Nama : Krisna Alif Meilana

NIM : L200180028

Modul 6

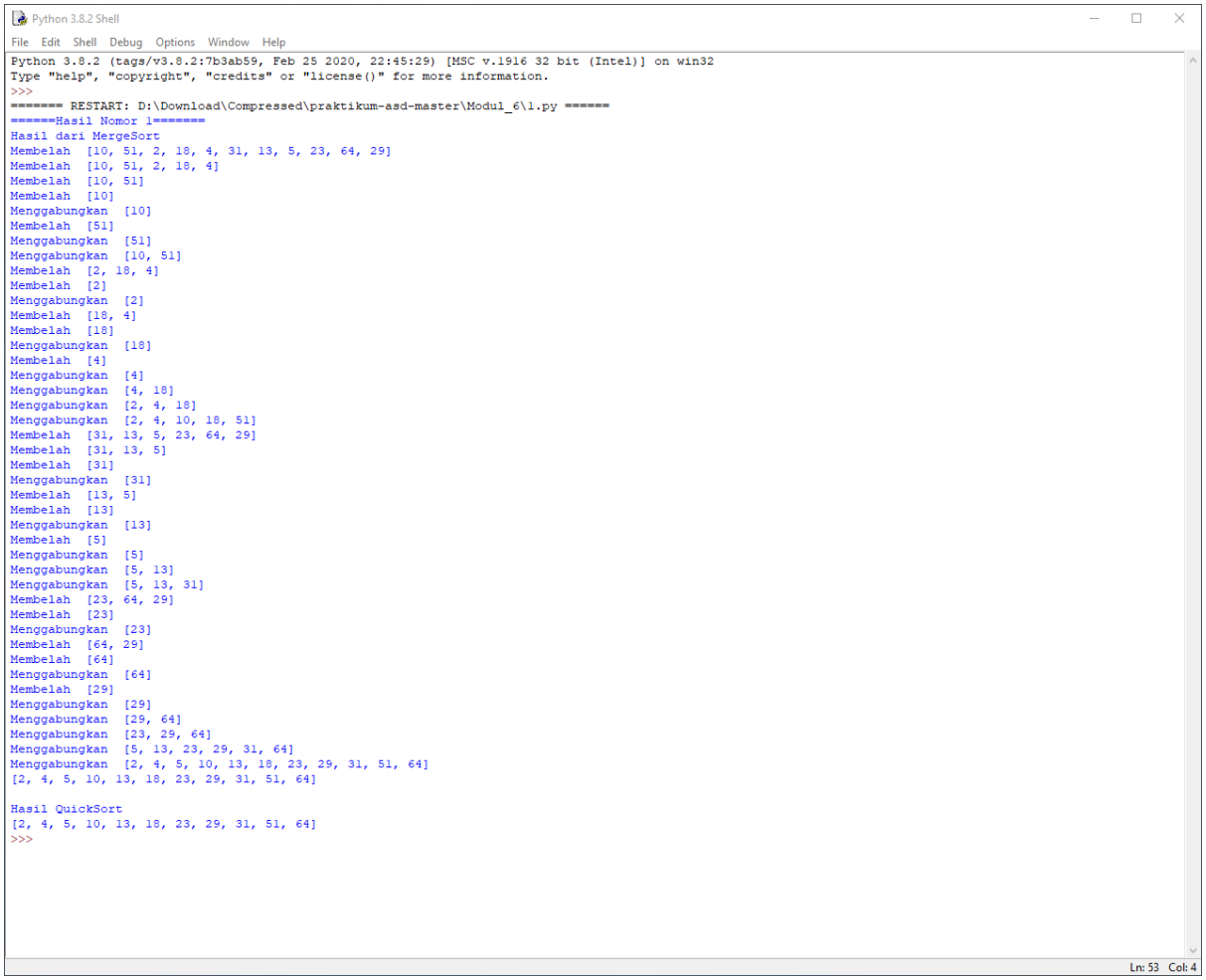
1.

