```
import numpy as np
import pandas as pd
```

```
mydict={}
```

```
n=100000
```

```
for i in range(n):
    mydict[i]=i**2
mydict
```

```
 943: 889249,
 944: 891136,
 945: 893025,
 946: 894916,
 947: 896809,
 948: 898704,
 949: 900601,
 950: 902500,
 951: 904401,
 952: 906304,
 953: 908209,
 954: 910116,
 955: 912025,
 956: 913936,
 957: 915849,
 958: 917764,
 959: 919681,
 960: 921600,
 961: 923521,
 962: 925444,
 963: 927369,
 964: 929296,
 965: 931225,
 966: 933156,
 967: 935089,
 968: 937024,
 969: 938961,
 970: 940900,
 971: 942841,
 972: 944784,
 973: 946729,
 974: 948676,
 975: 950625,
 976: 952576,
 977: 954529,
 978: 956484,
 979: 958441,
 980: 960400,
 981: 962361,
 982: 964324,
 983: 966289,
 984: 968256,
 985: 970225,
 986: 972196,
 987: 974169,
 988: 976144,
 989: 978121,
 990: 980100,
 991: 982081,
 992: 984064,
 993: 986049,
 994: 988036,
 995: 990025,
 996: 992016,
 997: 994009,
 998: 996004,
 999: 998001,
 ...}
```

```
myseries=pd.Series(mydict)
myseries
```

```
0                 0
1                 1
2                 4
3                 9
4                16
            ...
99995    9999000025
99996    9999200016
99997    9999400009
99998    9999600004
99999    9999800001
Length: 100000, dtype: int64
```

```
m=1000
```

```
n
```

```
100000
```

```
arr=np.random.randint(0,n,m)
arr
```

```
       38460, 15369, 93335, 54694, 54717, 26792,  5398, 15970, 33687,
       45763, 74154, 86932, 21164, 61808, 87895, 79410, 71110, 94443,
       37654, 63034, 56779,  4896, 93230, 71111, 41198,  2626, 71789,
        3208, 17134, 91764, 33419, 91468, 65399, 23504, 74777, 33368,
       77277, 98404, 84318, 92044, 24282, 22647, 17849, 63148, 28042,
       97801,  5259, 77543, 37016, 90861, 97318, 37654, 96664, 92811,
       97098, 28469, 48586, 52887, 87764, 40304, 53359, 56841, 48618,
       33690, 28246, 26031, 47672,    15, 65049, 16380, 37842, 15158,
         597, 59941, 98442, 68188, 73910, 43497, 97389, 34052, 13403,
       31916, 62947, 81901, 93965,   429, 66173,  6360, 65588, 74806,
       31344, 22602,  7799,  4704, 26727, 12181, 38968, 28176,  4820,
       74533, 16976, 94214, 36798, 34215, 44683, 20229, 53745, 34961,
       85567, 46125, 75214, 17751, 19553, 24860, 31421, 27787, 27242,
       28406, 45013, 74740, 65843, 28106, 63378, 66612, 57795, 16731,
       60167, 25778, 35045, 53741, 20544, 79928, 50363, 25004, 62879,
       32264, 38382, 75478, 94064, 85309, 65861, 38800, 72212, 85177,
       48955, 61099, 91701, 75754, 47710, 83532,  6819, 30711, 87066,
       76538, 96056, 89496, 55248, 28924, 85442, 91292,  7888, 65999,
       99393, 65270, 25102, 99023, 11944, 75511,  6201, 72876, 15351,
       62081, 62049, 57575, 51444, 58876,  5332, 38415, 29341, 31639,
       46169, 64103, 14724, 91650,   507, 62561,  3266, 16619,  3913,
