subjects :

1. generate list with n items
2. search
3. sort
4. binary search
5. mean,median,mode,weighted mean,geometric mean

generate list with n items

In [1]:

```
import random
```

In [3]:

```
random.random()
```

Out[3]:

```
0.9609762734157393
```

In [6]:

```
s=random.randint(1,100)   # random.randint(min,max)
s
```

Out[6]:

```
74
```

In [7]:

```
def get_n_random_numbers(n=10,min_=-5,max_=5):
    numbers=[]
    for i in range(n):
        numbers.append(random.randint(min_,max_))
    return numbers
get_n_random_numbers()
```

Out[7]:

```
[-2, -2, 4, -3, 1, -1, 5, -4, -5, 5]
```

In [17]:

```
my_list=get_n_random_numbers(15,-4,4)
my_list
```

Out[17]:

```
[4, -1, -2, 0, 2, -3, 0, -4, -3, 0, -4, -1, 3, -1, -4]
```

histogram with two methods

In [15]:

```
# for a list [0, -4, 8, -1, 0, -3, 6, 3, 0, 1]
# get the histogram , with array of tuples format
histgram_1=[
    (-4,1),
    (-3,1),
    (-1,1),
    (0,2),
    (1,1),
    (3,1),
    (6,1),
    (8,1)
]
```

In [18]:

```
my_list
```

Out[18]:

```
[4, -1, -2, 0, 2, -3, 0, -4, -3, 0, -4, -1, 3, -1, -4]
```

In [19]:

```
sorted(my_list)
```

Out[19]:

```
[-4, -4, -4, -3, -3, -2, -1, -1, -1, 0, 0, 0, 2, 3, 4]
```

In [20]:

```
def my_frequency_with_dict(list):
    frequency_dict={}  # dict()={}
    for item in list:
        if (item in frequency_dict):
            frequency_dict[item]=frequency_dict[item]+1
        else:
            frequency_dict[item]=1
    return  frequency_dict
```

In [21]:

```
my_frequency_with_dict(my_list)
```

Out[21]:

```
{4: 1, -1: 3, -2: 1, 0: 3, 2: 1, -3: 2, -4: 3, 3: 1}
```

In [25]:

```
def my_frequency_with_list_of_tuples(list_1):
    frequency_list=[]
    for i in range(len(list_1)):
        s=False
        for j in range(len(frequency_list)):
            if (list_1[i]==frequency_list[j][0]):
                frequency_list[j][1]=frequency_list[j][1]+1
                s=True
        if(s==False):
            frequency_list.append([list_1[i],1])
    return  frequency_list
```

In [26]:

```
my_list=[2,3,2,5,8,2,4,3,3,2,8,5,2,4,4,4,4,4]
result_1=my_frequency_with_dict(my_list)
result_2=my_frequency_with_list_of_tuples(my_list)
result_1,result_2
```

Out[26]:

```
({2: 5, 3: 3, 5: 2, 8: 2, 4: 6}, [[2, 5], [3, 3], [5, 2], [8, 2], [4, 6]])
```

mode of a list with histogram

In [30]:

```
my_list_1=get_n_random_numbers(5,-2,2)
my_hist_d=my_frequency_with_dict(my_list_1)
my_hist_d
```

Out[30]:

```
{1: 3, -2: 2}
```

In [29]:

```
my_hist_l=my_frequency_with_list_of_tuples(my_list_1)
my_hist_l
```

Out[29]:

```
[[-3, 2], [-5, 2], [5, 1], [4, 1], [1, 1], [2, 1], [-2, 1], [0, 1]]
```

In [31]:

```
# to get mode , we have to search all keys on hist_dict
frequency_max=-1   # mode değeri, döngüde karşılaştırılacak hafıza amaçlı değer
mode=-1
for key in my_hist_d.keys():
    print(key,my_hist_d[key])
    if my_hist_d[key]>frequency_max:
        frequency_max=my_hist_d[key]
        mode=key
mode,frequency_max
```

```
1 3
-2 2
```

