Assignment2Report

February 10, 2019

1 CS301 - Assignment 2

1.0.1 Lists vs Dictionaries in Python

By Joshua Swick, Evan Minor, and David Vandiver

Assignment Prompt Experimentally determine the running time of list and dictionary operations in Python. Write a report explaining what you found, and draw whatever conclusions you can about how lists and dictionaries might be structured ino order to give this behavior. You will probably find it useful to write a function that produces a .csv file of data about the behavior of a given function and then to use Excel to plot the data and find a curve that fits it.

We compared this by measuring the runtime of functions doing similar operations on lists and dictionary data structures of increasing size. The functions are listed below and their runtimes graphed accordingly.

1.1 Prepare Data (see ./Assignment2.py)

```
In [1]: from Assignment2 import *
        number_of_elements = 100000
        main()

In [2]: import matplotlib.pyplot as plt
        import pandas as pd

In [3]: !ls ./csvs/ ## Raw csv files
```

```
addFooMiddle_dict.csv                    getLastElement_dict.csv
addFooMiddle_list.csv                    getLastElement_list.csv
appendFoo_dict.csv                     getMiddleElement_dict.csv
appendFoo_list.csv                     getMiddleElement_list.csv
concatenateFooToEachElement_dict.csv  lengthOf_dict.csv
concatenateFooToEachElement_list.csv  lengthOf_list.csv
countNumOfElements_dict.csv              removeMiddleElement_dict.csv
countNumOfElements_list.csv              removeMiddleElement_list.csv
getFirstElement_dict.csv              sort_dict.csv
getFirstElement_list.csv              sort_list.csv
```

1.2 Compare Data

1.2.1 Append element to data structure

Dictionary

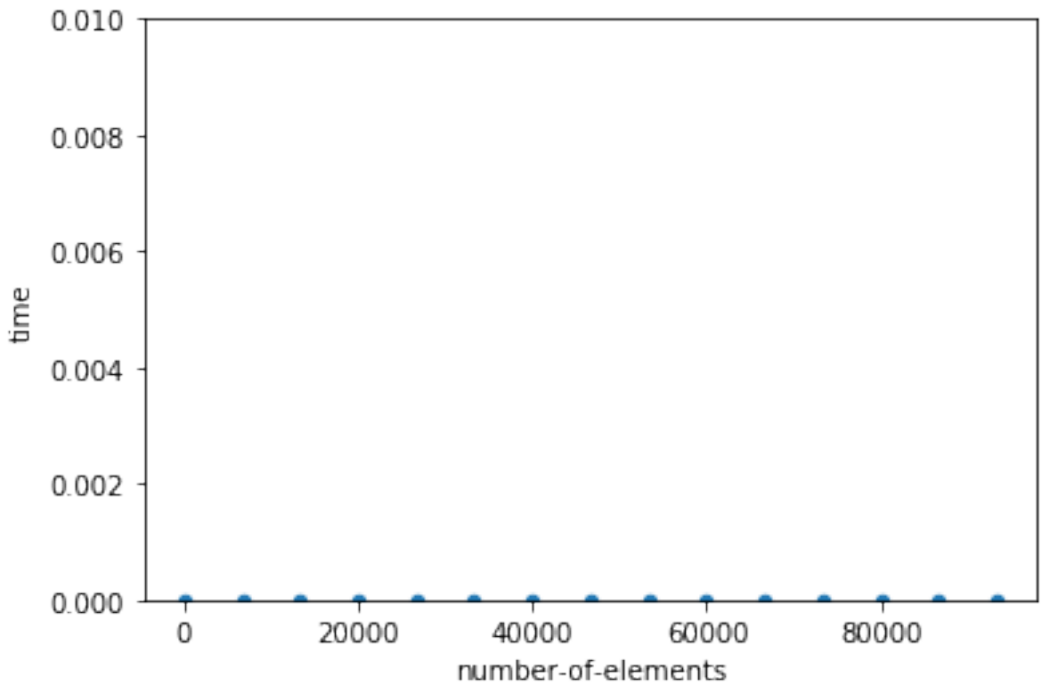
```
In [4]: file = "csvs/appendFoo_dict.csv"
        names = ['number-of-elements','time']
        dataset1 = pd.read_csv(file,names=names)
```

```
In [5]: print(dataset1)
```

```
    number-of-elements      time
0                    2  0.000004
1                 6668  0.000003
2                13334  0.000004
3                20000  0.000004
4                26666  0.000004
5                33332  0.000003
6                39998  0.000003
7                46664  0.000004
8                53330  0.000004
9                59996  0.000003
10               66662  0.000004
11               73328  0.000003
12               79994  0.000003
13               86660  0.000003
14               93326  0.000003
```

```
In [6]: ax1 = dataset1.plot(kind='scatter',x='number-of-elements',y='time')
        ax1.set_ylim(0,.01)
```

```
Out[6]: (0, 0.01)
```

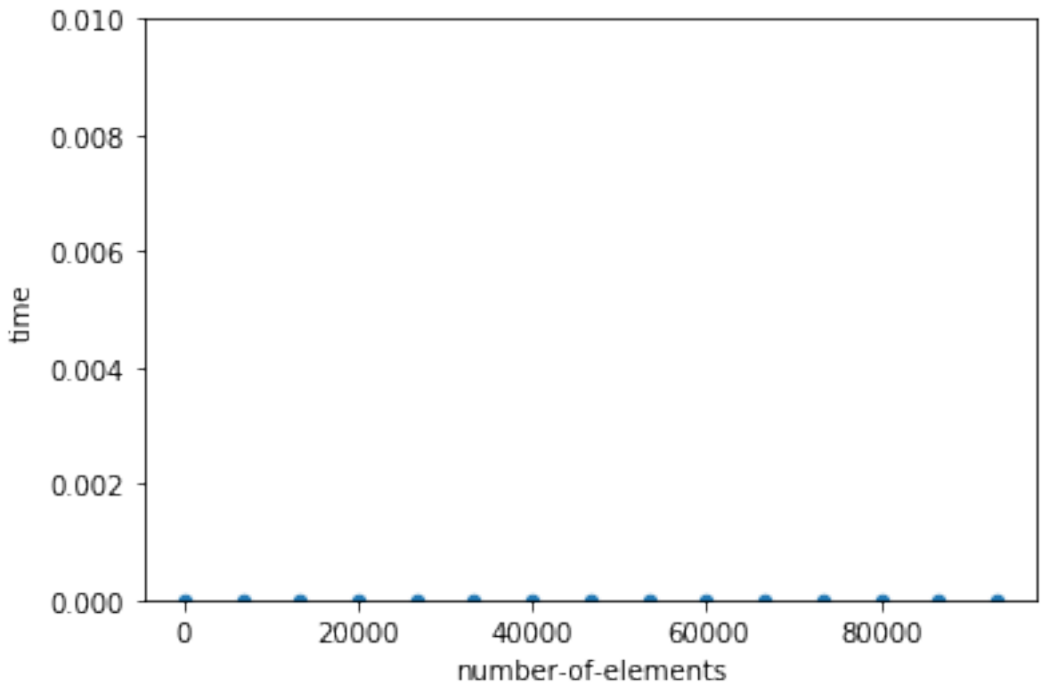


List

```
In [7]: file = "csvs/appendFoo_list.csv"
        names = ['number-of-elements','time']
        dataset2 = pd.read_csv(file,names=names)

In [8]: ax2 = dataset2.plot(kind='scatter',x='number-of-elements',y='time')
        ax2.set_ylim(0,.01)

Out[8]: (0, 0.01)
```



Explaination As of Python 3.7 dictionaries are ordered by insertion. The equivelent of appending to list would be to insert the element by key value pair as usual. This is a constant function.

