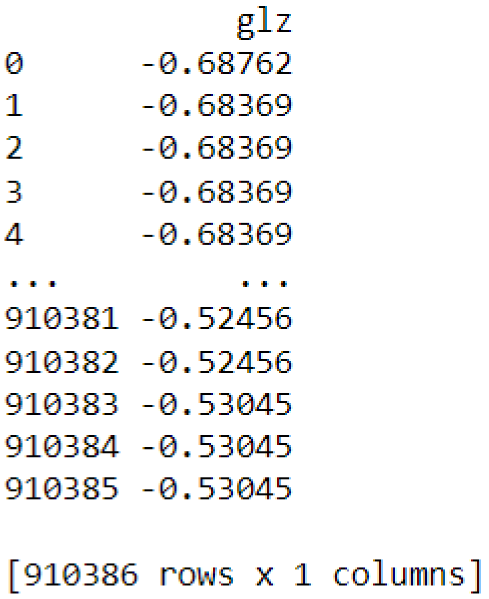
import pandas as pd

import numpy as np

from statistics import mean

df=pd.read_csv("Gait_Data.csv",usecols =['glz'])

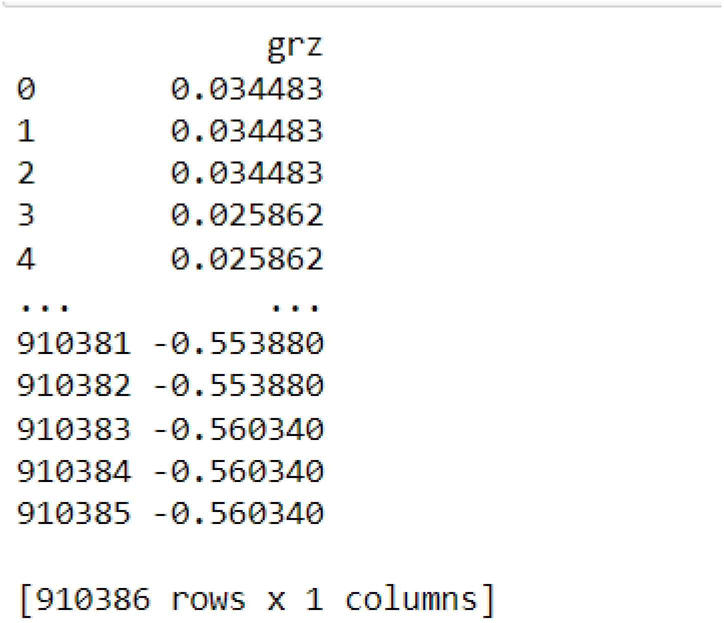
print(df)



```
             glz
0       -0.68762
1       -0.68369
2       -0.68369
3       -0.68369
4       -0.68369
...          ...
910381 -0.52456
910382 -0.52456
910383 -0.53045
910384 -0.53045
910385 -0.53045

[910386 rows x 1 columns]
```

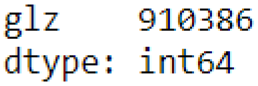
ef=pd.read_csv("Gait_Data.csv",usecols =['grz'])

print(ef)



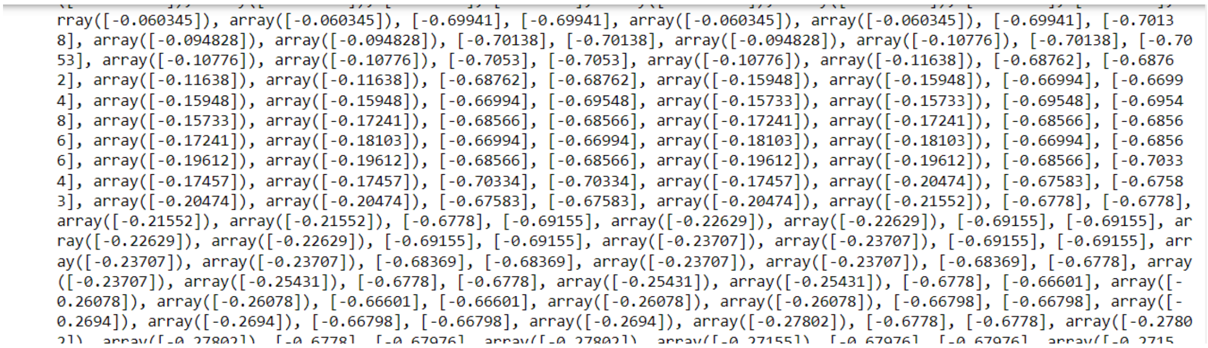
```
              grz
0        0.034483
1        0.034483
2        0.034483
3        0.025862
4        0.025862
...           ...
910381 -0.553880
910382 -0.553880
910383 -0.560340
910384 -0.560340
910385 -0.560340

[910386 rows x 1 columns]
```

```
C=90.123

cfc=0

a=df.values.tolist()

z=ef.values.tolist()

b=np.array(a)

z=np.array(z)

d=df.count()

print(d)
```