       11924, 35350, 51141, 49867, 35227, 18477, 40343,  2766, 93573,
       57111, 83583, 22036,  5261,  3818, 72743, 95841, 17759, 55296,
       39233,  8598, 39645, 14935, 22185, 52016, 28279, 40994, 12109,
       18363, 71939, 76324, 22849, 59296, 67298, 77575, 49408, 86388,
       29188, 93835, 20559, 84441, 96003, 39110, 99556, 80162, 12436,
       80159, 39731, 22164, 54842, 45273, 57252, 35466, 76917, 89345,
       75222, 21803, 81189, 20897, 34704, 10419, 90555, 33444, 32533,
       24096, 61120, 96750, 69708, 46231, 25776, 92677, 72619, 50142,
       64350, 81583, 89736, 84316,  4695, 41761, 71078, 97212, 37572,
       11366,   438, 11528, 22177, 54121, 78656, 59623, 26655, 19268,
       85208, 20236, 83536, 17403, 57683, 60418, 37356,  8758, 50171,
       75744, 47415, 90260, 12765, 21615, 45118, 83467, 40624, 18553,
       94046, 62049, 92821, 55861,   384, 95842, 87504, 70678, 92148,
       63907, 90993, 86390, 44835, 47425, 15063,  1271, 91084, 36062,
       25979,  8523, 21338, 54890, 37219, 47674, 53058,  6643, 87983,
       17828,   156, 36416, 69799, 66821,  8736, 66543, 57884, 11272,
       81689, 12998, 14217, 87875, 11652, 49461, 30103,  9966, 68193,
       21235, 57127, 49669, 84031,  5916, 47215, 28016, 34026, 19335,
       17089, 70697,  4748, 13380, 90066, 40089, 87429, 89562, 96120,
       54460, 67350, 33406, 54979, 70344, 36834, 30616,  2895, 59134,
       59538, 74438, 79894, 37448, 62863, 11790, 90169, 57818, 18978,
       20987, 40066, 35882, 68371, 94405, 38080, 94461, 74723, 18310,
       34715, 11779, 79299, 72926, 87680, 62850, 24560, 89272, 30125,
        7957, 80120, 66408, 52275, 57693,  1376, 24651, 18540, 51274,
       79048, 81705, 31034, 47126, 57808, 39901, 72858, 58174, 84642,
       49734,  6991, 41603, 26522, 43961, 54347, 76931, 13172, 25033,
       96548, 10584, 10839,  5462, 17846, 14421, 18166, 53422, 49105,
       57659, 16705, 89322, 27288, 93662, 40011, 44699, 19031, 49742,
       65100, 85972, 19418, 55286, 30540, 94958,  6906, 10587, 54758,
       51857, 89326, 29026, 54168, 64118, 96314, 84005,    21, 57580,
       18289, 71470, 24258, 85498, 50212, 31884, 69889, 83686, 58963,
       77897, 40094, 79377, 25288, 53257, 67576, 95262, 24299, 25839,
       99493, 21870, 52732, 10898, 36648, 32401, 61765, 72429, 10127,
       29075, 64431, 80204, 41132, 44945, 90831, 55896,  4599, 49565,
       11803, 22995,  1979, 90339, 80046, 45355, 18837, 58395, 48605,
       11903, 43968, 88757, 66009, 39256, 50317,  5363, 81743, 43907,
       80886])
```

```
%%time
for i in arr :
    i in mydict
```

```
CPU times: user 1.1 ms, sys: 12 µs, total: 1.11 ms
Wall time: 1.45 ms
```

```
%%time
for i in arr :
    i in myseries
```

```
CPU times: user 14.4 ms, sys: 0 ns, total: 14.4 ms
Wall time: 16.6 ms
```

