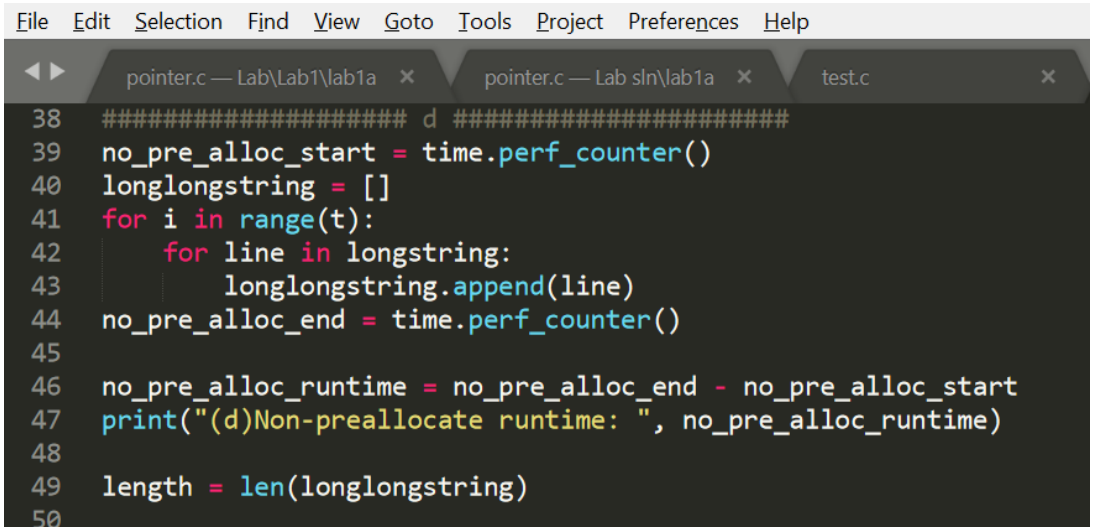
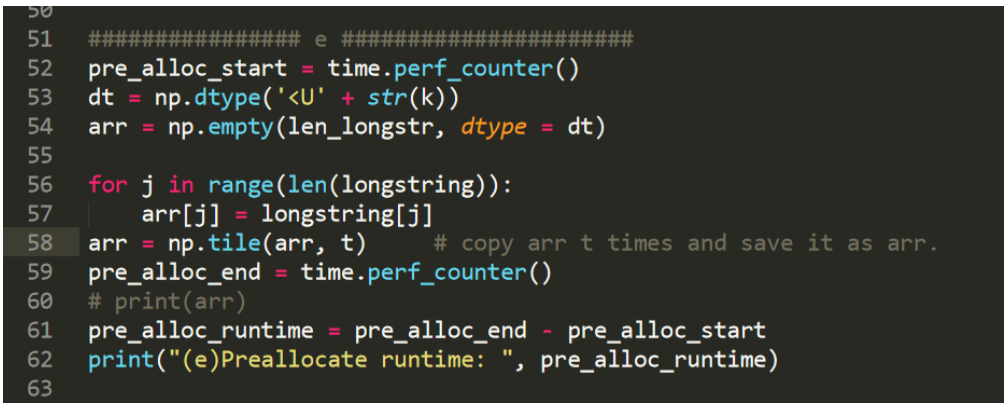
(d)



This is a nested for loop, which will take O(t*n) times. Also, without preallocation of the size of list, when we read a new string and want to put the value inside the list. This will force the memory to reallocate for every iteration, meaning that we need another O(n) time to allocate the memory. And the total run time will be bounded by $O(n^2)$

(e)



In this case, since we have already know the length of the list. We can use numpy.empty to create an empty array, meaning that we have already assigned a memory space for the value to store. Therefore, whenever we have a new value, we only need O(1) to allocate the value into the array. When I form the array for the first time, I used tile() function to expand the array 100 times. This will create a exactly array of the previous one, except the data type it store is a numpy array. The total run time of this implementation will be O(n) since I only have to run through the list for a first time.

(f)

```
(d)Non-preallocate runtime:  0.008652793000000103
(e)Preallocate runtime:  0.0005193949999999781
[Finished in 0.7s]
```

Clearly, we can see that the runtime is faster when I preallocate it .