please input N:"))
res=Random_integer(N)
print("Output:")
print(res)
```

```
Please input N:5
Output:
[5 0 2 1 7]
```

4.2 [15 points] Write a function Sum_averages to compute the sum of the average of all subsets of the array. For example, given an array of [1, 2, 3], you Sum_averages function should compute the sum of: average of [1], average of [2], average of [3], average of [1, 2], average of [1, 3], average of [2, 3], and average of [1, 2, 3].

```
Analysis: If an list 'res' with N elements is input, the sum of the means of all subsets can be
viewed as the sum of N fractions, the denominator is N, and the numerator is the sum of the
corresponding binomial coefficients times the values of all the elements of 'res',respectively.
```

In [40]:

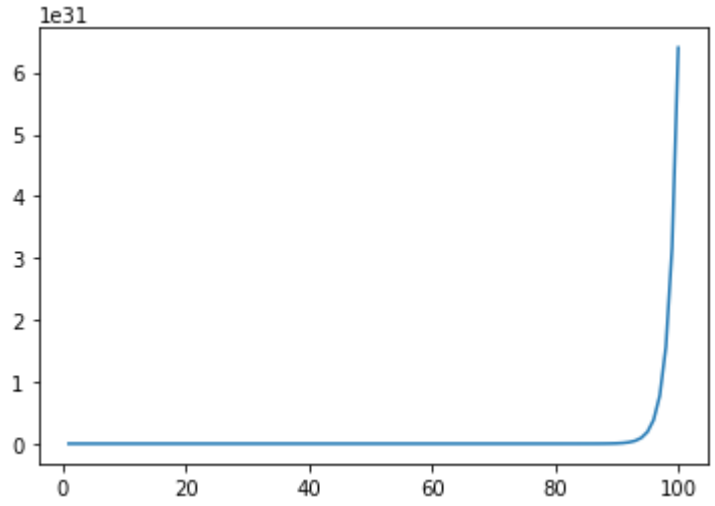
```
import numpy as np
import math
def Sum_averages(res):
    sum_ave=0
    N=len(res)
    S=sum(res)           #The sum of the elements of the array
    for i in range(N):
        bc = math.factorial(N-1)/(math.factorial(i) * math.factorial(N-1-i))  #The binomial coeffic
        sum_ave += bc*S/(i+1)
    return sum_ave
N=int(input("Please input N:"))
res=Random_integer(N)
print("Input:",res)
print("Output:")
print(Sum_averages(res))                    #Print the sum of the means for each subset
```

```
Please input N:3
Input: [1 0 8]
Output:
21.0
```

4.3 [5 points] Call Sum_averages with N increasing from 1 to 100, assign the output to a list called Total_sum_averages. Plot Total_sum_averages, describe what do you see.

In [24]:

```
import matplotlib.pyplot as plt
Total_sum_averages=[]
for i in range(1,101):
    res=list(range(1, i+1))           #Loop generated list with N increasing from 1 to 100
    Total_sum_averages.append(Sum_averages(res))
#print(Total_sum_averages)
x=list(range(1,101))
plt.plot(x, Total_sum_averages)       #plot
plt.show()
```



5. Path counting

5.1 [5 points] Create a matrix with N rows and M columns, fill the right-bottom corner and top-left corner cells with 1, and randomly fill the rest of matrix with integer 0 or 1.

In [41]:

```
import numpy as np
def creat_matrix(N,M):
    matrix=np.random.randint(2, size=(N,M))   #Create a random number matrix with N rows and M colum
    matrix[0,0]=1
    matrix[N-1,M-1]=1
    return matrix
N=int(input("Please input N:"))
M=int(input("Please input M:"))
res=creat_matrix(N,M)
print(res)
```

```
Please input N:5
Please input M:5
[[1 0 0 0 1]
 [1 1 1 1 1]
 [1 0 1 1 1]
 [1 1 1 1 0]
 [1 1 1 1 1]]
```

5.2 [25 points] Consider a cell marked with 0 as a blockage or dead-end, and a cell marked with 1 is good to go. Write a function Count_path to count total number of paths to reach the right-bottom corner cell from the top-left corner cell.

Notice: for a given cell, you are only allowed to move either rightward or downward.

In [46]:

```
def Count_path(matrix):
    path=np.zeros((N,M),dtype = int)#Create a matrix with N rows and M columns to record the number
    for i in range(0, N):           #For the 0th column of ‘matrix’ , ’path’ and ’matrix’ remain th
        if matrix[i,0] == 0:
            break
        else:
            path[i,0] = matrix[i,0]

    for j in range(0,M):            #For the 0th row of ‘matrix’ , ’path’ and ’matrix’ remain the s
        if matrix[0,j] == 0:
            break
        else:
            path[0,j] = matrix[0,j]


    for i in range(1, N):           #For elements inside the ’matrix’, if 1, then ’path’ accumulate
        for j in range(1,M):
            if matrix[i,j] == 1:
                path[i,j] = path[i - 1,j] + path[i,j - 1]
    return path[-1,-1]              #Returns the result of the last path accumulation
N=int(input("Please input N:"))
M=int(input("Please input M:"))
res=creat_matrix(N,M)
print(res)
Count_path(res)
```

```
Please input N:3
Please input M:3
[[1 1 0]
 [0 1 1]
 [0 1 1]]
```

Out[46]:

```
2
```

5.3 [5 points] Let N = 10, M = 8, run Count_path for 1000 times, each time the matrix (except the right-bottom corner and top-left corner cells, which remain being 1) is re-filled with integer 0 or 1 randomly, report the mean of total number of paths from the 1000 runs.

In [62]:

```
N=10
M=8
ave=sum(Count_path(creat_matrix(N,M)) for _ in range(1000))  #total number of paths from the 1000 r
print(ave/1000)      #print the mean of total number of paths from the 1000 runs.
```

```
0.347
```