in seraching pandas is slower, while operations pandas faster

```
%%timeit
sum(mydict.values)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-23-42f8bf0a9a4d> in <module>()
----> 1 get_ipython().run_cell_magic('timeit', '', 'sum(mydict.values)')

3 frames
<decorator-gen-52> in timeit(self, line, cell)

/usr/local/lib/python3.7/dist-packages/IPython/core/magics/execution.py in
timeit(self, number)
    137         gc.disable()
    138         try:
--> 139             timing = self.inner(it, self.timer)
    140         finally:
    141             if gcold:

<magic-timeit> in inner(_it, _timer)

TypeError: 'builtin_function_or_method' object is not iterable
```

SEARCH STACK OVERFLOW

```
%%timeit
sum(mydict.values())
```

```
1000 loops, best of 5: 1.18 ms per loop
```

```
%%timeit
sum(myseries)
```

```
100 loops, best of 5: 10 ms per loop
```

```
%%timeit
np.sum(myseries)
```

```
10000 loops, best of 5: 140 µs per loop
```

```
%%timeit
sum(mydict.values)/n
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-4efb91b69b58> in <module>()
----> 1 get_ipython().run_cell_magic('timeit', '', 'sum(mydict.values)/n')
```

3 frames

```
<decorator-gen-52> in timeit(self, line, cell)

/usr/local/lib/python3.7/dist-packages/IPython/core/magics/execution.py in
timeit(self, number)
    137         gc.disable()
    138         try:
```

```
%%timeit
mean=sum(mydict.values())/n
var= sum((x-mean)**2 for x in mydict.values())
std=var**0.5
```

```
100 loops, best of 5: 17.9 ms per loop
```

```
%%timeit
mean=np.mean(myseries)
var=np.var(myseries)
std=np.std(myseries)
```

```
1000 loops, best of 5: 741 µs per loop
```

```
#pd series operations faster than dic , while search faster in dic
```

## MINI PROJECT NIFTY

```
pd.read_csv("nifty.csv")
```

| nifty | Open | High | Low | Close | Shares Traded | Turnover (Rs. Cr) |
|---|---|---|---|---|---|---|
| 01-Jan- | | | | | | |

```
pd.read_csv("nifty.csv",index_col=0).iloc[:,0]
```

```
nifty
01-Jan-19    10881.70
02-Jan-19    10868.85
03-Jan-19    10796.80
04-Jan-19    10699.70
07-Jan-19    10804.85
               ...
24-Dec-19    12269.25
26-Dec-19    12211.85
27-Dec-19    12172.90
30-Dec-19    12274.90
31-Dec-19    12247.10
Name: Open, Length: 245, dtype: float64
```

| 240 | 19 | 12269.25 | 12283.70 | 12202.10 | 12214.55 | 470290298 | 13864.56 |
|---|---|---|---|---|---|---|---|

```
nifty=pd.read_csv("nifty.csv",index_col=0).iloc[:,0]
nifty
```

```
nifty
01-Jan-19    10881.70
02-Jan-19    10868.85
03-Jan-19    10796.80
04-Jan-19    10699.70
07-Jan-19    10804.85
               ...
24-Dec-19    12269.25
26-Dec-19    12211.85
27-Dec-19    12172.90
30-Dec-19    12274.90
31-Dec-19    12247.10
Name: Open, Length: 245, dtype: float64
```

```
nifty.head()
```

```
nifty
01-Jan-19    10881.70
02-Jan-19    10868.85
03-Jan-19    10796.80
04-Jan-19    10699.70
07-Jan-19    10804.85
Name: Open, dtype: float64
```

```
nifty.head(10)
```

```
nifty
01-Jan-19    10881.70
02-Jan-19    10868.85
03-Jan-19    10796.80
04-Jan-19    10699.70
07-Jan-19    10804.85
08-Jan-19    10786.25
```

```
09-Jan-19    10862.40
10-Jan-19    10859.35
11-Jan-19    10834.75
14-Jan-19    10807.00
Name: Open, dtype: float64
```

```
nifty.tail()
```

```
nifty
24-Dec-19    12269.25
26-Dec-19    12211.85
27-Dec-19    12172.90
30-Dec-19    12274.90
31-Dec-19    12247.10
Name: Open, dtype: float64
```

```
np.mean(nifty)
```

```
11444.261836734699
```

```
  np.median(nifty)
```

```
11536.15
```

```
  np.var(nifty)
```

```
207359.746247647
```

```
np.std(nifty)
```

```
455.3677044407596
```

```
np.max(nifty)
```

```
12274.9
```

```
np.min(nifty)
```

```
10636.7
```

What fraction of days did market close higher than previous day

```
x=nifty[1:]
x
```

```
nifty
02-Jan-19    10868.85
03-Jan-19    10796.80
04-Jan-19    10699.70
07-Jan-19    10804.85
08-Jan-19    10786.25
```

```
                ...
24-Dec-19    12269.25
26-Dec-19    12211.85
27-Dec-19    12172.90
30-Dec-19    12274.90
31-Dec-19    12247.10
Name: Open, Length: 244, dtype: float64
```

```
y=nifty[:-1]
y
```

```
nifty
01-Jan-19    10881.70
02-Jan-19    10868.85
03-Jan-19    10796.80
04-Jan-19    10699.70
07-Jan-19    10804.85
                ...
23-Dec-19    12235.45
24-Dec-19    12269.25
26-Dec-19    12211.85
27-Dec-19    12172.90
30-Dec-19    12274.90
Name: Open, Length: 244, dtype: float64
```

```
x=nifty[1:]   [   nifty[1:].values     >  nifty[:-1].values    ]
x
```

```
nifty
07-Jan-19    10804.85
09-Jan-19    10862.40
16-Jan-19    10899.65
17-Jan-19    10920.85
21-Jan-19    10919.35
                ...
18-Dec-19    12197.00
19-Dec-19    12223.40
20-Dec-19    12266.45
24-Dec-19    12269.25
30-Dec-19    12274.90
Name: Open, Length: 123, dtype: float64
```

```
len(x)/len(nifty)
```

```
0.5020408163265306
```