Out[31]:

```
(3, 1)
```

In [32]:

```
# to get mode , we have to search all keys on hist_dict
def my_mode_with_dict(my_hist_d):

    frequency_max=-1   # mode değeri, döngüde karşılaştırılacak hafıza amaçlı değer
    mode=-1
    for key in my_hist_d.keys():
        # print(key,my_hist_d[key])
        if my_hist_d[key]>frequency_max:
            frequency_max=my_hist_d[key]
            mode=key
    return mode,frequency_max
```

In [33]:

```
my_mode_with_dict(my_hist_d)
```

Out[33]:

```
(1, 3)
```

In [38]:

```
my_list_100=get_n_random_numbers(100,-40,40)
my_hist_1=my_frequency_with_dict(my_list_100)
my_mode_with_dict(my_hist_1)
```

Out[38]:

```
(-15, 5)
```

In [41]:

```
my_hist_1
```

...

In [40]:

```
sorted(my_list_100)
```

...

mode of a list with histogram (a list of tuples)

In [43]:

```
my_list_1=get_n_random_numbers(10)
my_hist_list=my_frequency_with_list_of_tuples(my_list_1)
my_hist_list
```

Out[43]:

```
[[-2, 1], [3, 1], [2, 1], [-4, 1], [-5, 3], [1, 1], [-3, 1], [-1, 1]]
```

In [44]:

```
# to get mode , we have to search all keys on hist_dict
frequency_max=-1   # mode değeri, döngüde karşılaştırılacak hafıza amaçlı değer
mode=-1
for item,frequency in my_hist_list:
    print(item,frequency)
    if frequency>frequency_max:
        frequency_max=frequency
        mode=item
mode,frequency_max
```

```
-2 1
3 1
2 1
-4 1
-5 3
1 1
-3 1
-1 1
```

Out[44]:

```
(-5, 3)
```

In [45]:

```
# to get mode , we have to search all keys on hist_dict
def my_mode_with_list(my_hist_list):
    frequency_max=-1   # mode değeri, döngüde karşılaştırılacak hafıza amaçlı değer
    mode=-1
    for item,frequency in my_hist_list:
        print(item,frequency)
        if frequency>frequency_max:
            frequency_max=frequency
            mode=item
    return mode,frequency_max
```

In [46]:

```
my_mode_with_list(my_hist_list)
```

```
-2 1
3 1
2 1
-4 1
-5 3
1 1
-3 1
-1 1
```

Out[46]:

```
(-5, 3)
```

In [47]:

```
my_list_100=get_n_random_numbers(20,-4,4)
my_hist_1=my_frequency_with_list_of_tuples(my_list_100)
my_mode_with_list(my_hist_1)
```

```
-4 2
1 2
-1 4
-2 1
4 2
2 6
0 2
-3 1
```

Out[47]:

```
(2, 6)
```

In [48]:

```
my_list_100
```

Out[48]:

```
[-4, 1, -1, -2, 4, 2, 2, 2, -1, 2, 2, 2, -1, -4, 0, 0, 1, -3, -1, 4]
```

linear search on list

In [51]:

```
def my_linear_search(my_list,item_search):
    found=(-1,-1)   # default, eğer listede yoksa
    n=len(my_list)
    for indis in range(n):
        if my_list[indis]==item_search:
            found=(my_list[indis],indis)   # listede bulundu, return bulunn sayı, indisi
            # break, uncomment for last found
    return found
```

In [61]:

```
my_list=get_n_random_numbers(10,-5,5)
my_list
```

Out[61]:

```
[-4, -5, -2, 5, -1, -2, 0, -4, 0, 4]
```

In [57]:

```
my_linear_search(my_list,10)
```

Out[57]:

```
(-1, -1)
```

mean of list

In [63]:

```
my_list=get_n_random_numbers(10,-50,50)
my_list
```

Out[63]:

```
[-44, 40, -46, -5, 49, -12, -49, -19, -20, 4]
```

In [64]:

```
s,t=0,0
for item in my_list:
    s=s+1
    t=t+item
mean_=t/s
mean_
```

Out[64]:

```
-10.2
```

In [65]:

```
def my_mean(my_list):
    s,t=0,0
    for item in my_list:
        s=s+1
        t=t+item
    mean_=t/s
    return mean_
```