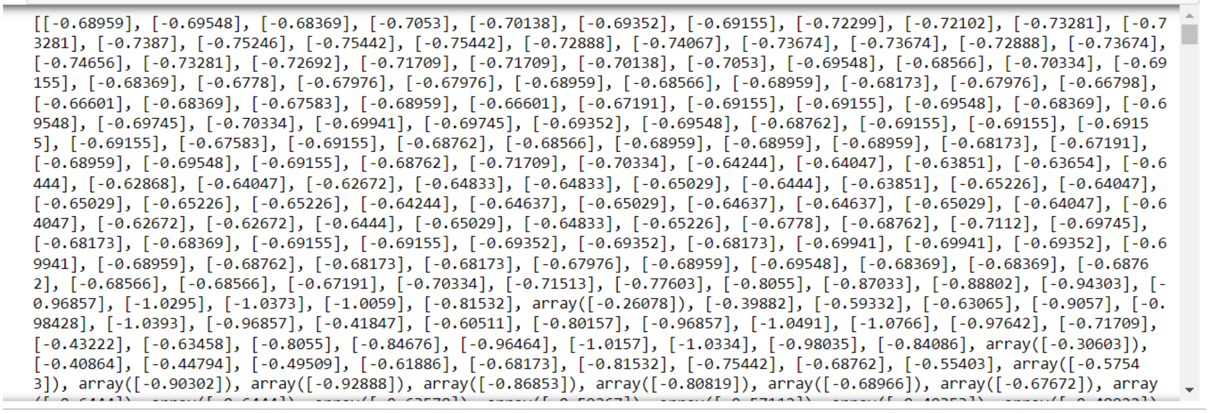
```
glz    910386
dtype: int64
```

```
c=[]

e=len(a)/10

for i in range(0,int(e),2):

      c.append(a[i])

      c.append(a[i+1])

      c.append(z[i])

      c.append(z[i+1])

f=len(c)


print(f)

print(c)
```



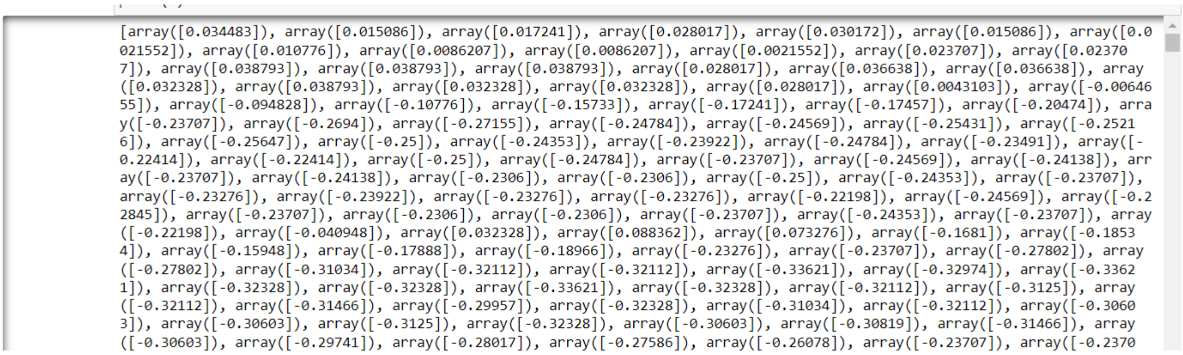
```
rray([-0.060345]), array([-0.060345]), [-0.69941], [-0.69941], array([-0.060345]), array([-0.060345]), [-0.69941], [-0.7013
8], array([-0.094828]), array([-0.094828]), [-0.70138], [-0.70138], array([-0.094828]), array([-0.10776]), [-0.70138], [-0.70
53], array([-0.10776]), array([-0.10776]), [-0.7053], [-0.7053], array([-0.10776]), array([-0.11638]), [-0.68762], [-0.6876
2], array([-0.11638]), array([-0.11638]), [-0.68762], [-0.68762], array([-0.15948]), array([-0.15948]), [-0.66994], [-0.6699
4], array([-0.15948]), array([-0.15948]), [-0.66994], [-0.69548], array([-0.15733]), array([-0.15733]), [-0.69548], [-0.6954
8], array([-0.15733]), array([-0.17241]), [-0.68566], [-0.68566], array([-0.17241]), array([-0.17241]), [-0.68566], [-0.6856
6], array([-0.17241]), array([-0.18103]), [-0.66994], [-0.66994], array([-0.18103]), array([-0.18103]), [-0.66994], [-0.6856
6], array([-0.19612]), array([-0.19612]), [-0.68566], [-0.68566], array([-0.19612]), array([-0.19612]), [-0.68566], [-0.7033
4], array([-0.17457]), array([-0.17457]), [-0.70334], [-0.70334], array([-0.17457]), array([-0.20474]), [-0.67583], [-0.6758
3], array([-0.20474]), array([-0.20474]), [-0.67583], [-0.67583], array([-0.20474]), array([-0.21552]), [-0.6778], [-0.6778],
array([-0.21552]), array([-0.21552]), [-0.6778], [-0.69155], array([-0.22629]), array([-0.22629]), [-0.69155], [-0.69155], ar
ray([-0.22629]), array([-0.22629]), [-0.69155], [-0.69155], array([-0.23707]), array([-0.23707]), [-0.69155], [-0.69155], arr
ay([-0.23707]), array([-0.23707]), [-0.68369], [-0.68369], array([-0.23707]), array([-0.23707]), [-0.68369], [-0.6778], array
([-0.23707]), array([-0.25431]), [-0.6778], [-0.6778], array([-0.25431]), array([-0.25431]), [-0.6778], [-0.66601], array([-
0.26078]), array([-0.26078]), [-0.66601], [-0.66601], array([-0.26078]), array([-0.26078]), [-0.66798], [-0.66798], array([-
0.2694]), array([-0.2694]), [-0.66798], [-0.66798], array([-0.2694]), array([-0.27802]), [-0.6778], [-0.6778], array([-0.2780
2]), array([-0.27802]), [-0.6778], [-0.67976], array([-0.27802]), array([-0.27155]), [-0.67976], [-0.67976], array([-0.2715
```

```
def find_peaks(c):

   peaks = []

   for i in range(1, len(c) - 1):
        if c[i] < c[i-1] and c[i] < c[i+1]:
            peaks.append(c[i])
    return peaks



g=find_peaks(c)
print(g)
k=len(g)
print(k)
```



