Assignment 1

1. Flowchart

```
import random

def Get_value(x, y, z):
    print("We get the value = {}.".format(x+y-10*z))
def Print_values(a, b, c):
    print("a={} b={} c={}".format(a,b,c))
    if (a > b):
        if(b > c):
            Get_value(a, b, c)
        else:
            if (a > c):
                Get_value(a, c, b)
            else:
                Get_value(c, a, b)
    else:
        if (b > c):
            if (a > c):
                Get_value(b, a, c)
            else:
                Get_value(b, c, a)
        else:
            Get_value(c, b, a)

def main():
    a = round(random.random()*10)
    b = round(random.random()*10)
    c = round(random.random()*10)
    Print_values(a, b, c)
    a,b,c = 10,5,1
    Print_values(a, b, c)


if __name__ == '__main__':
    main()
```

Output :

```
a=4 b=8 c=1
We get the value = 2.
a=10 b=5 c=1
We get the value = 5.
```

2. Continuous celing function

```
from math import ceil

```

```
def Get_fx(x):
    if(x != 1):
        return(Get_fx(ceil(x/3)) + 2*x)
    else:
        return 1

def main():
    n = eval(input("Please input your list number(>0):"))
    i = 0
    x_lst = []
    answer_lst = []
    while(i < n):
        x = eval(input("Please input you number:"))
        x_lst.append(x)
        i = i + 1

    print("Your number list is :")
    for i in x_lst:
        answer = Get_fx(i)
        answer_lst.append(answer)
        print(i, end = " ")
    print() # get a blank line
    print("Your answer list is :")
    for i in answer_lst:
        print(i, end = " ")


if __name__ == '__main__':
    main()
```

Output :

```
Please input your list number(>0):9
Please input you number:1
Please input you number:6
Please input you number:88
Please input you number:22
Please input you number:16
Please input you number:3
Please input you number:5
Please input you number:77
Please input you number:68
Your number list is :
1 6 88 22 16 3 5 77 68
Your answer list is :
1 17 269 67 49 7 15 231 205
```

First we define a function get the $f(x) = f(ceil(x/3)) + 2x, (f(1) = 1)$, then we get input list(let user input the list size and number). When we input a list [1, 6, 88, 22, 16, 3, 5, 77, 49], we get the answer list [1, 17, 269, 67, 49, 7, 15, 231, 205]

3. Dice rollint

3.1

```
import random

def go_or_not(x,face_sum,count):
    remain_sum = x - face_sum
    remain_count = 10 - count
    if (remain_count * 6 < remain_sum):
        return False
    elif (remain_count > remain_sum):
        return False
    else:
        return True

def Find_number_of_ways(x):

    Face = [1,2,3,4,5,6]
    face_sum = 0
    count = 0
    ways = 0
    for dice1 in range(1, 7):
        face_sum = dice1
        count = 1
        if(not go_or_not(x,face_sum,count)):
            face_sum -= dice1
            count -= 1
            continue
        for dice2 in range(1,7):
            face_sum = dice2 + dice1
            count = 2
            if(not go_or_not(x,face_sum,count)):
                face_sum -= dice2
                count -= 1
                continue
            for dice3 in range(1,7):
                face_sum = dice3 + dice2 + dice1
                count = 3
                if(not go_or_not(x,face_sum,count)):
                    face_sum -= dice3
                    count -= 1
                    continue
                for dice4 in range(1,7):
                    face_sum = dice4 + dice3 + dice2 + dice1
                    count = 4
                    if(not go_or_not(x,face_sum,count)):
                        face_sum -= dice4
                        count -= 1
                        continue
                    for dice5 in range(1,7):
                        face_sum = dice5 + dice4 + dice3 + dice2 + dice1
                        count = 5
                        if(not go_or_not(x,face_sum,count)):
                            face_sum -= dice5
                            count -= 1
                            continue
                        for dice6 in range(1,7):
```

```
                                face_sum = dice6 + dice5 + dice4 + dice3 + dice2
+ dice1
                                count = 6
                                if(not go_or_not(x,face_sum,count)):
                                    face_sum -= dice6
                                    count -= 1
                                    continue
                                for dice7 in range(1,7):
                                    face_sum = dice7 + dice6 + dice5 + dice4 +
dice3 + dice2 + dice1
                                    count = 7
                                    if(not go_or_not(x,face_sum,count)):
                                        continue
                                    for dice8 in range(1,7):
                                        face_sum = dice8 + dice7 + dice6 + dice5
+ dice4 + dice3 + dice2 + dice1
                                        count = 8
                                        if(not go_or_not(x,face_sum,count)):
                                            continue
                                        for dice9 in range(1,7):
                                            face_sum = dice9 + dice8 + dice7 +
dice6 + dice5 + dice4 + dice3 + dice2 + dice1
                                            count = 9
                                            if(not go_or_not(x,face_sum,count)):
                                                continue
                                            for dice10 in range(1,7):
                                                face_sum = dice10 + dice9 +dice8
+ dice7 + dice6 + dice5 + dice4 + dice3 + dice2 + dice1
                                                if(face_sum == x):
                                                    face_sum = 0
                                                    count = 0
                                                    ways += 1
                                                    break
                                                else:
                                                    face_sum -= dice10
                                                    count -= 1

    return ways
```