In [70]:

```
my_list=get_n_random_numbers(4,-5,5)
print(my_list)
my_mean(my_list)
```

```
[-2, -5, 1, 5]
```

Out[70]:

```
-0.25
```

sort the list

In [27]:

```
my_list
```

Out[27]:

```
[2, 3, 2, 5, 8, 2, 4, 3, 3, 2, 8, 5, 2, 4, 4, 4, 4, 4]
```

In [71]:

```
n=len(my_list)
print(my_list)
for i in range(n-1,-1,-1):
    for j in range(0,i):
        if not(my_list[j]<my_list[j+1]):
            # print("swap işlemi")
            temp=my_list[j]
            my_list[j]=my_list[j+1]
            my_list[j+1]=temp
print(my_list)
```

```
[-2, -5, 1, 5]
[-5, -2, 1, 5]
```

In [76]:

```
# with function
def my_bubble_sort(my_list):
    n=len(my_list)
    #print(my_list)
    for i in range(n-1,-1,-1):
        for j in range(0,i):
            if not(my_list[j]<my_list[j+1]):
                # print("swap işlemi")
                temp=my_list[j]
                my_list[j]=my_list[j+1]
                my_list[j+1]=temp
    return my_list
```

In [77]:

```
my_list=get_n_random_numbers(4,-5,5)
print(my_list)
my_bubble_sort(my_list)
```

```
[-3, 0, 4, 0]
```

Out[77]:

```
[-3, 0, 0, 4]
```

binary search on a sorted list

In [78]:

```
def my_binary_search(my_list, item_search):
    found=(-1,-1)
    low = 0
    high = len(my_list) - 1

    while low <= high:
        mid = (low + high) // 2

        if my_list[mid] == item_search:
            return my_list[mid],mid
        elif my_list[mid] > item_search:
            high = mid - 1
        else:
            low = mid + 1

    return found  # None
```

In [79]:

```
my_list_1=get_n_random_numbers(10)
print("liste ",my_list_1)
my_list_2=bubble_sort(my_list_1)
print("sırali liste",my_list_2)
my_binary_search(my_list_2,3)  # 1
```

```
liste  [3, -5, 1, 2, -2, 4, 2, -4, 3, 1]
[3, -5, 1, 2, -2, 4, 2, -4, 3, 1]
sırali liste [-5, -4, -2, 1, 1, 2, 2, 3, 3, 4]
```

Out[79]:

```
(3, 7)
```

median of a list

In [80]:

```
size=input("dizi boyutunu giriniz")
size=int(size) # convert str to int
my_list_1=get_n_random_numbers(size)

print("liste ",my_list_1)
```

```
dizi boyutunu giriniz7
liste  [-5, 5, 3, 3, 5, -4, 0]
```

In [81]:

```
my_list_2=bubble_sort(my_list_1)
```

```
[-5, 5, 3, 3, 5, -4, 0]
```

In [82]:

```
print(my_list_2)
n=len(my_list_2)
if n%2==1:
    middle=int(n/2)+1
    median=my_list_2[middle]
    print(median)

else:
    middle_1=my_list_2[int(n/2)]
    middle_2=my_list_2[int(n/2)+1]
    median=(middle_1+middle_2)/2
    print (median)
```

```
[-5, -4, 0, 3, 3, 5, 5]
3
```

In [83]:

```
def my_median(my_list):
    my_list_2=bubble_sort(my_list)
    #print(my_list_2)
    n=len(my_list_2)
    if n%2==1:
        middle=int(n/2)+1
        median=my_list_2[middle]
        #print(median)

    else:
        middle_1=my_list_2[int(n/2)]
        middle_2=my_list_2[int(n/2)+1]
        median=(middle_1+middle_2)/2
        #print (median)

    return median
```

In [85]:

```
my_list_2=get_n_random_numbers(6,-10,10)
my_median(my_list_2)
```

```
[-5, -4, 0, 3, 3, 5, 5]
```

Out[85]:

```
3
```

In [ ]:

```

```