1.py - D:\Download\Compressed\praktikum-asd-master\Modul_6\1.py (3.8.2)

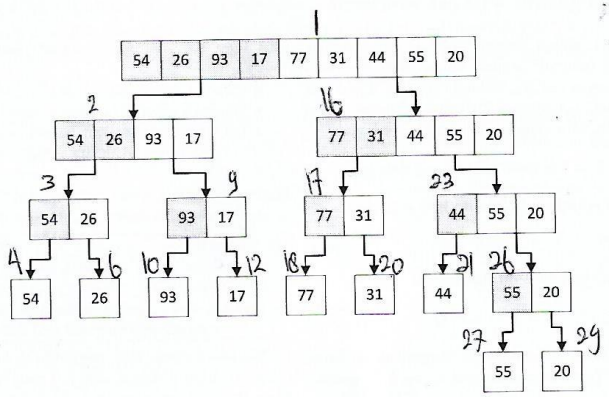
File Edit Format Run Options Window Help

```
print("======Hasil Nomor 1=======")
class MhsTIF(object):
    """Class MhsTIF yang dibangun dari class Mahasiswa"""

    def __init__(self,nama,NIM,kota,us):
        """Metode inisialisasi ini menutupimetode inisiasi di class Manusia."""
        self.nama=nama
        self.NIM=NIM
        self.kotaTinggal=kota
        self.uangsaku=us


c0 = MhsTIF('Budi', 10, 'Sukoharjo', 240000)
c1 = MhsTIF('Krisna', 51, 'Sragen', 230000)
c2 = MhsTIF('Chad', 2, 'Surakarta', 250000)
c3 = MhsTIF('Rocky', 18, 'Yogyakarta', 235000)
c4 = MhsTIF('Andre', 4, 'Boyolali', 240000)
c5 = MhsTIF('Fendi', 31, 'Salatiga', 250000)
c6 = MhsTIF('Doni', 13, 'Klaten', 245000)
c7 = MhsTIF('Jaluh', 5, 'Wonogiri', 245000)
c8 = MhsTIF('Ranto', 23, 'Ngawi', 245000)
c9 = MhsTIF('Hassan', 64, 'Karanganyar', 270000)
c10 = MhsTIF('Kamaru', 29, 'Purwadadi', 265000)


Daftar = [c0.NIM,c1.NIM,c2.NIM,c3.NIM,c4.NIM,c5.NIM,c6.NIM,c7.NIM,c8.NIM,c9.NIM,c10.NIM]

def mergeSort(nlist):
    print("Membelah ",nlist)
    if len(nlist)>1:
        mid = len(nlist)//2
        lefthalf = nlist[:mid]
        righthalf = nlist[mid:]

        mergeSort(lefthalf)
        mergeSort(righthalf)
        i=j=k=0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                nlist[k]=lefthalf[i]
                i=i+1
            else:
                nlist[k]=righthalf[j]
                j=j+1
            k=k+1

        while i < len(lefthalf):
            nlist[k]=lefthalf[i]
            i=i+1
            k=k+1

        while j < len(righthalf):
            nlist[k]=righthalf[j]
            j=j+1
            k=k+1
    print("Menggabungkan ",nlist)
nlist = Daftar
print("Hasil dari MergeSort")
mergeSort(nlist)
print(nlist)
```

