Lab 2: Runtime Analysis

Your Exercise: Performance of `del` keyword

Devise an experiment that compares the performance of the `del` operator on lists and dictionaries.

Helpful Code

In [ ]:

```
# Make two lists of random 'data'
import random
arry1 = [random.random() for i in range(5000)]
arry2 = [random.random() for i in range(5000)]

# Create a simple dictionary of key:val pairs using these two lists
d1 = dict(zip(arry1,arry2))

# How the deletion of a dictionary key works
#print(d1)
d_keys = list(d1.keys())
del d1[d_keys[0]]
#print(d1)

# Print an empty line for spacing
print()

# How the deletion of a dictionary key works
#print(arry1)
del arry1[0]
#print(arry1)
```

Thanks for providing this useful hint :)

Your Solution

Use a variety of code, Markdown (text) cells below to create your solution. Nice outputs would be timing results, and even plots. You will be graded not only on correctness, but the clarity of your code, descriptive text and other output. Keep it succinct.

GitHub: abhijit-baruah

Start by creating our random lists and dictionaries

In [1]:

```
import random

# Two random lists of size 5000 are created
arry1 = [random.random() for i in range(5000)]
arry2 = [random.random() for i in range(5000)]
print(f'The first two entries in array #1 are: {arry1[0]} and {arry1[1]}')
print(f'The first two entries in array #2 are: {arry2[0]} and {arry2[1]}')

# We create a simple dictionary of key:val pairs using these two lists
d1 = dict(zip(arry1,arry2))
d_keys = list(d1.keys())
print(f'The first two values in our dictionary are : {d1[d_keys[0]]} and {d1[d_keys[1]]}')
```

```
The first two entries in array #1 are: 0.05671829948588647 and 0.10430637865
875936
The first two entries in array #2 are: 0.5952783119509709 and 0.044725617653
81189
The first two values in our dictionary are : 0.5952783119509709 and 0.044725
61765381189
```

Perform `del` operation our list and dictionary and calculate time taken for these processes

We create two functions called `del_list` and `del_dict` that uses `del` operators on list and dictionary respectively.

In [2]:

```
def del_list(array, index):
    del array[index]

def del_dict(dictionary, key):
    del dictionary[key]
```

Next, we need to time the `del` operator on our list and dictionary. We first apply the operator on our dictionary to delete N number of entries from it and calculate the time taken to perform this operation for each of these N entries as well as the average and elapsed time. We then proceed to do the same for `del` operator on a list. We use `perf_counter()` from `time` to get the most accurate results when testing the difference between two times.

In [3]:

```
import time
import numpy as np

N = 100 # number of entries to delete
time1 = [] # this will be a list of time taken to perform individual del operation for dict
net_time1 = [] # this will keep count of elapsed time to perform del operation on dictionar
t = 0

for i in range(0, N):
    t1 = time.perf_counter() # start stopwatch
    del_dict(d1, d_keys[i]) # call the del_dict function to implement del operation on dict
    t2 = time.perf_counter() # stop stopwatch
    time1.append(t2-t1) # updates the list containing the difference in time
    t += (t2-t1) # adds consequtive time differences
    net_time1.append(t) # updates net time taken

#print(time1)
print(f'Average time taken to delete {N} entries from our dictionary is: {np.average(time1)
print(f'Total elapsed time to delete {N} entries from our dictionary is: {net_time1[N-1]} s
```

```
Average time taken to delete 100 entries from our dictionary is: 5.910000001
208005e-07 seconds
Total elapsed time to delete 100 entries from our dictionary is: 5.910000001
2080045e-05 seconds
```

In [4]:

```
# We porform similar execution as above to implement del operator on list

time2 = []
net_time2 = []
t = 0

for i in range(0, N):
    t1 = time.perf_counter()
    del_list(arry1, i)
    t2 = time.perf_counter()
    time2.append(t2-t1)
    t += (t2-t1)
    net_time2.append(t)

#print(time2)
print(f'Time taken to delete {N} entries from our list is: {np.average(time2)} seconds')
print(f'Total elapsed time to delete {N} entries from our list is: {net_time2[N-1]} seconds
```

```
Time taken to delete 100 entries from our list is: 1.5900000002488924e-06 se
conds
Total elapsed time to delete 100 entries from our list is: 0.000159000000024
88923 seconds
```

Testing

For full credit, you must also test your solution so that you can prove to the grade your solution works.

We can check whether our `del` operator was implemented correctly as follows. Printed at the start of our solution are the first two entries from our original list/array and dictionary. After performing `del` operation on the first N entries, we can print below the first two entries from the new list and new dictionary to check the correctness of the operation.