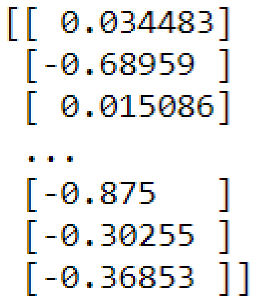
```
[[-0.68959], [-0.69548], [-0.68369], [-0.7053], [-0.70138], [-0.69352], [-0.69155], [-0.72299], [-0.72102], [-0.73281], [-0.7
3281], [-0.7387], [-0.75246], [-0.75442], [-0.75442], [-0.72888], [-0.74067], [-0.73674], [-0.73674], [-0.72888], [-0.73674],
[-0.74656], [-0.73281], [-0.72692], [-0.71709], [-0.71709], [-0.70138], [-0.7053], [-0.69548], [-0.68566], [-0.70334], [-0.69
155], [-0.68369], [-0.6778], [-0.67976], [-0.67976], [-0.68959], [-0.68566], [-0.68959], [-0.68173], [-0.67976], [-0.66798],
[-0.66601], [-0.68369], [-0.67583], [-0.68959], [-0.66601], [-0.67191], [-0.69155], [-0.69155], [-0.69548], [-0.68369], [-0.6
9548], [-0.69745], [-0.70334], [-0.69941], [-0.69745], [-0.69352], [-0.69548], [-0.68762], [-0.69155], [-0.69155], [-0.6915
5], [-0.69155], [-0.67583], [-0.69155], [-0.68762], [-0.68566], [-0.68959], [-0.68959], [-0.68959], [-0.68173], [-0.67191],
[-0.68959], [-0.69548], [-0.69155], [-0.68762], [-0.71709], [-0.70334], [-0.64244], [-0.64047], [-0.63851], [-0.63654], [-0.6
444], [-0.62868], [-0.64047], [-0.62672], [-0.64833], [-0.64833], [-0.65029], [-0.6444], [-0.63851], [-0.65226], [-0.64047],
[-0.65029], [-0.65226], [-0.65226], [-0.64244], [-0.64637], [-0.65029], [-0.64637], [-0.64637], [-0.65029], [-0.64047], [-0.6
4047], [-0.62672], [-0.62672], [-0.6444], [-0.65029], [-0.64833], [-0.65226], [-0.6778], [-0.68762], [-0.7112], [-0.69745],
[-0.68173], [-0.68369], [-0.69155], [-0.69155], [-0.69352], [-0.69352], [-0.68173], [-0.69941], [-0.69941], [-0.69352], [-0.6
9941], [-0.68959], [-0.68762], [-0.68173], [-0.68173], [-0.67976], [-0.68959], [-0.69548], [-0.68369], [-0.68369], [-0.6876
2], [-0.68566], [-0.68566], [-0.67191], [-0.70334], [-0.71513], [-0.77603], [-0.8055], [-0.87033], [-0.88802], [-0.94303], [-
0.96857], [-1.0295], [-1.0373], [-1.0059], [-0.81532], array([-0.26078]), [-0.39882], [-0.59332], [-0.63065], [-0.9057], [-0.
98428], [-1.0393], [-0.96857], [-0.41847], [-0.60511], [-0.80157], [-0.96857], [-1.0491], [-1.0766], [-0.97642], [-0.71709],
[-0.43222], [-0.63458], [-0.8055], [-0.84676], [-0.96464], [-1.0157], [-1.0334], [-0.98035], [-0.84086], array([-0.30603]),
[-0.40864], [-0.44794], [-0.49509], [-0.61886], [-0.68173], [-0.81532], [-0.75442], [-0.68762], [-0.55403], array([-0.5754
3]), array([-0.90302]), array([-0.92888]), array([-0.86853]), array([-0.80819]), array([-0.68966]), array([-0.67672]), array
```

```
def find_valley(c):