compute moving average of last 5 days sub set data to only include data of friday

```
nifty.index[0]
```

```
'01-Jan-19'
```

```
d=pd.Timestamp(nifty.index[0])
```

```
d
```

```
Timestamp('2019-01-01 00:00:00')
```

```
d.day_of_week #monday 0 starts , tue 1,w 2, th 3 , fr 4 ,  sa 5 , sun 6
```

```
1
```

```
pd.Timestamp(nifty.index[1]).day_of_week
```

```
2
```

```
NewIndex=list(map(pd.Timestamp,nifty.index))
NewIndex
```

```
 Timestamp('2019-10-07 00:00:00'),
 Timestamp('2019-10-09 00:00:00'),
 Timestamp('2019-10-10 00:00:00'),
 Timestamp('2019-10-11 00:00:00'),
 Timestamp('2019-10-14 00:00:00'),
 Timestamp('2019-10-15 00:00:00'),
 Timestamp('2019-10-16 00:00:00'),
 Timestamp('2019-10-17 00:00:00'),
 Timestamp('2019-10-18 00:00:00'),
 Timestamp('2019-10-22 00:00:00'),
 Timestamp('2019-10-23 00:00:00'),
 Timestamp('2019-10-24 00:00:00'),
 Timestamp('2019-10-25 00:00:00'),
 Timestamp('2019-10-27 00:00:00'),
 Timestamp('2019-10-29 00:00:00'),
 Timestamp('2019-10-30 00:00:00'),
 Timestamp('2019-10-31 00:00:00'),
 Timestamp('2019-11-01 00:00:00'),
 Timestamp('2019-11-04 00:00:00'),
 Timestamp('2019-11-05 00:00:00'),
 Timestamp('2019-11-06 00:00:00'),
 Timestamp('2019-11-07 00:00:00'),
 Timestamp('2019-11-08 00:00:00'),
 Timestamp('2019-11-11 00:00:00'),
 Timestamp('2019-11-13 00:00:00'),
 Timestamp('2019-11-14 00:00:00'),
 Timestamp('2019-11-15 00:00:00'),
 Timestamp('2019-11-18 00:00:00'),
 Timestamp('2019-11-19 00:00:00'),
 Timestamp('2019-11-20 00:00:00'),
 Timestamp('2019-11-21 00:00:00'),
 Timestamp('2019-11-22 00:00:00'),
 Timestamp('2019-11-25 00:00:00'),
 Timestamp('2019-11-26 00:00:00'),
 Timestamp('2019-11-27 00:00:00'),
 Timestamp('2019-11-28 00:00:00'),
 Timestamp('2019-11-29 00:00:00'),
 Timestamp('2019-12-02 00:00:00'),
 Timestamp('2019-12-03 00:00:00'),
 Timestamp('2019-12-04 00:00:00'),
 Timestamp('2019-12-05 00:00:00'),
 Timestamp('2019-12-06 00:00:00'),
 Timestamp('2019-12-09 00:00:00'),
 Timestamp('2019-12-10 00:00:00'),
 Timestamp('2019-12-11 00:00:00'),
 Timestamp('2019-12-12 00:00:00'),
 Timestamp('2019-12-13 00:00:00'),
 Timestamp('2019-12-16 00:00:00'),
 Timestamp('2019-12-17 00:00:00'),
 Timestamp('2019-12-18 00:00:00'),
 Timestamp('2019-12-19 00:00:00'),
 Timestamp('2019-12-20 00:00:00'),
 Timestamp('2019-12-23 00:00:00'),
 Timestamp('2019-12-24 00:00:00'),
 Timestamp('2019-12-26 00:00:00'),
 Timestamp('2019-12-27 00:00:00'),
 Timestamp('2019-12-30 00:00:00'),
 Timestamp('2019-12-31 00:00:00')]
```

```
newnifty=pd.Series(nifty,index=NewIndex)
newnifty #nan error
```

```
2019-01-01   NaN
2019-01-02   NaN
2019-01-03   NaN
2019-01-04   NaN
2019-01-07   NaN
              ..
2019-12-24   NaN
2019-12-26   NaN
2019-12-27   NaN
2019-12-30   NaN
2019-12-31   NaN
Name: Open, Length: 245, dtype: float64
```

```
newnifty=pd.Series(nifty.values,index=NewIndex)
newnifty
```

```
2019-01-01    10881.70
2019-01-02    10868.85
2019-01-03    10796.80
2019-01-04    10699.70
2019-01-07    10804.85
                ...
2019-12-24    12269.25
2019-12-26    12211.85
2019-12-27    12172.90
2019-12-30    12274.90
2019-12-31    12247.10