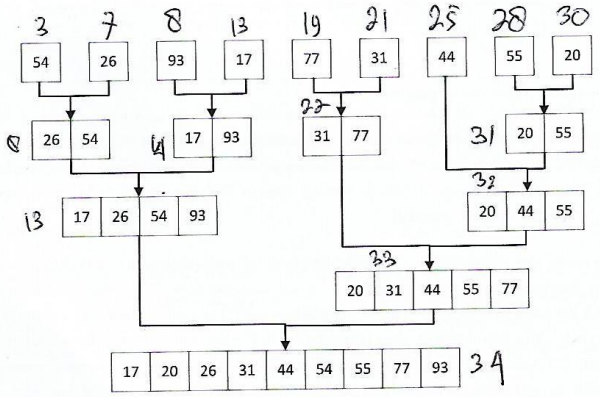
Ln: 20 Col: 43


Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: D:\Download\Compressed\praktikum-asd-master\Modul_6\1.py ======
======Hasil Nomor 1=======
Hasil dari MergeSort
Membelah [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
Membelah [10, 51, 2, 18, 4]
Membelah [10, 51]
Membelah [10]
Menggabungkan [10]
Membelah [51]
Menggabungkan [51]
Menggabungkan [10, 51]
Membelah [2, 18, 4]
Membelah [2]
Menggabungkan [2]
Membelah [18, 4]
Membelah [18]
Menggabungkan [18]
Membelah [4]
Menggabungkan [4]
Menggabungkan [4, 18]
Menggabungkan [2, 4, 18]
Menggabungkan [2, 4, 10, 18, 51]
Membelah [31, 13, 5, 23, 64, 29]
Membelah [31, 13, 5]
Membelah [31]
Menggabungkan [31]
Membelah [13, 5]
Membelah [13]
Menggabungkan [13]
Membelah [5]
Menggabungkan [5]
Menggabungkan [5, 13]
Menggabungkan [5, 13, 31]
Membelah [23, 64, 29]
Membelah [23]
Menggabungkan [23]
Membelah [64, 29]
Membelah [64]
Menggabungkan [64]
Membelah [29]
Menggabungkan [29]
Menggabungkan [29, 64]
Menggabungkan [23, 29, 64]
Menggabungkan [5, 13, 23, 29, 31, 64]
Menggabungkan [2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
Hasil QuickSort
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
>>>
Ln: 53 Col: 4

2.

