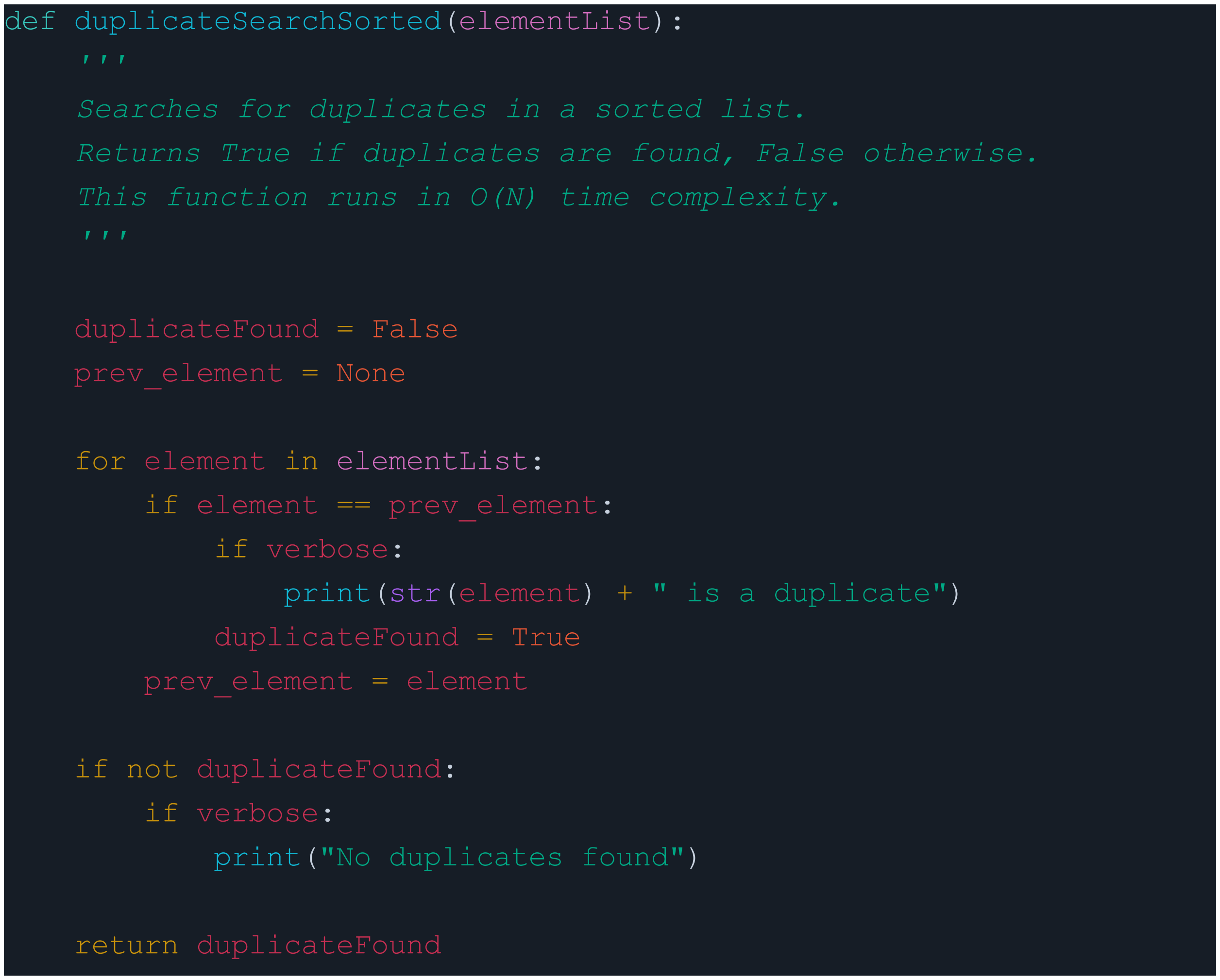


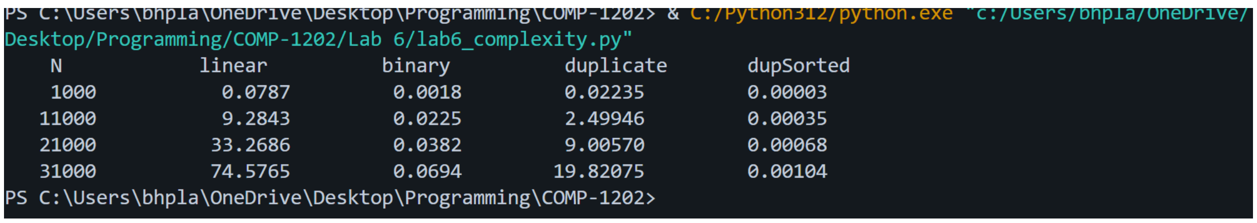
```python
def duplicateSearchSorted(elementList):
    '''
    Searches for duplicates in a sorted list.
    Returns True if duplicates are found, False otherwise.
    This function runs in O(N) time complexity.
    '''

    duplicateFound = False
    prev_element = None

    for element in elementList:
        if element == prev_element:
            if verbose:
                print(str(element) + " is a duplicate")
            duplicateFound = True
        prev_element = element

    if not duplicateFound:
        if verbose:
            print("No duplicates found")

    return duplicateFound
```



```
PS C:\Users\bhpla\OneDrive\Desktop\Programming\COMP-1202> & C:/Python312/python.exe "c:/Users/bhpla/OneDrive/
Desktop/Programming/COMP-1202/Lab 6/lab6_complexity.py"
     N            linear          binary          duplicate        dupSorted
   1000            0.0787          0.0018          0.02235         0.00003
  11000            9.2843          0.0225          2.49946         0.00035
  21000           33.2686          0.0382          9.00570         0.00068
  31000           74.5765          0.0694         19.82075         0.00104
PS C:\Users\bhpla\OneDrive\Desktop\Programming\COMP-1202>
```

Section 2: O(N^2)

- Each search iterates through the entire list making it just O(N) but the numElements searches make the overall complexity O(N^2)

Section 3: O(log N)

- This time its binary numElements searches making it O(log N)

Section 4: O(N^2)

- This has a duplicate search in the unsorted list, this uses nested loops to compare every pair of elements in the list.

Section 5: O(N)

- This again performs a duplicate search but in the sorted list instead, the function goes through the list once comparing each element with the last. Because the list is sorted it makes it O(N).