In [5]:

```
new_key = list(d1.keys())
print(f'{[new_key[0]]} is the first key after deletion with value {d1[new_key[0]]}')
print(f'Is {new_key[0]} equal to the {N}-th key in original dictionary? {new_key[0] == d_ke
```

```
[0.8268752819406704] is the first key after deletion with value 0.4432956200
648427
Is 0.8268752819406704 equal to the 100-th key in original dictionary? True
```

So, our implemention of the `del` operator on dictionaries works as expected. We can similarly check the correctness of the `del` operator on any list with a simple boolean statement as follows:

In [6]:

```
s = [1,2,3,4,5]
del_list(s, 0)
print(f'For a list {[1, 2, 3, 4 ,5]} applying the del_list function on first index gives us
```

```
For a list [1, 2, 3, 4, 5] applying the del_list function on first index giv
es us [2, 3, 4, 5]
```

Now that we have checked the correctness of our `del` operator on both lists and dictionaries, we are required to compare the performance of the operator on the same. We do so by:

1. Compare the average time to impplement `del` operator between list and dictionary
2. Making plots of elapsed times and individual times over N

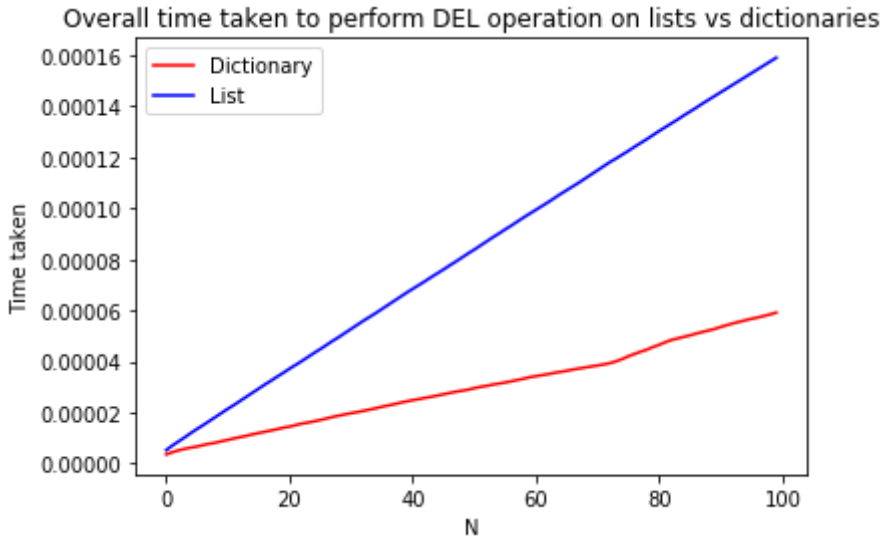
In [7]:

```
print(f'Does del operation on dictionary take same time as del operation on list?: {np.aver
print(f'Is del operation on dictionary faster than del operation on list?: {np.average(time
print(f'Del operator on dictionary is {np.average(time1)/np.average(time2)*100}% faster tha
```

```
Does del operation on dictionary take same time as del operation on list?: F
alse
Is del operation on dictionary faster than del operation on list?: True
Del operator on dictionary is 37.16981132253381% faster than on list by aver
age
```

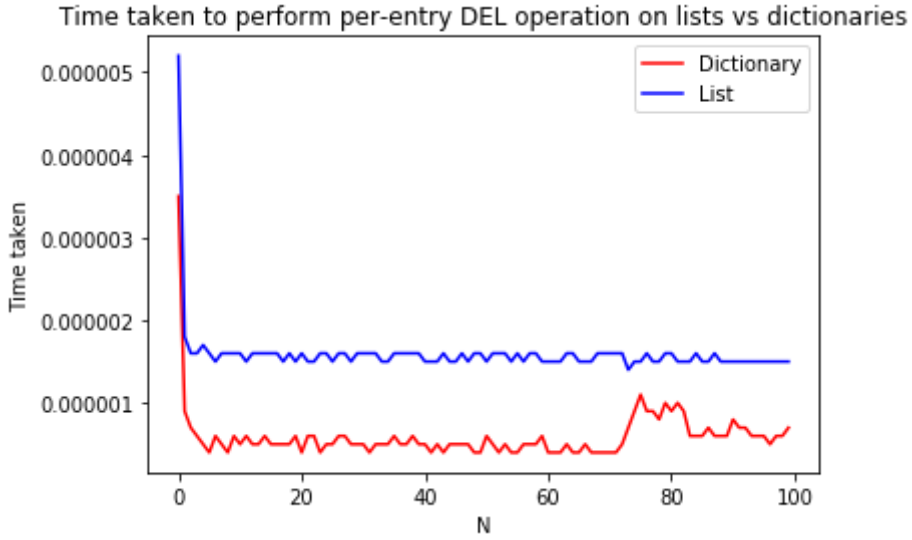
In [9]:

```
import matplotlib.pyplot as plt
x = range(0, N)
plt.plot(x, net_time1, 'r', label = 'Dictionary')
plt.plot(x, net_time2, 'b', label = 'List')
plt.legend()
plt.ylabel("Time taken")
plt.xlabel("N")
plt.title("Overall time taken to perform DEL operation on lists vs dictionaries")
plt.show()
```



In [10]:

```
import matplotlib.pyplot as plt
x = range(0, N)
plt.plot(x, time1, 'r', label = 'Dictionary')
plt.plot(x, time2, 'b', label = 'List')
plt.legend()
plt.ylabel("Time taken")
plt.xlabel("N")
plt.title("Time taken to perform per-entry DEL operation on lists vs dictionaries")
plt.show()
```



Conclusion:

`del` operator on dictionary is faster than that on lists.

In [ ]: