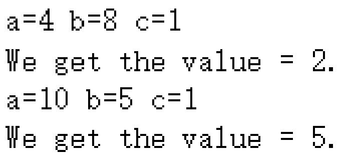
Assignment 1

1. Flowchart

```python
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 :



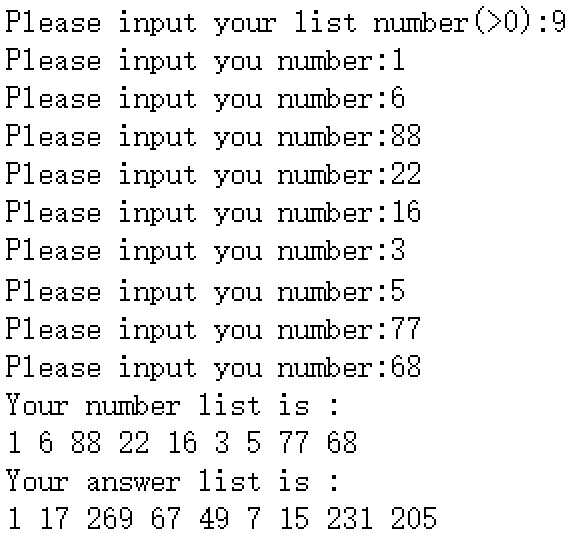
```
a=4 b=8 c=1
We get the value = 2.
a=10 b=5 c=1
We get the value = 5.
```

2. Continuous celing function

```python
from math import ceil

```

```python
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
```

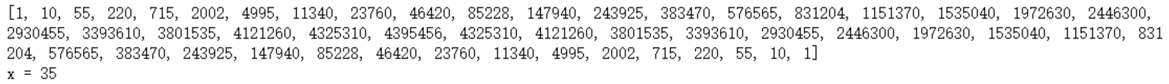
We use the traversal method to obtain possible results, but we have 6^{10} possibilities, it's too big to computer all possibilities. So we define a function go_or_not to judge whether to continue.

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, 831
204, 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)
    i = 1
    sum_average = []
    while (i <= lst_len):
        #print("get {} element(s) list: ".format(i))
        index = 0
        while(index + i <= lst_len):
            #print(lst[index:index+i], end = " ")
            average = Get_average(lst[index:index+i])
            sum_average.append(average)
            #print("average = {}".format(average), end = " ")
            index += 1
        #print()
        i += 1
    return sum(sum_average)
```

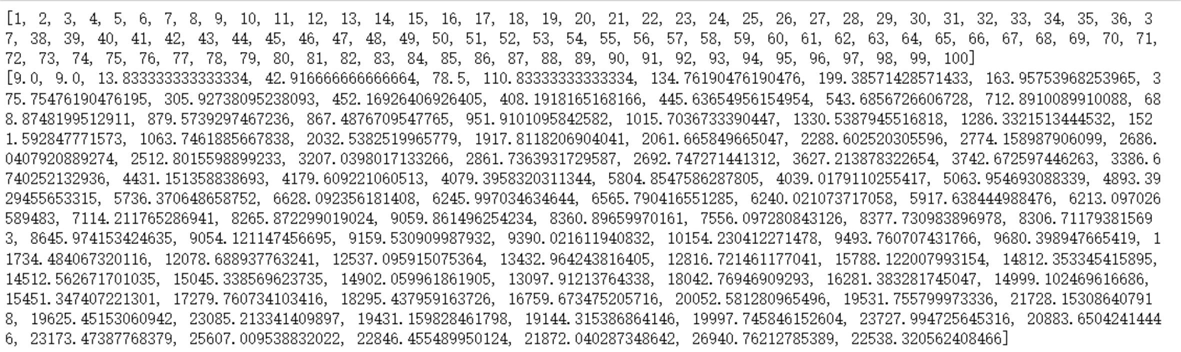
We define a function Get_average to get average of list.

4.3

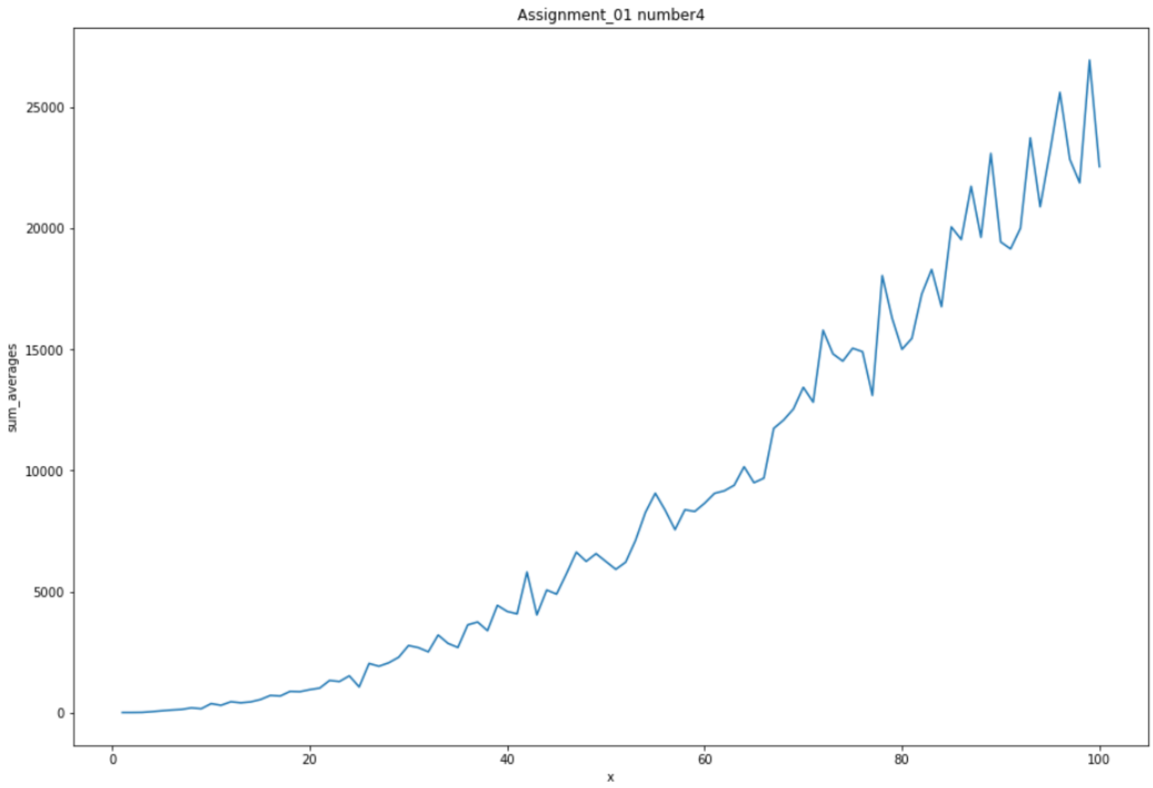
We add some codes to get a list called Total_sum_averages and plot Total_sum_averages.

```python
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, 3
7, 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]
[9.0, 9.0, 13.833333333333334, 42.916666666666664, 78.5, 110.83333333333334, 134.76190476190476, 199.38571428571433, 163.95753968253965, 3
75.75476190476195, 305.92738095238093, 452.16926406926405, 408.1918165168166, 445.63654956154954, 543.6856726606728, 712.8910089910088, 68
8.8748199512911, 879.5739297467236, 867.4876709547765, 951.9101095842582, 1015.7036733390447, 1330.5387945516818, 1286.3321513444532, 152
1.592847771573, 1063.7461885667838, 2032.5382519965779, 1917.8118206904041, 2061.665849665047, 2288.602520305596, 2774.158987906099, 2686.
0407920889274, 2512.8015598899233, 3207.0398017133266, 2861.7363931729587, 2692.747271441312, 3627.213878322654, 3742.672597446263, 3386.6
740252132936, 4431.151358838693, 4179.609221060513, 4079.3958320311344, 5804.8547586287805, 4039.0179110255417, 5063.954693088339, 4893.39
29455653315, 5736.370648658752, 6628.092356181408, 6245.997034634644, 6565.790416551285, 6240.021073717058, 5917.638444988476, 6213.097026
589483, 7114.211765286941, 8265.872299019024, 9059.861496254234, 8360.89659970161, 7556.097280843126, 8377.730983896978, 8306.71179381569
3, 8645.974153424635, 9054.121147456695, 9159.530909987932, 9390.021611940832, 10154.230412271478, 9493.760707431766, 9680.398947665419, 1
1734.484067320116, 12078.688937763241, 12537.095915075364, 13432.964243816405, 12816.721461177041, 15788.122007993154, 14812.353345415895,
14512.562671701035, 15045.338569623735, 14902.059961861905, 13097.91213764338, 18042.76946909293, 16281.383281745047, 14999.102469616686,
15451.347407221301, 17279.760734103416, 18295.437959163726, 16759.673475205716, 20052.581280965496, 19531.755799973336, 21728.15308640791
8, 19625.45153060942, 23085.213341409897, 19431.159828461798, 19144.315386864146, 19997.745846152604, 23727.994725645316, 20883.6504241444
6, 23173.47387768379, 25607.009538832022, 22846.455489950124, 21872.040287348642, 26940.76212785389, 22538.320562408466]
```