    valley = []
    for i in range(1, len(c) - 1):
        if c[i] > c[i-1] and c[i] > c[i+1]:
            valley.append(c[i])
    return valley
h=find_valley(c)
print(h)
l=len(h)
print(l)
```



```
[array([0.034483]), array([0.015086]), array([0.017241]), array([0.028017]), array([0.030172]), array([0.015086]), array([0.0
021552]), array([0.010776]), array([0.0086207]), array([0.0086207]), array([0.0021552]), array([0.023707]), array([0.02370
7]), array([0.038793]), array([0.038793]), array([0.038793]), array([0.028017]), array([0.036638]), array([0.036638]), array
([0.032328]), array([0.038793]), array([0.032328]), array([0.032328]), array([0.028017]), array([0.0043103]), array([-0.00646
55]), array([-0.094828]), array([-0.10776]), array([-0.15733]), array([-0.17241]), array([-0.17457]), array([-0.20474]), arra
y([-0.23707]), array([-0.2694]), array([-0.27155]), array([-0.24784]), array([-0.24569]), array([-0.25431]), array([-0.2521
6]), array([-0.25647]), array([-0.25]), array([-0.24353]), array([-0.23922]), array([-0.24784]), array([-0.23491]), array([-
0.22414]), array([-0.22414]), array([-0.25]), array([-0.24784]), array([-0.23707]), array([-0.24569]), array([-0.24138]), arr
ay([-0.23707]), array([-0.24138]), array([-0.2306]), array([-0.2306]), array([-0.25]), array([-0.24353]), array([-0.23707]),
array([-0.23276]), array([-0.23922]), array([-0.23276]), array([-0.23276]), array([-0.22198]), array([-0.24569]), array([-0.2
2845]), array([-0.23707]), array([-0.2306]), array([-0.2306]), array([-0.23707]), array([-0.24353]), array([-0.23707]), array
([-0.22198]), array([-0.040948]), array([0.032328]), array([0.088362]), array([0.073276]), array([-0.1681]), array([-0.1853
4]), array([-0.15948]), array([-0.17888]), array([-0.18966]), array([-0.23276]), array([-0.23707]), array([-0.27802]), array
([-0.27802]), array([-0.31034]), array([-0.32112]), array([-0.32112]), array([-0.33621]), array([-0.32974]), array([-0.3362
1]), array([-0.32328]), array([-0.32328]), array([-0.33621]), array([-0.32328]), array([-0.32112]), array([-0.3125]), array
([-0.32112]), array([-0.31466]), array([-0.29957]), array([-0.32328]), array([-0.31034]), array([-0.32112]), array([-0.3060
3]), array([-0.30603]), array([-0.3125]), array([-0.32328]), array([-0.30603]), array([-0.30819]), array([-0.31466]), array
([-0.30603]), array([-0.29741]), array([-0.28017]), array([-0.27586]), array([-0.26078]), array([-0.23707]), array([-0.2370
```