Python must check each element in the list to find the end as the operation is linear.

1.2.2 Concatenate 'foo' to Each Element in the Data Structure

Dictionary

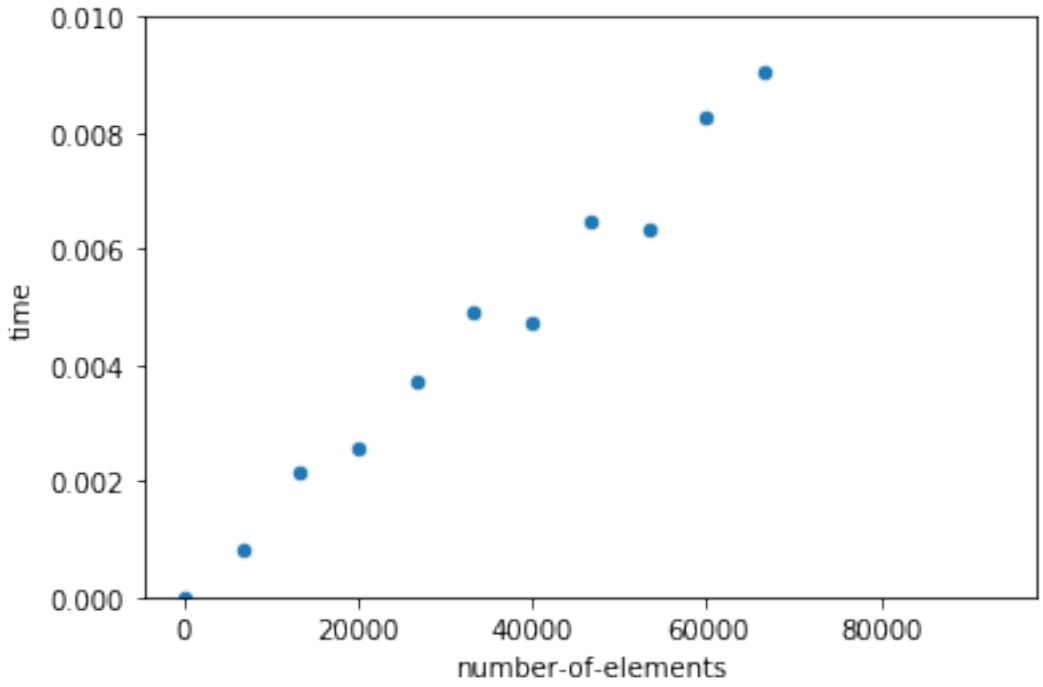
```
In [9]: file = "csvs/concatenateFooToEachElement_dict.csv"
        dataset3 = pd.read_csv(file,names=names)
        print(dataset3)
```

```
    number-of-elements      time
0                    1  0.000006
1                 6667  0.000821
2                13333  0.002174
3                19999  0.002583
4                26665  0.003731
5                33331  0.004907
6                39997  0.004710
7                46663  0.006483
8                53329  0.006334
9                59995  0.008263
10               66661  0.009065
11               73327  0.010756
```

```
12                79993  0.011292
13                86659  0.012216
14                93325  0.012706
```

```
In [10]: ax3 = dataset3.plot(kind='scatter',x='number-of-elements',y='time')
         ax3.set_ylim(0,.01)
```

```
Out[10]: (0, 0.01)
```



List

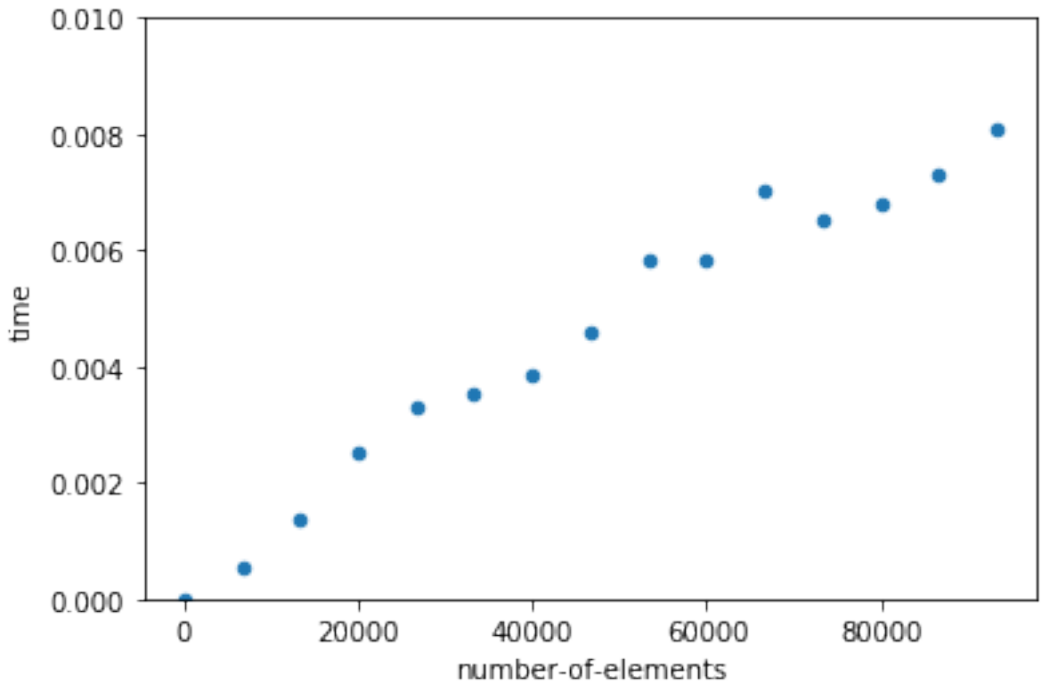
```
In [11]: file = "csvs/concatenateFooToEachElement_list.csv"
         dataset4 = pd.read_csv(file,names=names)
         print(dataset4)
```

```
    number-of-elements      time
0                    2  0.000005
1                 6668  0.000558
2                13334  0.001348
3                20000  0.002528
4                26666  0.003281
5                33332  0.003526
6                39998  0.003846
7                46664  0.004595
```

```
8                53330  0.005841
9                59996  0.005844
10               66662  0.007005
11               73328  0.006511
12               79994  0.006788
13               86660  0.007304
14               93326  0.008094
```

```
In [12]: ax4 = dataset4.plot(kind='scatter',x='number-of-elements',y='time')
         ax4.set_ylim(0,.01)
```

```
Out[12]: (0, 0.01)
```



Explanation Python must iterate through each element in the data structure which is a linear operation. Since the list and the dictionary are linear in this respect, it must add foo to each element individually before it can move on to the next element