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

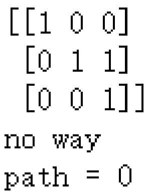
```
import numpy as np

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

    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
        Move(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
        Move(Matrix,x+1 ,y,endx,endy)
    if (okay == 1):
        return path

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


a = Get_matrix(3,3)
print(a)
Count_path(a,0,0,2,2)
if(not okay):
    print("path = 0");
```

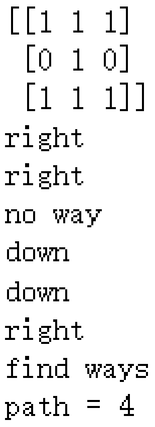
5.2 Output :

situation01 :



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

situation02 :

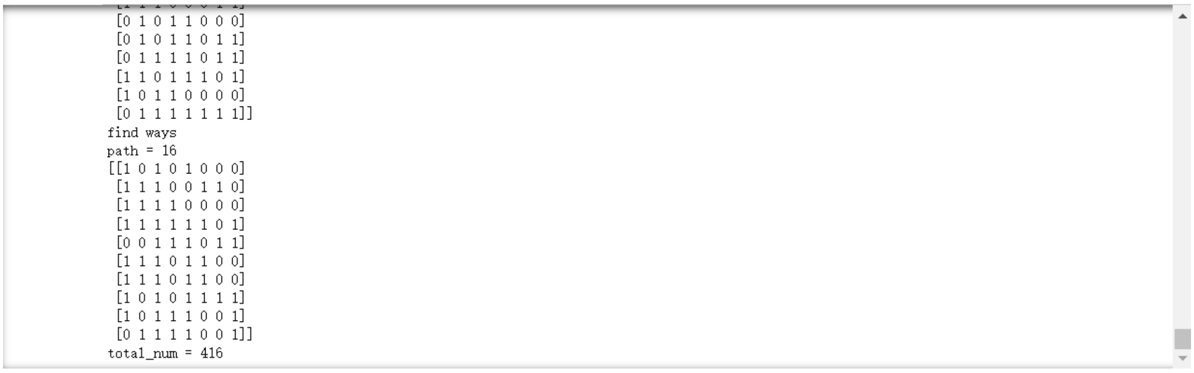


```
[[1 1 1]
 [0 1 0]
 [1 1 1]]
right
right
no way
down
down
right
find ways
path = 4
```

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 :



```
 [0 1 0 1 1 0 0 0]
 [0 1 0 1 1 0 1 1]
 [0 1 1 1 1 0 1 1]
 [1 1 0 1 1 1 0 1]
 [1 0 1 1 0 0 0 0]
 [0 1 1 1 1 1 1 1]]
find ways
path = 16
[[1 0 1 0 1 0 0 0]
 [1 1 1 0 0 1 1 0]
 [1 1 1 1 0 0 0 0]
 [1 1 1 1 1 1 0 1]
 [0 0 1 1 1 0 1 1]
 [1 1 1 0 1 1 0 0]
 [1 1 1 0 1 1 0 0]
 [1 0 1 0 1 1 1 1]
 [1 0 1 1 1 0 0 1]
 [0 1 1 1 1 0 0 1]]
total_num = 416
```