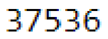
```
m=np.array(c)

def countlist(valley,peak):

   return[sub[item] for item in range(len(h))

                for sub in[valley,peak]]


com=(countlist(find_valley(m),find_peaks(m)))

type(com)

g=np.array(com)

print(g)
```



```
[[ 0.034483]
 [-0.68959 ]
 [ 0.015086]
 ...
 [-0.875   ]
 [-0.30255 ]
 [-0.36853 ]]
```

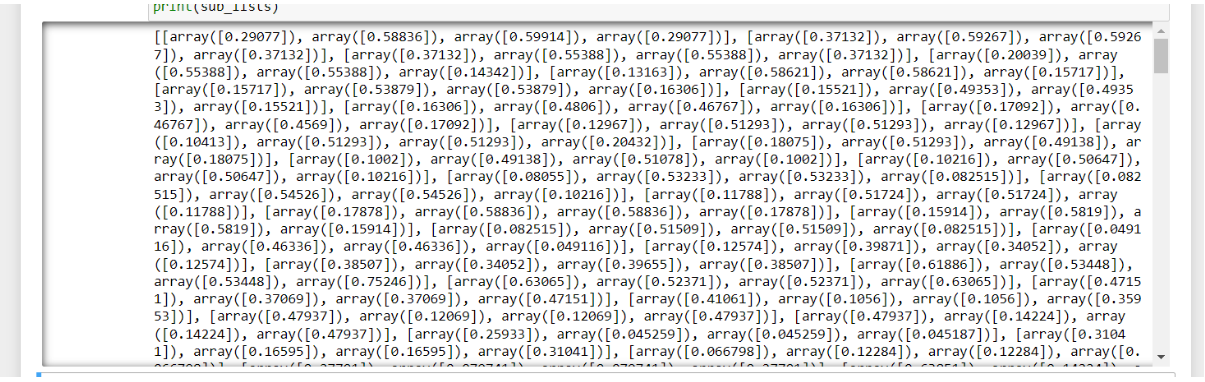
```
n=len(g)+len(h)

C=20

temp=[]

cfc=0

i=0

fc=0

lam=0

lc=0

msc=0
```

```
print(n)
alg=[]
for i in range(1,n,4):
    if((C-m[i])*(C-m[i-1])>0):
        cfc=cfc+1
        if(cfc==2):
            fc=m[i]
            cfc=0
            if((C1*m[i]<=m[i]<=C)and (fc!=0)):
                lam=m[i+1]
                if((m[i+1]<=C2)and (lam!=0)):
                    lc=m[i+2]
                    if((m[i+2]>0)and(lc!=0)):
                        msc=m[i+3]
                        alg.append(fc)
                        alg.append(lam)
                        alg.append(lc)
                        alg.append(msc)
                        cfc==0
                    else:
                        i=i-1
                else:
                    i=i-2
```



```
37536
```

```
def divide_list(lst, size):
    return [lst[i:i+size] for i in range(0, len(lst), size)]
sub_lists = divide_list(alg, 4)
print(sub_lists)
```

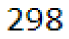


```
print(sub_lists)
```

```
[[array([0.29077]), array([0.58836]), array([0.59914]), array([0.29077])], [array([0.37132]), array([0.59267]), array([0.5926
7]), array([0.37132])], [array([0.37132]), array([0.55388]), array([0.55388]), array([0.37132])], [array([0.20039]), array
([0.55388]), array([0.55388]), array([0.14342])], [array([0.13163]), array([0.58621]), array([0.58621]), array([0.15717])],
[array([0.15717]), array([0.53879]), array([0.53879]), array([0.16306])], [array([0.15521]), array([0.49353]), array([0.4935
3]), array([0.15521])], [array([0.16306]), array([0.4806]), array([0.46767]), array([0.16306])], [array([0.17092]), array([0.
46767]), array([0.4569]), array([0.17092])], [array([0.12967]), array([0.51293]), array([0.51293]), array([0.12967])], [array
([0.10413]), array([0.51293]), array([0.51293]), array([0.20432])], [array([0.18075]), array([0.51293]), array([0.49138]), ar
ray([0.18075])], [array([0.1002]), array([0.49138]), array([0.51078]), array([0.1002])], [array([0.10216]), array([0.50647]),
array([0.50647]), array([0.10216])], [array([0.08055]), array([0.53233]), array([0.53233]), array([0.082515])], [array([0.082
515]), array([0.54526]), array([0.54526]), array([0.10216])], [array([0.11788]), array([0.51724]), array([0.51724]), array
([0.11788])], [array([0.17878]), array([0.58836]), array([0.58836]), array([0.17878])], [array([0.15914]), array([0.5819]), a
rray([0.5819]), array([0.15914])], [array([0.082515]), array([0.51509]), array([0.51509]), array([0.082515])], [array([0.0491
16]), array([0.46336]), array([0.46336]), array([0.049116])], [array([0.12574]), array([0.39871]), array([0.34052]), array
([0.12574])], [array([0.38507]), array([0.34052]), array([0.39655]), array([0.38507])], [array([0.61886]), array([0.53448]),
array([0.53448]), array([0.75246])], [array([0.63065]), array([0.52371]), array([0.52371]), array([0.63065])], [array([0.4715
1]), array([0.37069]), array([0.37069]), array([0.47151])], [array([0.41061]), array([0.1056]), array([0.1056]), array([0.359
53])], [array([0.47937]), array([0.12069]), array([0.12069]), array([0.47937])], [array([0.47937]), array([0.14224]), array
([0.14224]), array([0.47937])], [array([0.25933]), array([0.045259]), array([0.045259]), array([0.045187])], [array([0.3104
1]), array([0.16595]), array([0.16595]), array([0.31041])], [array([0.066798]), array([0.12284]), array([0.12284]), array([0.
```