We use the traversal method to obtain possible results, but we have 6^{10} possibilities, it's too big to computer all possibilities. So we define a function go_or_not to judge whether to continue.

3.2

Then we add some code to get Number_of_ways

```
def main():
    Number_of_ways = []
    for number in range(10,61):
        ways = Find_number_of_ways(number)
        Number_of_ways.append(ways)
    print(Number_of_ways)
    max_value = 0
    max_num = 0
    for i in range(len(Number_of_ways)):
        if(Number_of_ways[i] > max_value):
            max_value = Number_of_ways[i]
```

```
            max_num = i
    x = 10 + max_num
    print("x = {}".format(x))

if __name__ == '__main__':
    main()
```

Output :

```
[1, 10, 55, 220, 715, 2002, 4995, 11340, 23760, 46420, 85228, 147940, 243925, 383470, 576565, 831204, 1151370, 1535040, 1972630, 2446300, 2930455, 3393610, 3801535, 4121260, 4325310, 4395456, 4325310, 4121260, 3801535, 3393610, 2930455, 2446300, 1972630, 1535040, 1151370, 831204, 576565, 383470, 243925, 147940, 85228, 46420, 23760, 11340, 4995, 2002, 715, 220, 55, 10, 1]
x = 35
```

4. Dynamic programmint

4.1

```
import random

def Random_integer(N):
    lst = []
    for i in range(N):
        lst.append(round(random.random()*10))
    return lst
```

4.2

```
def Get_average(lst):
    lst_sum = 0
    lst_count = len(lst)
    for i in lst:
        lst_sum += i
    return lst_sum/lst_count

def Sum_averages(lst):
    lst_len = len(lst)
    sum_average = []
    #I got inspired by reading https://blog.csdn.net/weixin_43827397/article/details/107567071
    for i in range(lst_len ** 2):
        sub_lst = []
        for j in range(lst_len):
            if(i >> j) % 2:
                sub_lst.append(lst[j])
        if(len(sub_lst) <= 0):
            continue
        average = Get_average(sub_lst)
        sum_average.append(average)
    return sum(sum_average)
```

We define a function Get_average to get average of list.

4.3

We add some codes to get a list called Total_sum_averages and plot Total_sum_averages.