1.2.3 Count the Number of Elements in the Data Structure

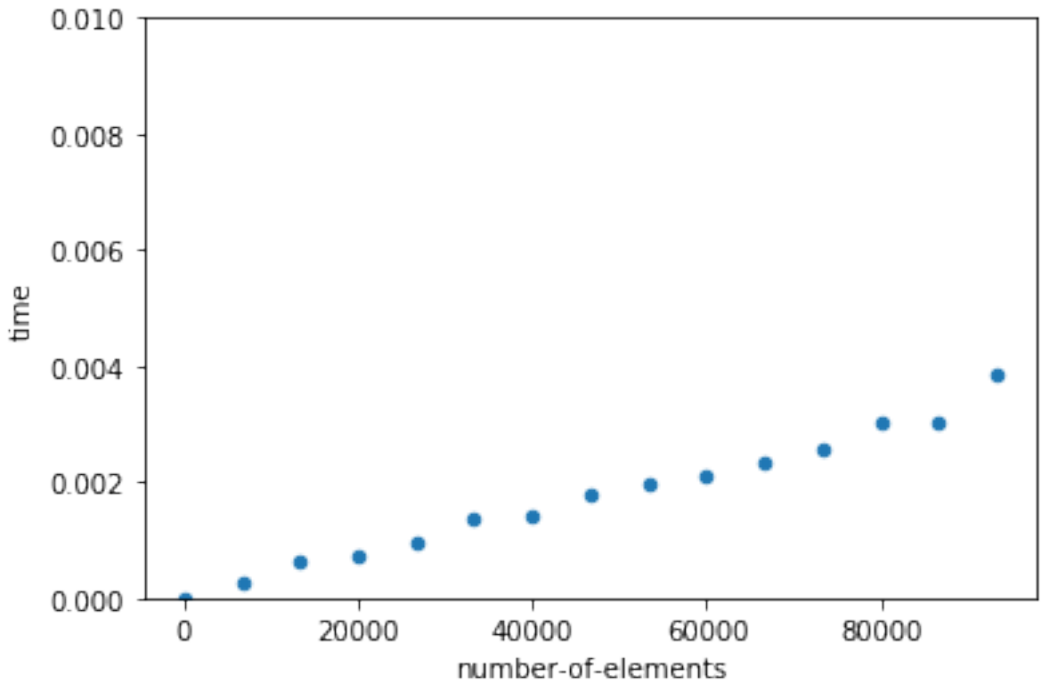
Dictionary

```
In [13]: file = "csvs/countNumOfElements_dict.csv"
         dataset5 = pd.read_csv(file,names=names)
         print(dataset5)
```

	number-of-elements	time
0	1	0.000003
1	6667	0.000248
2	13333	0.000639
3	19999	0.000745
4	26665	0.000972
5	33331	0.001357
6	39997	0.001429
7	46663	0.001792
8	53329	0.001959
9	59995	0.002106
10	66661	0.002316
11	73327	0.002550
12	79993	0.003011
13	86659	0.003011
14	93325	0.003853

In [14]:
```
ax5 = dataset5.plot(kind='scatter',x='number-of-elements',y='time')
ax5.set_ylim(0,.01)
```

Out[14]: (0, 0.01)



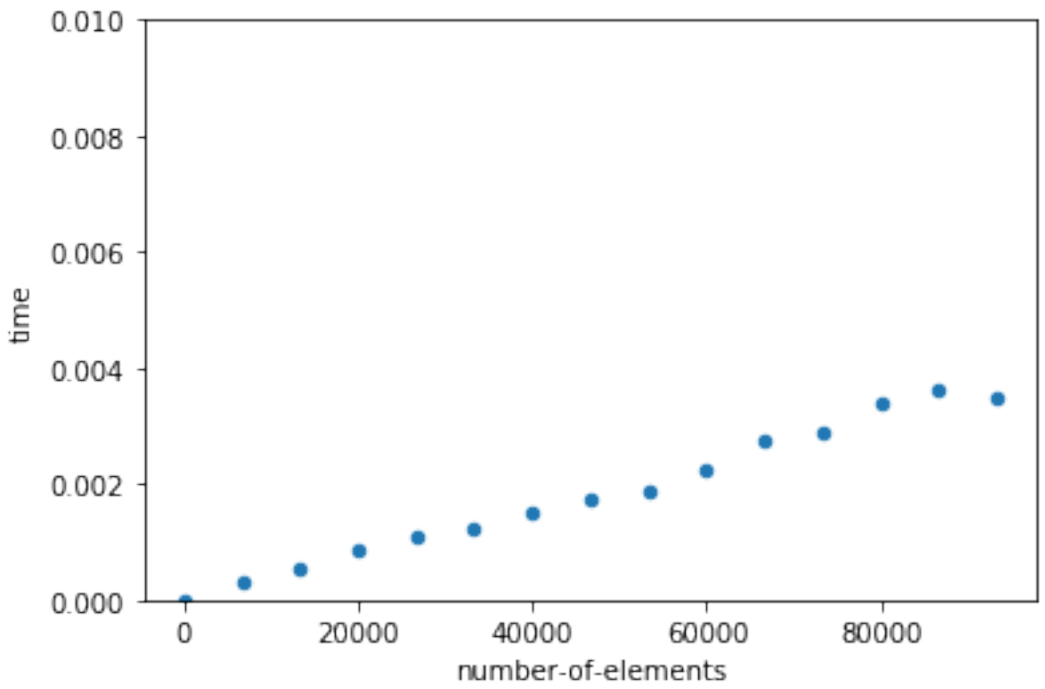
List

```
In [15]: file = "csvs/countNumOfElements_list.csv"
         dataset6 = pd.read_csv(file,names=names)
         print(dataset6)
```

```
    number-of-elements      time
0                    2  0.000002
1                 6668  0.000328
2                13334  0.000536
3                20000  0.000877
4                26666  0.001074
5                33332  0.001233
6                39998  0.001511
7                46664  0.001745
8                53330  0.001879
9                59996  0.002263
10               66662  0.002767
11               73328  0.002886
12               79994  0.003383
13               86660  0.003607
14               93326  0.003504
```

```
In [16]: ax6 = dataset6.plot(kind='scatter',x='number-of-elements',y='time')
         ax6.set_ylim(0,.01)
```

```
Out[16]: (0, 0.01)
```



Explaination Python must iterate through each element in the data structure while counting, which is a linear process. Since this is linear time will in increase in the typical mx+b fashion.

1.2.4 Get the First Element in the Data Structure

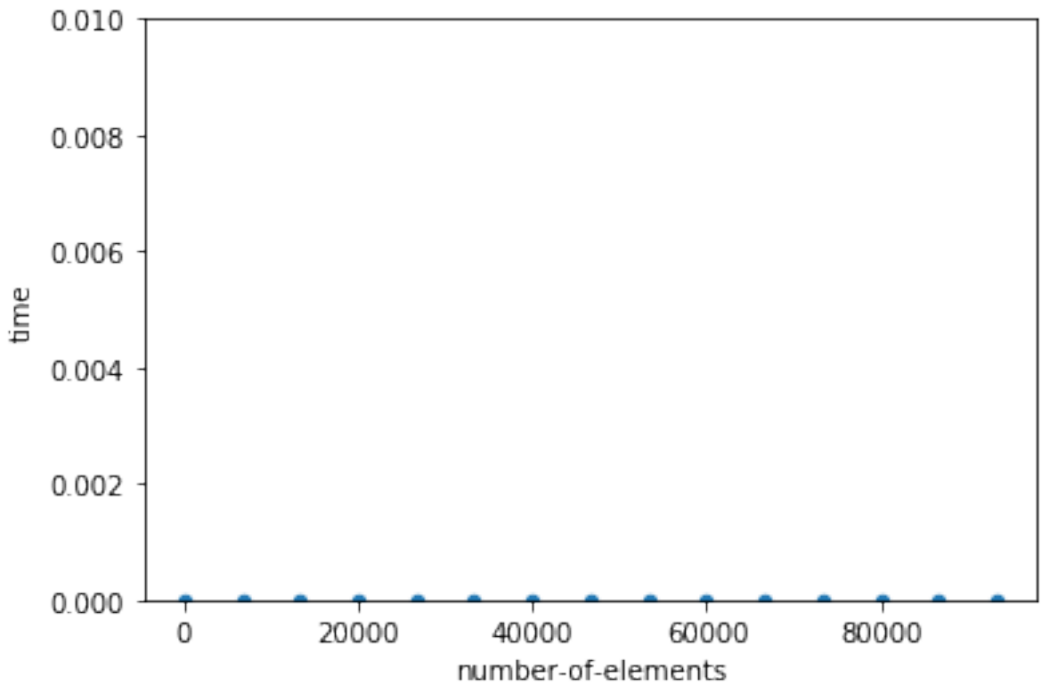
Dictionary

```
In [17]: file = "csvs/getFirstElement_dict.csv"
         dataset7 = pd.read_csv(file,names=names)
         print(dataset7)
```

```
    number-of-elements      time
0                    1  0.000004
1                 6667  0.000004
2                13333  0.000002
3                19999  0.000002
4                26665  0.000002
5                33331  0.000002
6                39997  0.000002
7                46663  0.000002
8                53329  0.000002
9                59995  0.000002
10               66661  0.000002
11               73327  0.000002
12               79993  0.000002
13               86659  0.000002
14               93325  0.000002
```

```
In [18]: ax7 = dataset7.plot(kind='scatter',x='number-of-elements',y='time')
         ax7.set_ylim(0,.01)
```

```
Out[18]: (0, 0.01)
```



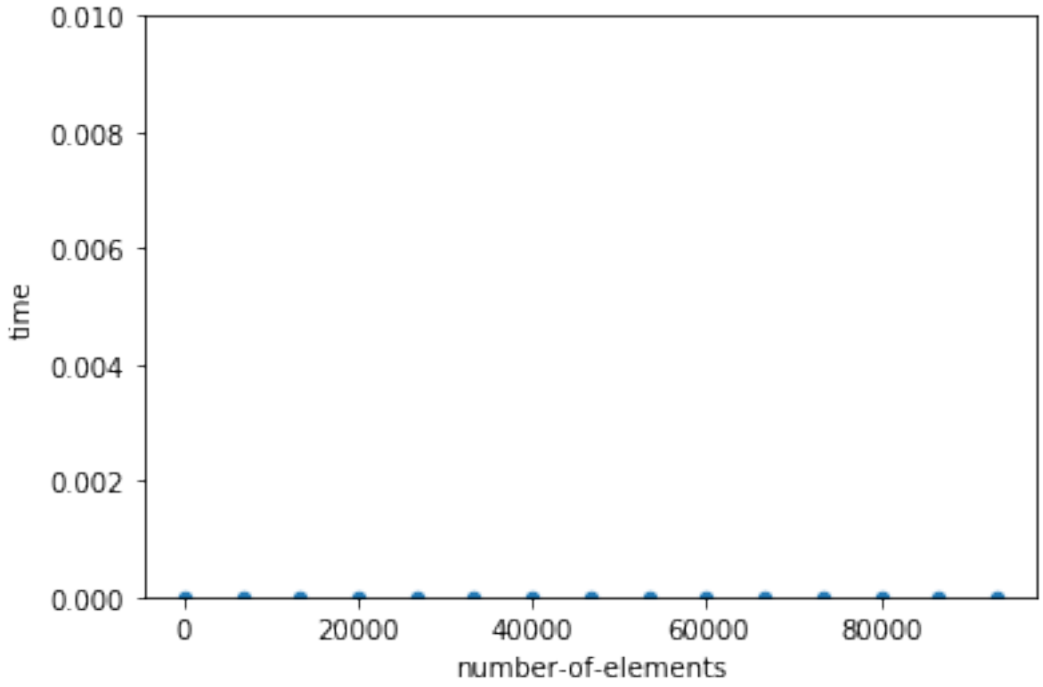
List

```
In [19]: file = "csvs/getFirstElement_list.csv"
         dataset8 = pd.read_csv(file,names=names)
         print(dataset8)
```

```
    number-of-elements          time
0                    2  1.192093e-06
1                 6668  4.768372e-07
2                13334  9.536743e-07
3                20000  9.536743e-07
4                26666  1.192093e-06
5                33332  1.192093e-06
6                39998  2.384186e-06
7                46664  1.668930e-06
8                53330  1.668930e-06
9                59996  1.907349e-06
10               66662  1.907349e-06
11               73328  1.907349e-06
12               79994  1.430511e-06
13               86660  1.907349e-06
14               93326  2.861023e-06
```

```
In [20]: ax8 = dataset8.plot(kind='scatter',x='number-of-elements',y='time')
         ax8.set_ylim(0,.01)
```

Out[20]: (0, 0.01)



Explaination The first element of a data structure is always in the same index, hense a constant operation. This function will be constant for both a list and a dictionary as it is acessing the first lement.

1.2.5 Get the Last Element of the Data Structure

Dictionary

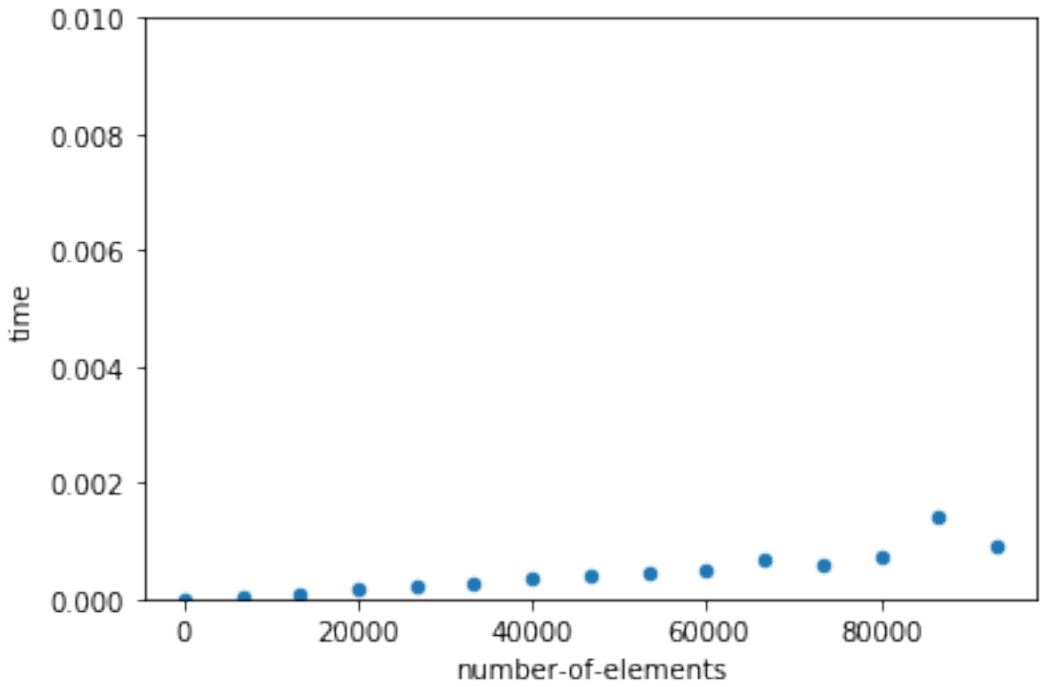
```
In [21]: file = "csvs/getLastElement_dict.csv"
         dataset9 = pd.read_csv(file,names=names)
         print(dataset9)
```

```
    number-of-elements      time
0                    1  0.000007
1                 6667  0.000051
2                13333  0.000096
3                19999  0.000184
4                26665  0.000236
5                33331  0.000270
6                39997  0.000337
7                46663  0.000389
8                53329  0.000452
9                59995  0.000491
```

```
10                66661  0.000669
11                73327  0.000607
12                79993  0.000744
13                86659  0.001409
14                93325  0.000927
```

```
In [22]: ax9 = dataset9.plot(kind='scatter',x='number-of-elements',y='time')
         ax9.set_ylim(0,.01)
```

```
Out[22]: (0, 0.01)
```