cc=len(sub_lists)

print(cc)



```
298
```

cd=[]

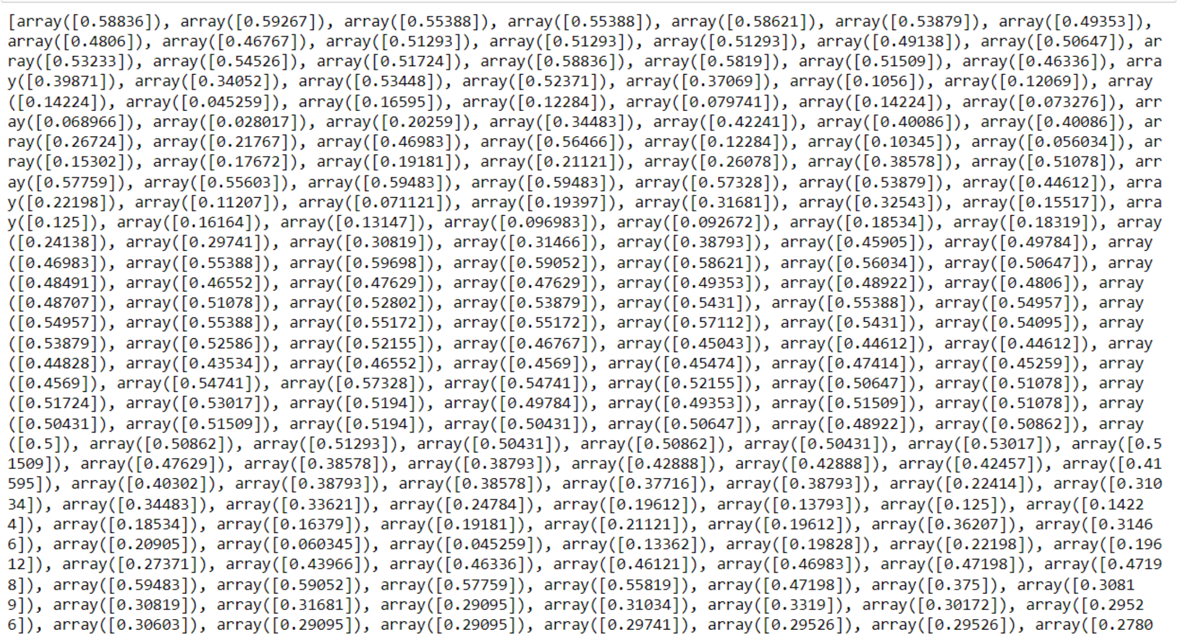
lad=[]

for i in range(cc):

    cd.append(sub_lists[i][0])

    lad.append(sub_lists[i][1])