Length: 245, dtype: float64
```

```
newnifty.index[0]
```

```
Timestamp('2019-01-01 00:00:00')
```

```
newnifty.rolling('5d') #when time stamp as index
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/window/rolling.py:287: FutureWarni
  if getattr(self, attr_name, None) is not None and attr_name[0] != "_"
```

```
Rolling [window=5d,min_periods=1,center=False,win_type=freq,axis=0,method=single]
```

```
x=newnifty.rolling('5d').mean()
x
```

```
2019-01-01    10881.700000
2019-01-02    10875.275000
2019-01-03    10849.116667
2019-01-04    10811.762500
2019-01-07    10767.116667
                  ...
2019-12-24    12257.050000
2019-12-26    12238.850000
2019-12-27    12222.362500
2019-12-30    12219.883333
2019-12-31    12231.633333
Length: 245, dtype: float64
```

```
dow=newnifty.copy() #copy right ,  not =
dow
```

```
2019-01-01    10881.70
2019-01-02    10868.85
2019-01-03    10796.80
2019-01-04    10699.70
2019-01-07    10804.85
                ...
2019-12-24    12269.25
2019-12-26    12211.85
2019-12-27    12172.90
2019-12-30    12274.90
2019-12-31    12247.10
Length: 245, dtype: float64
```

```
for i in newnifty.index :
    dow[i]= i.day_of_week
dow
```

```
2019-01-01    1.0
2019-01-02    2.0
2019-01-03    3.0
2019-01-04    4.0
2019-01-07    0.0
             ...
2019-12-24    1.0
2019-12-26    3.0
2019-12-27    4.0
2019-12-30    0.0
2019-12-31    1.0
Length: 245, dtype: float64
```

```
newnifty
```

```
2019-01-01    10881.70
```

```
2019-01-02    10868.85
2019-01-03    10796.80
2019-01-04    10699.70
2019-01-07    10804.85
                ...
2019-12-24    12269.25
2019-12-26    12211.85
2019-12-27    12172.90
2019-12-30    12274.90
2019-12-31    12247.10
Length: 245, dtype: float64
```

```
newnifty[dow.values==4]
```

```
2019-01-04    10699.70
2019-01-11    10834.75
2019-01-18    10914.85
2019-01-25    10859.75
2019-02-01    10851.35
2019-02-08    11023.50
2019-02-15    10780.25
2019-02-22    10782.70
2019-03-01    10842.65
2019-03-08    11038.85
2019-03-15    11376.85
2019-03-22    11549.20
2019-03-29    11625.45
2019-04-05    11638.40
2019-04-12    11612.85
2019-04-26    11683.75
2019-05-03    11722.60
2019-05-10    11314.15
2019-05-17    11261.90
2019-05-24    11748.00
2019-05-31    11999.80
2019-06-07    11865.20
2019-06-14    11910.10
2019-06-21    11827.60
2019-06-28    11861.15
2019-07-05    11964.75
2019-07-12    11601.15
2019-07-19    11627.95
2019-07-26    11247.45
2019-08-02    10930.30
2019-08-09    11087.90
2019-08-16    11043.65
2019-08-23    10699.60
2019-08-30    10987.80
2019-09-06    10883.80
2019-09-13    10986.80
2019-09-20    10746.80
2019-09-27    11556.35
2019-10-04    11388.45
2019-10-11    11257.70
2019-10-18    11580.30
2019-10-25    11646.15
2019-11-01    11886.60
2019-11-08    11987.15
2019-11-15    11904.20
```

```
2019-11-22    11967.30
2019-11-29    12146.20
2019-12-06    12047.35
2019-12-13    12026.40
2019-12-20    12266.45
2019-12-27    12172.90
dtype: float64
```

## my alternate

```
y=np.array
y
```

```
<function numpy.array>
```

```
for x in range(0,len(nifty)-5) :
    np.mean(nifty[x-5:x])

```