List

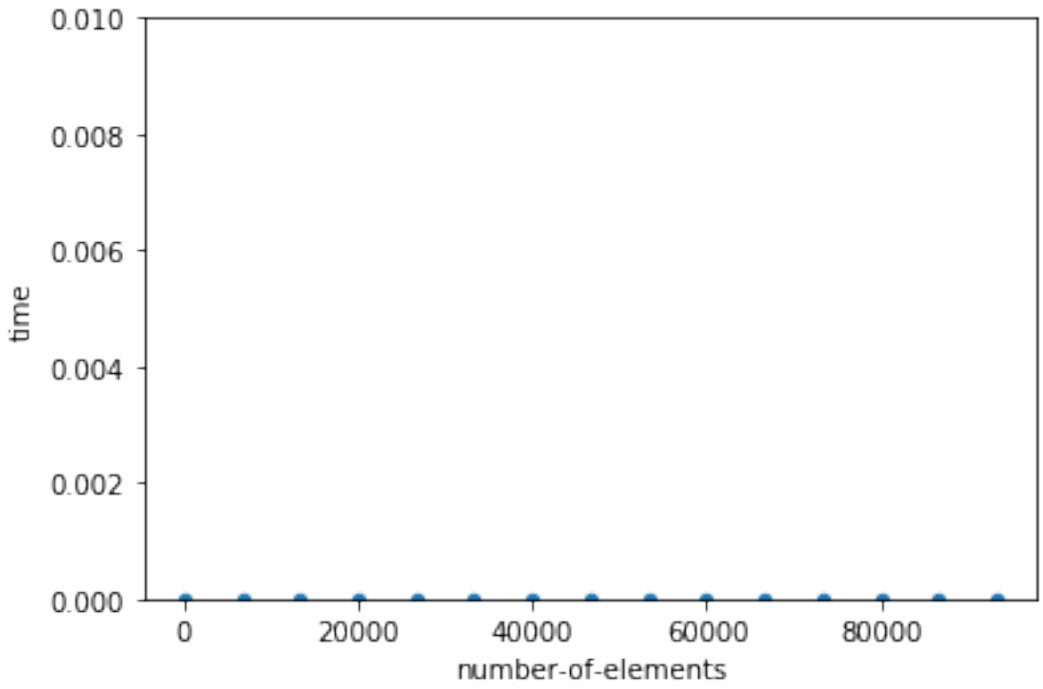
```
In [23]: file = "csvs/getLastElement_list.csv"
         dataset10 = pd.read_csv(file,names=names)
         print(dataset10)
```

```
    number-of-elements          time
0                    2  2.145767e-06
1                 6668  9.536743e-07
2                13334  9.536743e-07
3                20000  1.430511e-06
4                26666  1.430511e-06
5                33332  1.907349e-06
```

```
6                39998  1.907349e-06
7                46664  2.622604e-06
8                53330  2.384186e-06
9                59996  1.907349e-06
10               66662  2.145767e-06
11               73328  1.907349e-06
12               79994  1.430511e-06
13               86660  2.145767e-06
14               93326  2.145767e-06
```

```
In [24]: ax10 = dataset10.plot(kind='scatter',x='number-of-elements',y='time')
         ax10.set_ylim(0,.01)
```

```
Out[24]: (0, 0.01)
```



Explaination Since the list is acessing its same indexed element each time, it will be constant. The dictionary is linear as it must list out the data keys and find the last one.

1.2.6 Get the Length of the Data Structure

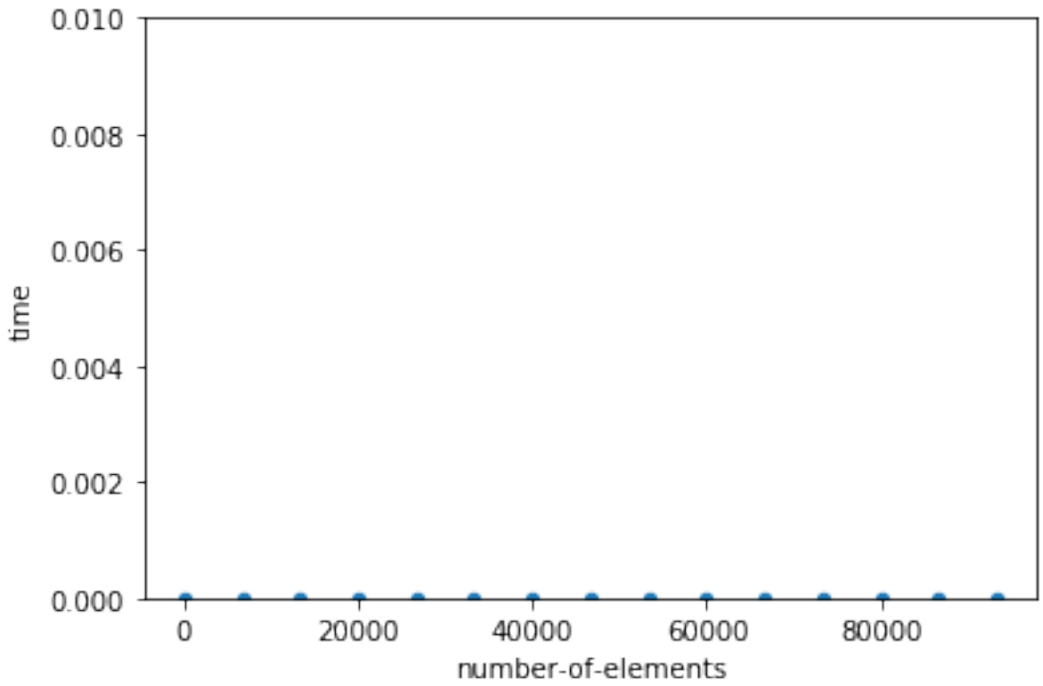
Dictionary

```
In [25]: file = "csvs/lengthOf_dict.csv"
         dataset11 = pd.read_csv(file,names=names)
         print(dataset11)
```

```
    number-of-elements      time
0                    1  0.000004
1                 6667  0.000004
2                13333  0.000003
3                19999  0.000003
4                26665  0.000003
5                33331  0.000003
6                39997  0.000003
7                46663  0.000003
8                53329  0.000003
9                59995  0.000002
10               66661  0.000002
11               73327  0.000003
12               79993  0.000003
13               86659  0.000003
14               93325  0.000003
```

```
In [26]: ax11 = dataset11.plot(kind='scatter',x='number-of-elements',y='time')
         ax11.set_ylim(0,.01)
```

```
Out[26]: (0, 0.01)
```



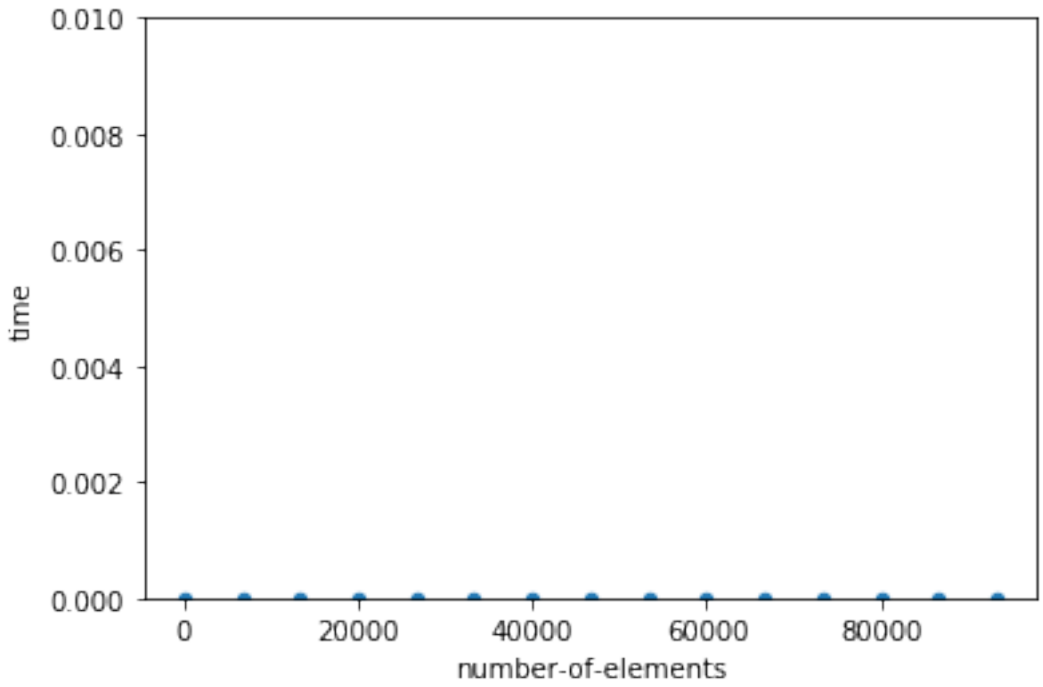
Lists

```
In [27]: file = "csvs/lengthOf_list.csv"
         dataset12 = pd.read_csv(file,names=names)
         print(dataset12)
```

```
    number-of-elements      time
0                    2  0.000002
1                 6668  0.000002
2                13334  0.000001
3                20000  0.000003
4                26666  0.000003
5                33332  0.000002
6                39998  0.000003
7                46664  0.000002
8                53330  0.000002
9                59996  0.000003
10               66662  0.000003
11               73328  0.000003
12               79994  0.000002
13               86660  0.000003
14               93326  0.000003
```

```
In [28]: ax12 = dataset12.plot(kind='scatter',x='number-of-elements',y='time')
         ax12.set_ylim(0,.01)
```

```
Out[28]: (0, 0.01)
```



Explaination Getting the length of a list and dictionary is a constant operation, the length could either besaved as meta data or easily calculated.

1.2.7 Sort the Data Structure

Dictionary Dictionaries cannot be ordered by key, value elements.

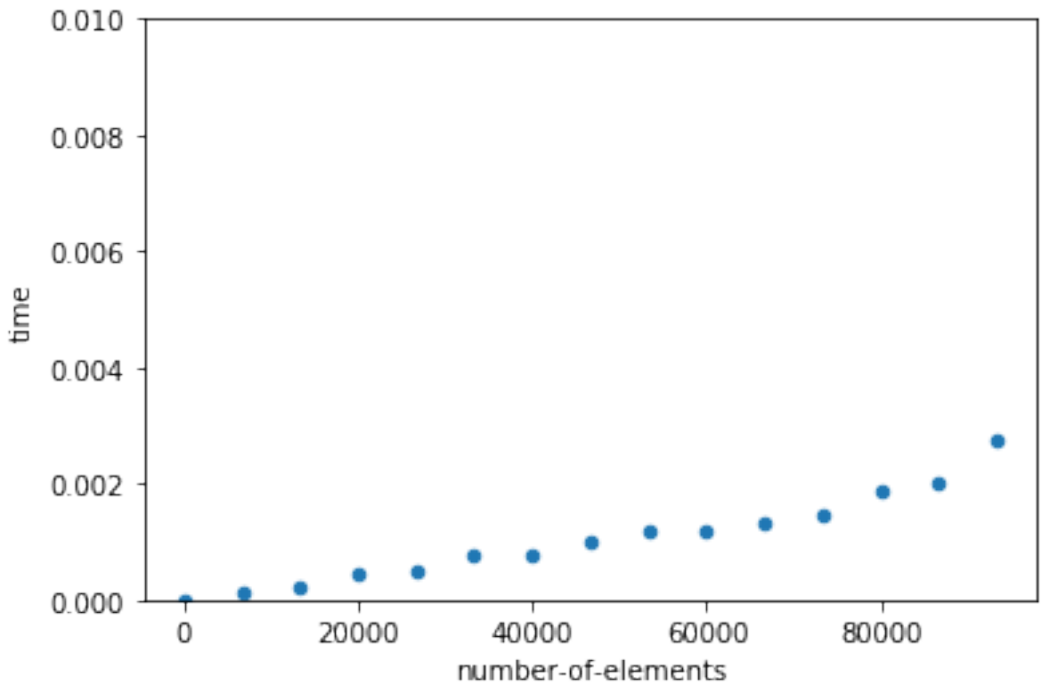
List

```
In [29]: file = "csvs/sort_list.csv"
         dataset14 = pd.read_csv(file,names=names)
         print(dataset14)
```

```
    number-of-elements      time
0                    2  0.000003
1                 6668  0.000142
2                13334  0.000240
3                20000  0.000428
4                26666  0.000489
5                33332  0.000783
6                39998  0.000766
7                46664  0.001014
8                53330  0.001205
9                59996  0.001191
10               66662  0.001315
11               73328  0.001459
12               79994  0.001866
13               86660  0.002030
14               93326  0.002761
```

```
In [30]: ax14 = dataset14.plot(kind='scatter',x='number-of-elements',y='time')
         ax14.set_ylim(0,.01)
```

```
Out[30]: (0, 0.01)
```



Explanation Sorting of a list is linear as it must go through each element individually..

1.2.8 Add to Middle of Data Structure

Dictionary Dictionaries have no "middle" to add to, they will always add to the "end" of the structure.