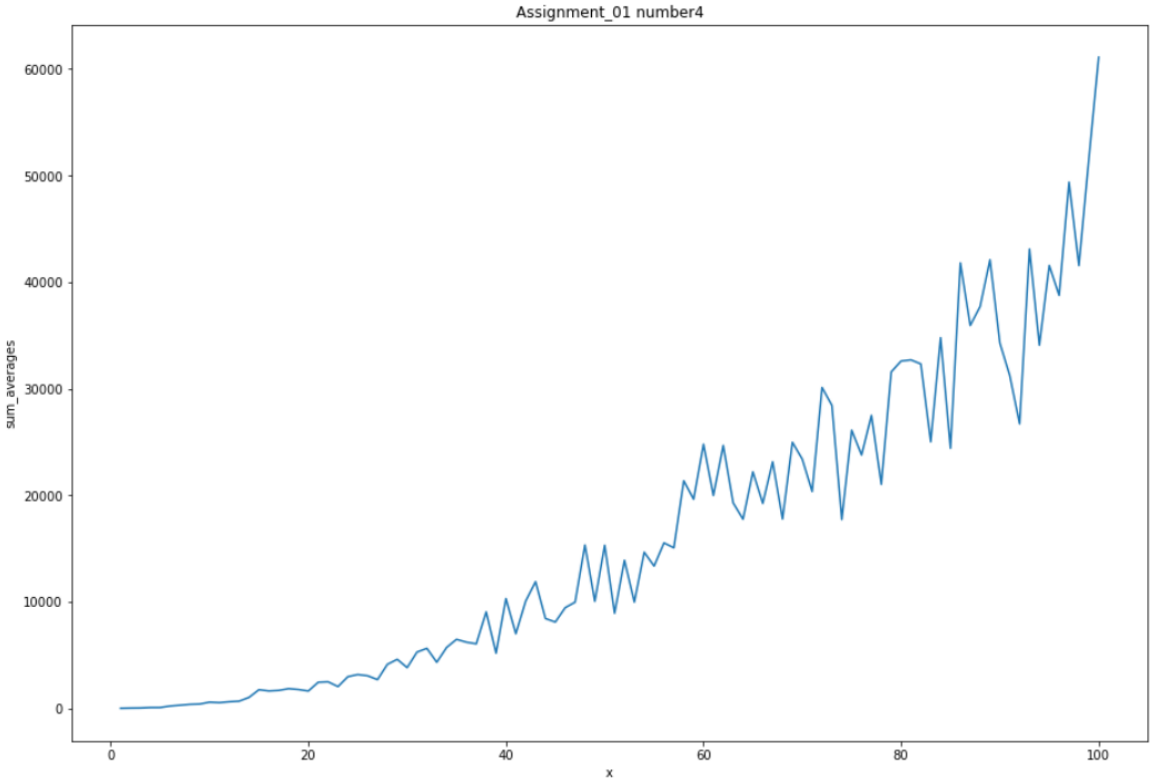
```
import matplotlib.pyplot as plt
%matplotlib inline

def main():
#     lst1 = Random_integer(10)
#     print("Your list is :")
#     print(lst1)
#     Sum_averages(lst1)
    N_lst = []
    Total_sum_averages = []
    for i in range(1, 101):
        N_lst.append(i)
        Total_sum_averages.append(Sum_averages(Random_integer(i)))
    print(N_lst)
    print(Total_sum_averages)
    fig = plt.figure(figsize = (12,8),
                     )
    ax = fig.add_subplot(111)
    ax = fig.add_axes([0,0,1,1])
    ax.plot(N_lst, Total_sum_averages)
    ax.set_title("Assignment_01 number4")
    ax.set_xlabel("x")
    ax.set_ylabel("sum_averages")


if __name__ == '__main__':
    main()
```

Output :

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
[0, 18.0, 28.0, 67.5, 70.16666666666666, 209.3, 286.05, 367.5, 397.04999999999995, 573.5999999999999, 529.5666666666668, 612.7285714285714, 668.2071428571429, 1020.928571428572, 1742.7714285714294, 1625.625, 1676.7499999999993, 1837.529761904762, 1760.135714285712, 1619.3392857142867, 2442.119047619047, 2483.3178571428566, 2034.2444444444448, 2962.4365079365066, 3165.592063492066, 3048.941666666665, 2684.640476190478, 4122.147222222226, 4595.545238095241, 3812.8297619047607, 5275.168650793652, 5626.499999999989, 4317.214285714284, 5710.715476190476, 6470.610714285701, 6202.277777777781, 6047.630952380957, 9047.445238095237, 5171.225793650799, 10298.926984126989, 6995.004761904762, 10060.804761904761, 11882.253174603191, 8426.00595238095, 8092.528968253981, 9440.180735930735, 9956.410173160173, 15321.671717171712, 10036.841414141416, 15299.348773448766, 8917.794660894644, 13889.757972582995, 9965.818686868666, 14644.461868686862, 13345.96525974026, 15520.240584415524, 15059.901370851418, 21351.874531024554, 19618.636616161613, 24783.26825396823, 19970.98088023086, 24669.066161616156, 19284.00569985569, 17745.000000000015, 22189.857142857116, 19227.888888888898, 23134.32063492063, 17763.150000000023, 24960.851587301608, 23378.373015873018, 20342.70436507937, 30104.729870129893, 28427.08354978352, 17716.556168831154, 26100.230303030443, 23770.769480519535, 27490.394227994253, 21015.124639249603, 31566.314285714263, 32586.5128787879, 32696.801587301532, 32316.008405483364, 25007.839177489175, 34784.34314574316, 24404.507142857183, 41801.36435786434, 35922.88997113996, 37706.819408369396, 42096.80404040401, 34278.36136363628, 31236.031135531106, 26693.237179487172, 43105.88928571441, 34073.09218559219, 41562.30619658121, 38750.23076923084, 49384.82132867134, 41543.359940005976, 51486.637165611966, 61101.368722943014]
```



We can see that the sum of the average values of all subsets increase with the increase of the array capacity.

5. Path counting

5.1