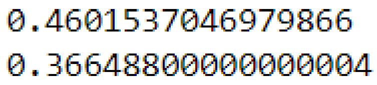
print(lad)



```
[array([0.58836]), array([0.59267]), array([0.55388]), array([0.55388]), array([0.58621]), array([0.53879]), array([0.49353]),
array([0.4806]), array([0.46767]), array([0.51293]), array([0.51293]), array([0.51293]), array([0.49138]), array([0.50647]), ar
ray([0.53233]), array([0.54526]), array([0.51724]), array([0.58836]), array([0.5819]), array([0.51509]), array([0.46336]), arra
y([0.39871]), array([0.34052]), array([0.53448]), array([0.52371]), array([0.37069]), array([0.1056]), array([0.12069]), array
([0.14224]), array([0.045259]), array([0.16595]), array([0.12284]), array([0.079741]), array([0.14224]), array([0.073276]), arr
ay([0.068966]), array([0.028017]), array([0.20259]), array([0.34483]), array([0.42241]), array([0.40086]), array([0.40086]), ar
ray([0.26724]), array([0.21767]), array([0.46983]), array([0.56466]), array([0.12284]), array([0.10345]), array([0.056034]), ar
ray([0.15302]), array([0.17672]), array([0.19181]), array([0.21121]), array([0.26078]), array([0.38578]), array([0.51078]), arr
ay([0.57759]), array([0.55603]), array([0.59483]), array([0.59483]), array([0.57328]), array([0.53879]), array([0.44612]), arra
y([0.22198]), array([0.11207]), array([0.071121]), array([0.19397]), array([0.31681]), array([0.32543]), array([0.15517]), arra
y([0.125]), array([0.16164]), array([0.13147]), array([0.096983]), array([0.092672]), array([0.18534]), array([0.18319]), array
([0.24138]), array([0.29741]), array([0.30819]), array([0.31466]), array([0.38793]), array([0.45905]), array([0.49784]), array
([0.46983]), array([0.55388]), array([0.59698]), array([0.59052]), array([0.58621]), array([0.56034]), array([0.50647]), array
([0.48491]), array([0.46552]), array([0.47629]), array([0.47629]), array([0.49353]), array([0.48922]), array([0.4806]), array
([0.48707]), array([0.51078]), array([0.52802]), array([0.53879]), array([0.5431]), array([0.55388]), array([0.54957]), array
([0.54957]), array([0.55388]), array([0.55172]), array([0.55172]), array([0.57112]), array([0.5431]), array([0.54095]), array
([0.53879]), array([0.52586]), array([0.52155]), array([0.46767]), array([0.45043]), array([0.44612]), array([0.44612]), array
([0.44828]), array([0.43534]), array([0.46552]), array([0.4569]), array([0.45474]), array([0.47414]), array([0.45259]), array
([0.4569]), array([0.54741]), array([0.57328]), array([0.54741]), array([0.52155]), array([0.50647]), array([0.51078]), array
([0.51724]), array([0.53017]), array([0.5194]), array([0.49784]), array([0.49353]), array([0.51509]), array([0.51078]), array
([0.50431]), array([0.51509]), array([0.5194]), array([0.50431]), array([0.50647]), array([0.48922]), array([0.50862]), array
([0.5]), array([0.50862]), array([0.51293]), array([0.50431]), array([0.50862]), array([0.50431]), array([0.53017]), array([0.5
1509]), array([0.47629]), array([0.38578]), array([0.38793]), array([0.42888]), array([0.42888]), array([0.42457]), array([0.41
595]), array([0.40302]), array([0.38793]), array([0.38578]), array([0.37716]), array([0.38793]), array([0.22414]), array([0.310
34]), array([0.34483]), array([0.33621]), array([0.24784]), array([0.19612]), array([0.13793]), array([0.125]), array([0.1422
4]), array([0.18534]), array([0.16379]), array([0.19181]), array([0.21121]), array([0.19612]), array([0.36207]), array([0.3146
6]), array([0.20905]), array([0.060345]), array([0.045259]), array([0.13362]), array([0.19828]), array([0.22198]), array([0.196
12]), array([0.27371]), array([0.43966]), array([0.46336]), array([0.46121]), array([0.46983]), array([0.47198]), array([0.4719
8]), array([0.59483]), array([0.59052]), array([0.57759]), array([0.55819]), array([0.47198]), array([0.375]), array([0.3081
9]), array([0.30819]), array([0.31681]), array([0.29095]), array([0.31034]), array([0.3319]), array([0.30172]), array([0.2952
6]), array([0.30603]), array([0.29095]), array([0.29095]), array([0.29741]), array([0.29526]), array([0.29526]), array([0.2780
```

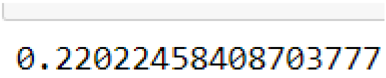
cm=np.average(cd)

```
print(cm)

lam=np.average(lad)

print(lam)
```



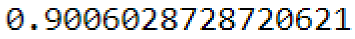
```
0.4601537046979866
0.36648800000000004
```

```
sd=np.std(cd)

print(sd)
```