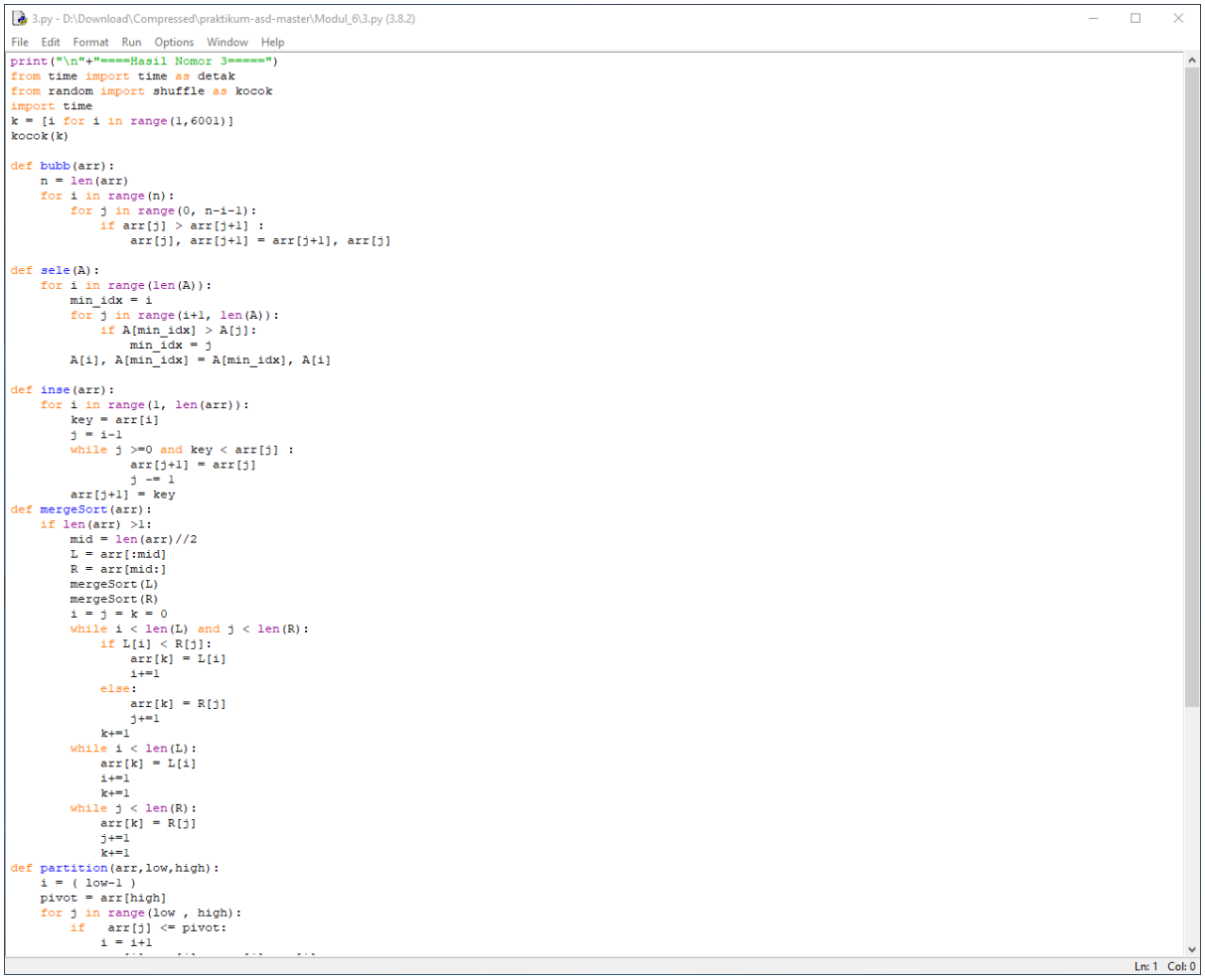


Gambar 6.1: Membelah list sampai tiap sub-list berisi satu elemen atau kosong. Sesudah itu digabung seperti ditunjukkan di Gambar 6.2.



Gambar 6.2: Menggabungkan list satu demi satu.

3.


3.py - D:\Download\Compressed\praktikum-asd-master\Modul_6\3.py (3.8.2)
File Edit Format Run Options Window Help
print("\n"+"====Hasil Nomor 3=====")
from time import time as detak
from random import shuffle as kocok
import time
k = [i for i in range(1,6001)]
kocok(k)
def bubb(arr):
n = len(arr)
for i in range(n):
for j in range(0, n-i-1):
if arr[j] > arr[j+1] :
arr[j], arr[j+1] = arr[j+1], arr[j]
def sele(A):
for i in range(len(A)):
min_idx = i
for j in range(i+1, len(A)):
if A[min_idx] > A[j]:
min_idx = j
A[i], A[min_idx] = A[min_idx], A[i]
def inse(arr):
for i in range(1, len(arr)):
key = arr[i]
j = i-1
while j >=0 and key < arr[j] :
arr[j+1] = arr[j]
j -= 1
arr[j+1] = key
def mergeSort(arr):
if len(arr) >1:
mid = len(arr)//2
L = arr[:mid]
R = arr[mid:]
mergeSort(L)
mergeSort(R)
i = j = k = 0
while i < len(L) and j < len(R):
if L[i] < R[j]:
arr[k] = L[i]
i+=1
else:
arr[k] = R[j]
j+=1
k+=1
while i < len(L):
arr[k] = L[i]
i+=1
k+=1
while j < len(R):
arr[k] = R[j]
j+=1
k+=1
def partition(arr,low,high):
i = (low-1)
pivot = arr[high]
for j in range(low , high):
if arr[j] <= pivot:
i = i+1
Ln: 1 Col: 0

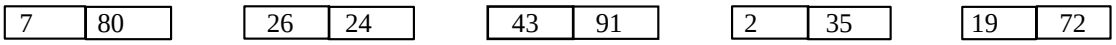

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: D:\Download\Compressed\praktikum-asd-master\Modul_6\3.py ======
====Hasil Nomor 3=====
bubble : 5.83465 detik
selection : 2.6095 detik
insertion : 2.65547 detik
merge : 0.040976 detik
quick : 0.0269794 detik
>>>
Ln: 12 Col: 4

4.

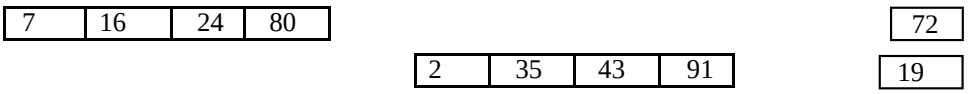
MergeSort

L=[80,7,24,16,43,91,35,2,19,72]

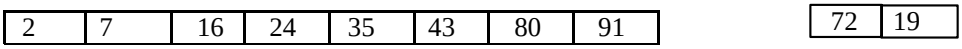
Proses 1



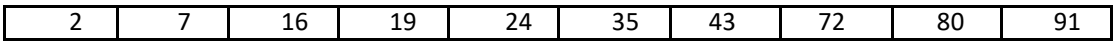
Proses2



Proses3



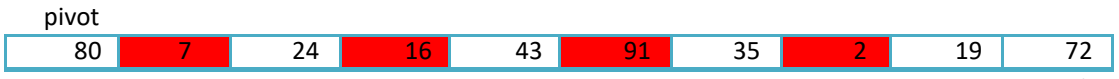
Proses4

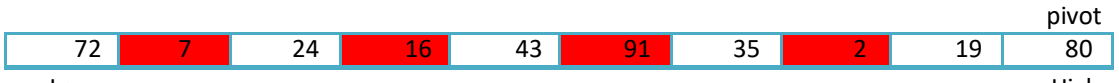


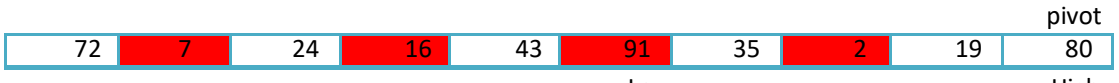
No.4 (B)

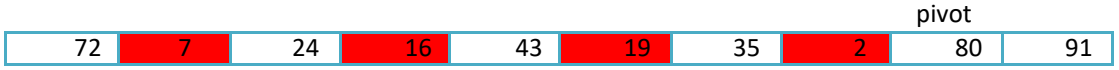
QuickSort

L=[80,7,24,16,43,91,35,2,19,72]