```
import numpy as np

def Get_matrix(N, M):
    a = np.random.choice(a = [0,1], size = [N,M])
    a[0][0] = 1
    a[-1][-1] = 1
    return a
```

5.2

```
import numpy as np

def Get_matrix(N, M):
    a = np.random.choice(a = [0,1], size = [N,M])
    a[0][0] = 1
    a[-1][-1] = 1
    return a

path = 0
okay = 0
def Count_path(Matrix,x,y,endx,endy):
    global path
    global okay

    if (okay == 1):
        return path
```

```
17
18      if(x == endx and y == endy):
19          okay = 1
20          print("find ways")
21          print("path = {}".format(path))
22          return path
23
24      if(y != endy and Matrix[x][y+1] != 0 and okay!=1):
25          print("right")
26          path += 1
27          Move(Matrix,x,y+1,endx,endy)
28      if (okay == 1):
29          return path
30
31      if(x != endx and Matrix[x+1][y] != 0 and okay!=1):
32          print("down")
33          path += 1
34          Move(Matrix,x+1 ,y,endx,endy)
35      if (okay == 1):
36          return path
37
38      if (okay == 0):
39          print("no way")
40          path -= 1
41          return 0
42
43
44  a = Get_matrix(3,3)
45  print(a)
46  Count_path(a,0,0,2,2)
47  if(not okay):
48      print("path = 0");
```

5.2 Output :

situation01 :

```
[[1 0 0]
 [0 1 1]
 [0 0 1]]
no way
path = 0
```

situation02 :

```
[[1 1 1]
 [0 1 0]
 [1 1 1]]
right
right
no way
down
down
right
find ways
path = 4
```

5.3

```python
import numpy as np

def Get_matrix(N, M):
    a = np.random.choice(a = [0,1], size = [N,M])
    a[0][0] = 1
    a[-1][-1] = 1
    return a

path = 0
okay = 0
def Count_path(Matrix,x,y,endx,endy):
    global path
    global okay

    if (okay == 1):
        return path

    if(x == endx and y == endy):
        okay = 1
        print("find ways")
        print("path = {}".format(path))
        return path

    if(y != endy and Matrix[x][y+1] != 0 and okay!=1):
        #print("right")
        path += 1
        Count_path(Matrix,x,y+1,endx,endy)
    if (okay == 1):
        return path

    if(x != endx and Matrix[x+1][y] != 0 and okay!=1):
        #print("down")
        path += 1
        Count_path(Matrix,x+1 ,y,endx,endy)
    if (okay == 1):
        return path

    if (okay == 0):
        #print("no way")
        path -= 1
        return 0

def main():
    global path
    global okay
    N = 10
    M = 8
    path_lst = []
    for i in range(1000):
        a = Get_matrix(N,M)
        path_ans = Count_path(a,0,0,N-1,M-1)
        path_lst.append(path_ans);
        if(okay):
            print(a)
        path = 0
```

```
            okay = 0
        print(path_lst)
        total_num = sum(path_lst)
        print(total_num)


if __name__ == '__main__':
    main()
```

Output :