List

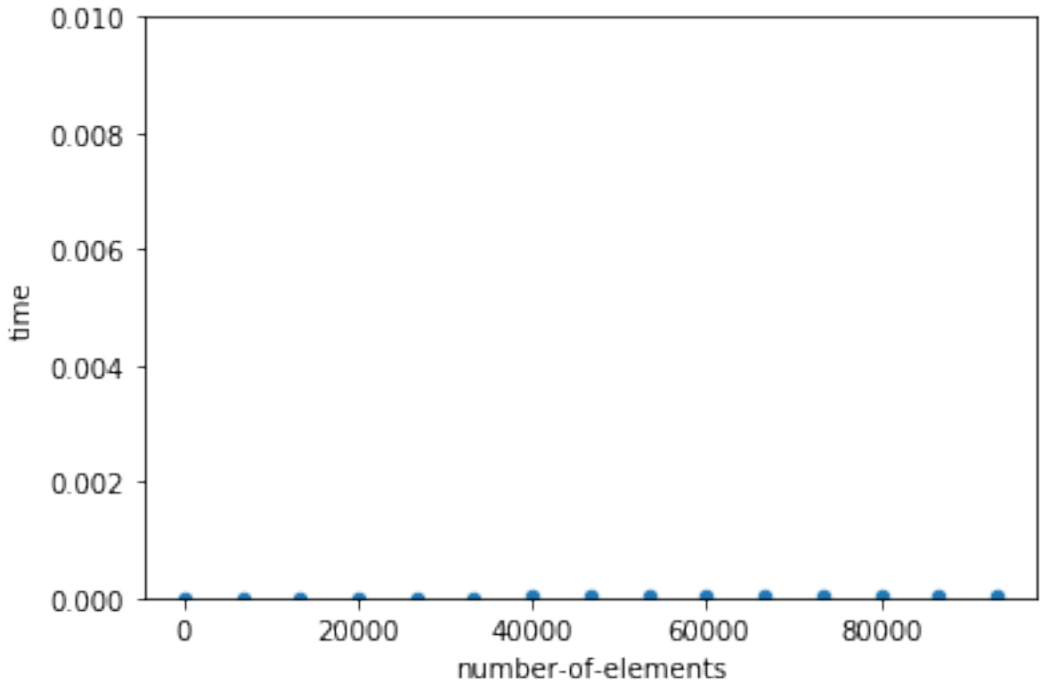
```
In [31]: file = "csvs/addFooMiddle_list.csv"
         dataset15 = pd.read_csv(file,names=names)
         print(dataset15)
```

```
    number-of-elements      time
0                    3  0.000004
1                 6669  0.000003
2                13335  0.000006
3                20001  0.000005
4                26667  0.000008
5                33333  0.000011
6                39999  0.000019
7                46665  0.000017
8                53331  0.000024
9                59997  0.000026
```

```
10                66663  0.000022
11                73329  0.000033
12                79995  0.000031
13                86661  0.000039
14                93327  0.000040
```

```
In [32]: ax15 = dataset15.plot(kind='scatter',x='number-of-elements',y='time')
         ax15.set_ylim(0,.01)
```

```
Out[32]: (0, 0.01)
```



Explanation Adding data to the middle of a list is constant, it seems to go directly to the index and place the data directly.

1.2.9 Remove Middle Element of Data Structure

Dictionary

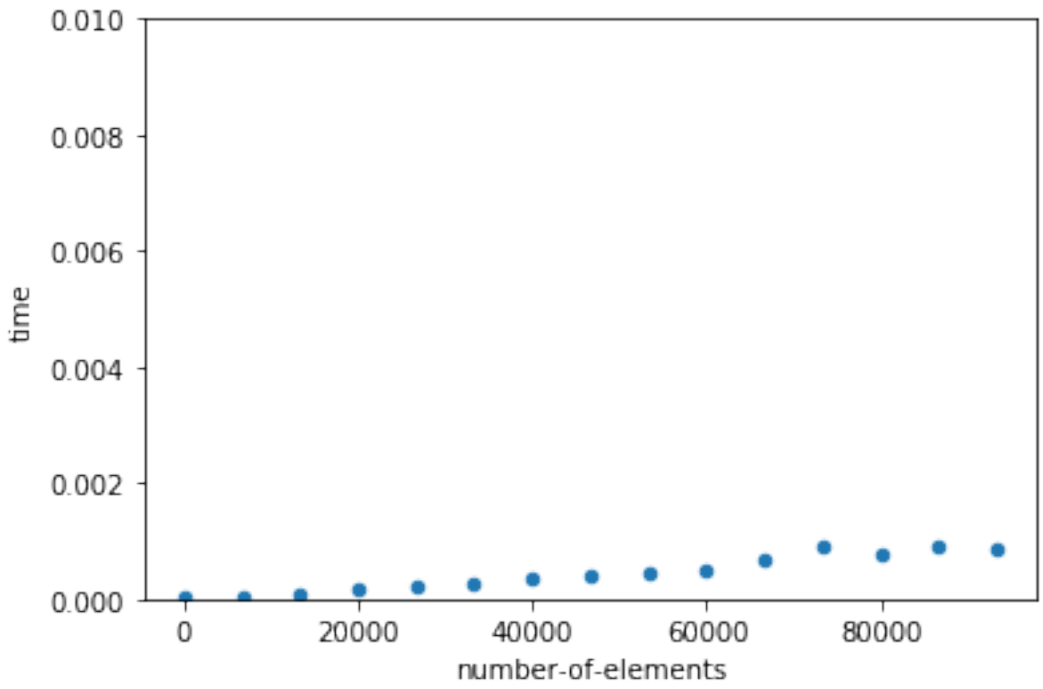
```
In [33]: file = "csvs/removeMiddleElement_dict.csv"
         dataset16 = pd.read_csv(file,names=names)
         print(dataset16)
```

```
    number-of-elements      time
0                    0  0.000022
```

```
1                6666  0.000053
2               13332  0.000100
3               19998  0.000168
4               26664  0.000228
5               33330  0.000268
6               39996  0.000341
7               46662  0.000388
8               53328  0.000443
9               59994  0.000510
10              66660  0.000679
11              73326  0.000933
12              79992  0.000762
13              86658  0.000926
14              93324  0.000875
```

```
In [34]: ax16 = dataset16.plot(kind='scatter',x='number-of-elements',y='time')
         ax16.set_ylim(0,.01)
```

```
Out[34]: (0, 0.01)
```



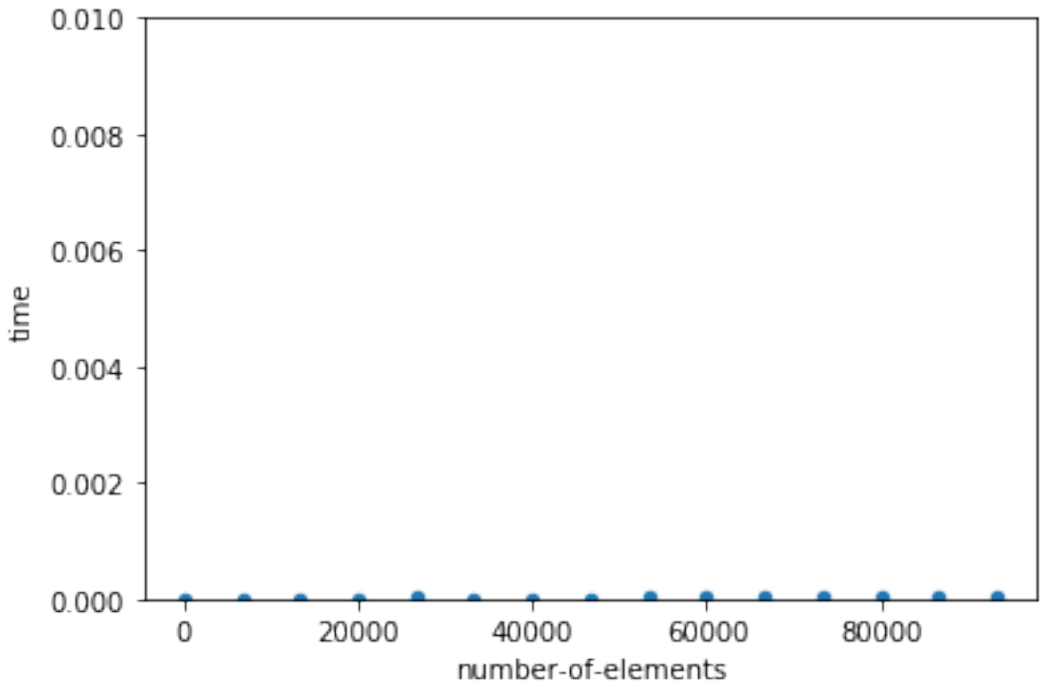
List

```
In [35]: file = "csvs/removeMiddleElement_list.csv"
         dataset17 = pd.read_csv(file,names=names)
         print(dataset17)
```

```
    number-of-elements      time
0                    1  0.000004
1                 6667  0.000005
2                13333  0.000005
3                19999  0.000006
4                26665  0.000015
5                33331  0.000009
6                39997  0.000012
7                46663  0.000013
8                53329  0.000021
9                59995  0.000019
10               66661  0.000020
11               73327  0.000031
12               79993  0.000026
13               86659  0.000020
14               93325  0.000028
```

```
In [36]: ax17 = dataset17.plot(kind='scatter',x='number-of-elements',y='time')
         ax17.set_ylim(0,.01)
```

```
Out[36]: (0, 0.01)
```



1.2.10 Get Middle Element of Data Structure

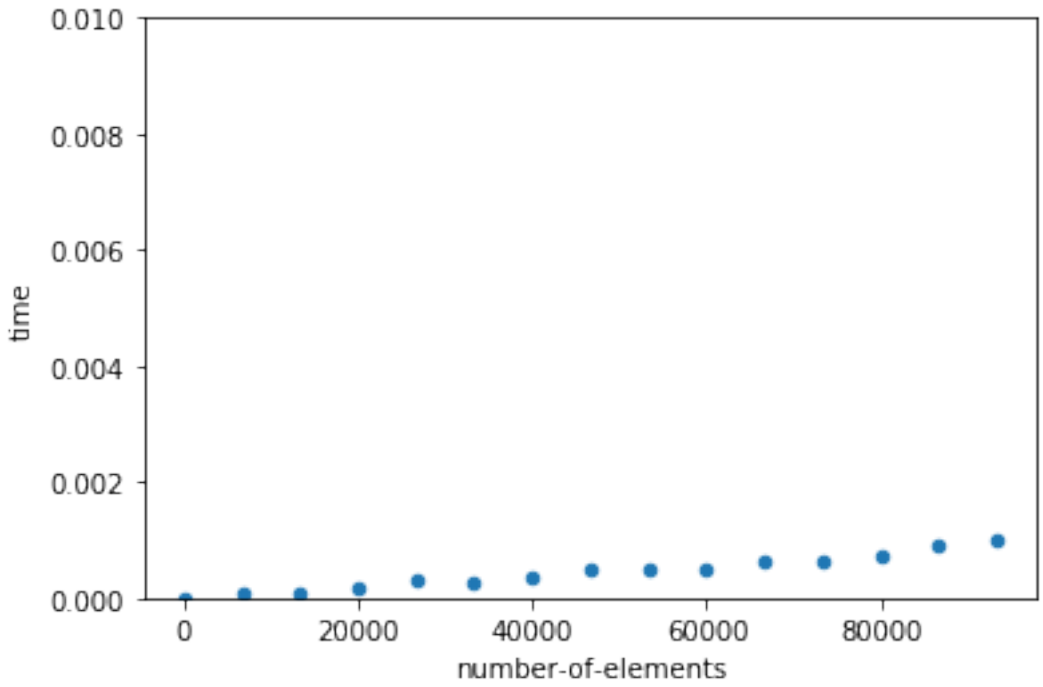
Dictionary

```
In [37]: file = "csvs/getMiddleElement_dict.csv"
         dataset18 = pd.read_csv(file,names=names)
         print(dataset18)
```

```
    number-of-elements      time
0                    1  0.000008
1                 6667  0.000075
2                13333  0.000101
3                19999  0.000191
4                26665  0.000314
5                33331  0.000277
6                39997  0.000341
7                46663  0.000497
8                53329  0.000486
9                59995  0.000495
10               66661  0.000621
11               73327  0.000657
12               79993  0.000734
13               86659  0.000889
14               93325  0.001006
```

```
In [38]: ax18 = dataset18.plot(kind='scatter',x='number-of-elements',y='time')
         ax18.set_ylim(0,.01)
```

```
Out[38]: (0, 0.01)
```



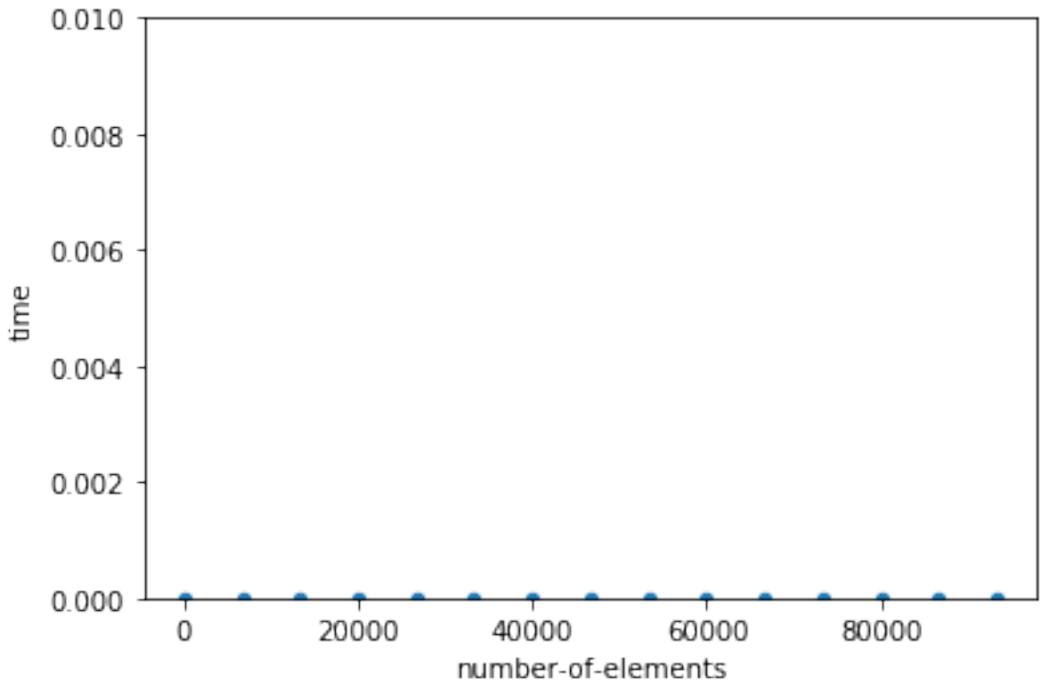
List

```
In [39]: file = "csvs/getMiddleElement_list.csv"
         dataset19 = pd.read_csv(file,names=names)
         print(dataset19)
```

```
    number-of-elements      time
0                    2  0.000003
1                 6668  0.000001
2                13334  0.000001
3                20000  0.000001
4                26666  0.000004
5                33332  0.000003
6                39998  0.000004
7                46664  0.000005
8                53330  0.000004
9                59996  0.000005
10               66662  0.000006
11               73328  0.000005
12               79994  0.000004
13               86660  0.000005
14               93326  0.000005
```

```
In [40]: ax19 = dataset19.plot(kind='scatter',x='number-of-elements',y='time')
         ax19.set_ylim(0,.01)
```

```
Out[40]: (0, 0.01)
```



1.2.11 Summary

From our investigation operations on a list or dictionary are either constant, when independent of the number of elements in a data structure, or linear, when the operation is dependent on the number of elements in the structure.

Based on the results, dictionaries seem to be more performant when searching or updating an element. This is because the elements can be accessed based on their key, relatively instantly, rather than having to iterate through an entire list looking for the particular element.

Lists are useful when the order of the elements in the data structure matter. With dictionaries, the order is immutable.