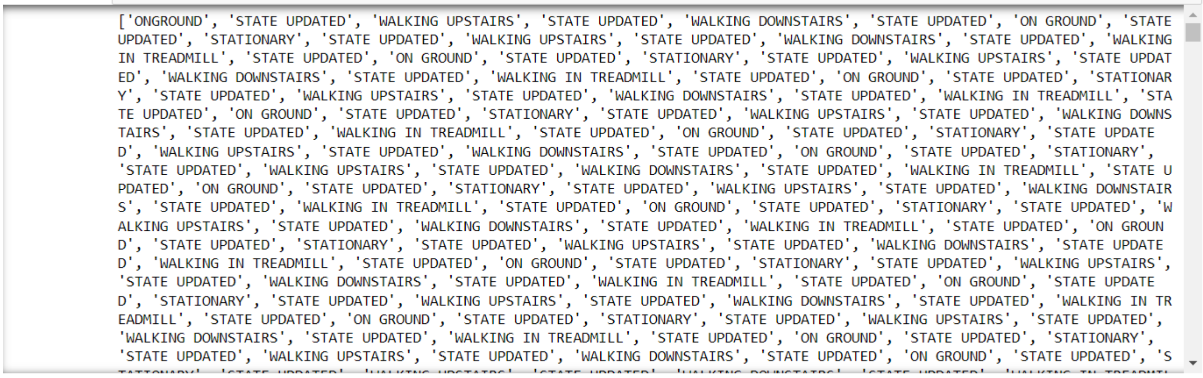
```
0.22022458408703777
```

```
alpha=cm+(2*sd)

print(alpha)
```



```
0.9006028728720621
```

```
C2=3

result=["ONGROUND"]

cmax=max(c)

cmin=min(c)

dif=abs(cmax-cmin)


for i in range(cc):

   if(lad[i]!=0):

      if(lad[i]>0):

         result.append("STATE UPDATED")

         result.append("WALKING UPSTAIRS")

      if(lad[i]>lam):

         result.append("STATE UPDATED")

         result.append("WALKING DOWNSTAIRS")

      if((cd[i]>alpha)and(result[i-1]!="WALKING IN TREADMILL")):

          result.append("STATE UPDATED")
          result.append("TRANSITION STATE")
        if((cd[i]>alpha)and(result[i-1]!="WALKING UPSTAIRS")or(result[i-1]!="WALKING DOWNSTAIRS")):
          result.append("STATE UPDATED")
          result.append("WALKING IN TREADMILL")
        if(cd[i]<=alpha):
          result.append("STATE UPDATED")
          result.append("ON GROUND")
    if(dif<C2):
        result.append("STATE UPDATED")
        result.append("STATIONARY")


print(result)
```



```
['ONGROUND', 'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNSTAIRS', 'STATE UPDATED', 'ON GROUND', 'STATE
UPDATED', 'STATIONARY', 'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNSTAIRS', 'STATE UPDATED', 'WALKING
IN TREADMILL', 'STATE UPDATED', 'ON GROUND', 'STATE UPDATED', 'STATIONARY', 'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDAT
ED', 'WALKING DOWNSTAIRS', 'STATE UPDATED', 'WALKING IN TREADMILL', 'STATE UPDATED', 'ON GROUND', 'STATE UPDATED', 'STATIONAR
Y', 'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNSTAIRS', 'STATE UPDATED', 'WALKING IN TREADMILL', 'STA
TE UPDATED', 'ON GROUND', 'STATE UPDATED', 'STATIONARY', 'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNS
TAIRS', 'STATE UPDATED', 'WALKING IN TREADMILL', 'STATE UPDATED', 'ON GROUND', 'STATE UPDATED', 'STATIONARY', 'STATE UPDATE
D', 'WALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNSTAIRS', 'STATE UPDATED', 'ON GROUND', 'STATE UPDATED', 'STATIONARY',
'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNSTAIRS', 'STATE UPDATED', 'WALKING IN TREADMILL', 'STATE U
PDATED', 'ON GROUND', 'STATE UPDATED', 'STATIONARY', 'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNSTAIR
S', 'STATE UPDATED', 'WALKING IN TREADMILL', 'STATE UPDATED', 'ON GROUND', 'STATE UPDATED', 'STATIONARY', 'STATE UPDATED', 'W
ALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNSTAIRS', 'STATE UPDATED', 'WALKING IN TREADMILL', 'STATE UPDATED', 'ON GROUN
D', 'STATE UPDATED', 'STATIONARY', 'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNSTAIRS', 'STATE UPDATE
D', 'WALKING IN TREADMILL', 'STATE UPDATED', 'ON GROUND', 'STATE UPDATED', 'STATIONARY', 'STATE UPDATED', 'WALKING UPSTAIRS',
'STATE UPDATED', 'WALKING DOWNSTAIRS', 'STATE UPDATED', 'WALKING IN TREADMILL', 'STATE UPDATED', 'ON GROUND', 'STATE UPDATE
D', 'STATIONARY', 'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNSTAIRS', 'STATE UPDATED', 'WALKING IN TR
EADMILL', 'STATE UPDATED', 'ON GROUND', 'STATE UPDATED', 'STATIONARY', 'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDATED',
'WALKING DOWNSTAIRS', 'STATE UPDATED', 'WALKING IN TREADMILL', 'STATE UPDATED', 'ON GROUND', 'STATE UPDATED', 'STATIONARY',
'STATE UPDATED', 'WALKING UPSTAIRS', 'STATE UPDATED', 'WALKING DOWNSTAIRS', 'STATE UPDATED', 'ON GROUND', 'STATE UPDATED', 'S
```