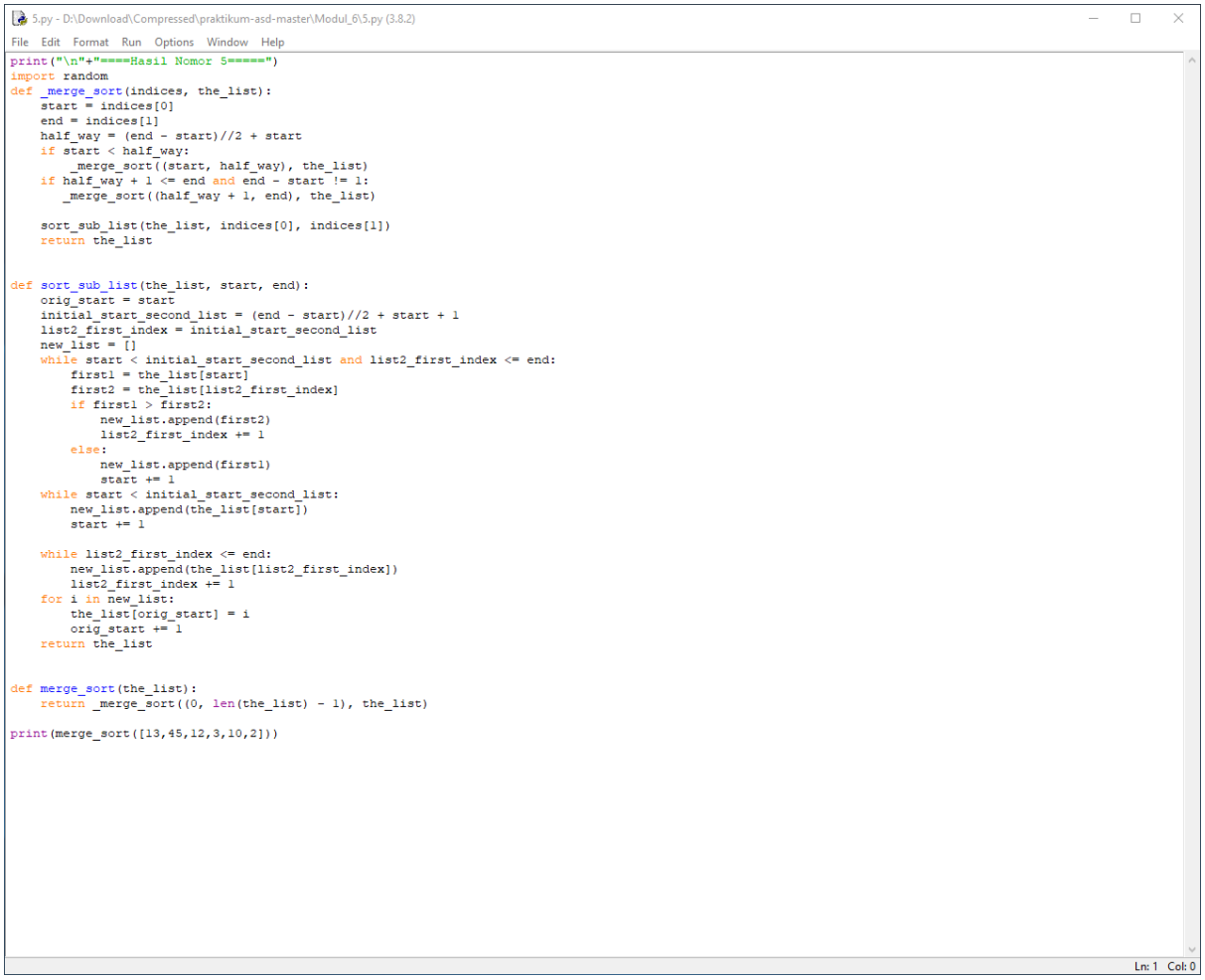








5.


5.py - D:\Download\Compressed\praktikum-asd-master\Modul_6\5.py (3.8.2)
File Edit Format Run Options Window Help
print("\n"+"====Hasil Nomor 5=====")
import random
def _merge_sort(indices, the_list):
start = indices[0]
end = indices[1]
half_way = (end - start)//2 + start
if start < half_way:
_merge_sort((start, half_way), the_list)
if half_way + 1 <= end and end - start != 1:
_merge_sort((half_way + 1, end), the_list)
sort_sub_list(the_list, indices[0], indices[1])
return the_list
def sort_sub_list(the_list, start, end):
orig_start = start
initial_start_second_list = (end - start)//2 + start + 1
list2_first_index = initial_start_second_list
new_list = []
while start < initial_start_second_list and list2_first_index <= end:
first1 = the_list[start]
first2 = the_list[list2_first_index]
if first1 > first2:
new_list.append(first2)
list2_first_index += 1
else:
new_list.append(first1)
start += 1
while start < initial_start_second_list:
new_list.append(the_list[start])
start += 1
while list2_first_index <= end:
new_list.append(the_list[list2_first_index])
list2_first_index += 1
for i in new_list:
the_list[orig_start] = i
orig_start += 1
return the_list
def merge_sort(the_list):
return _merge_sort((0, len(the_list) - 1), the_list)
print(merge_sort([13,45,12,3,10,2]))
Ln: 1 Col: 0


Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: D:\Download\Compressed\praktikum-asd-master\Modul_6\5.py ======
====Hasil Nomor 5=====
[2, 3, 10, 12, 13, 45]
>>>
Ln: 8 Col: 4

6.

6.py - D:\Download\Compressed\praktikum-asd-master\Modul_6\6.py (3.8.2)

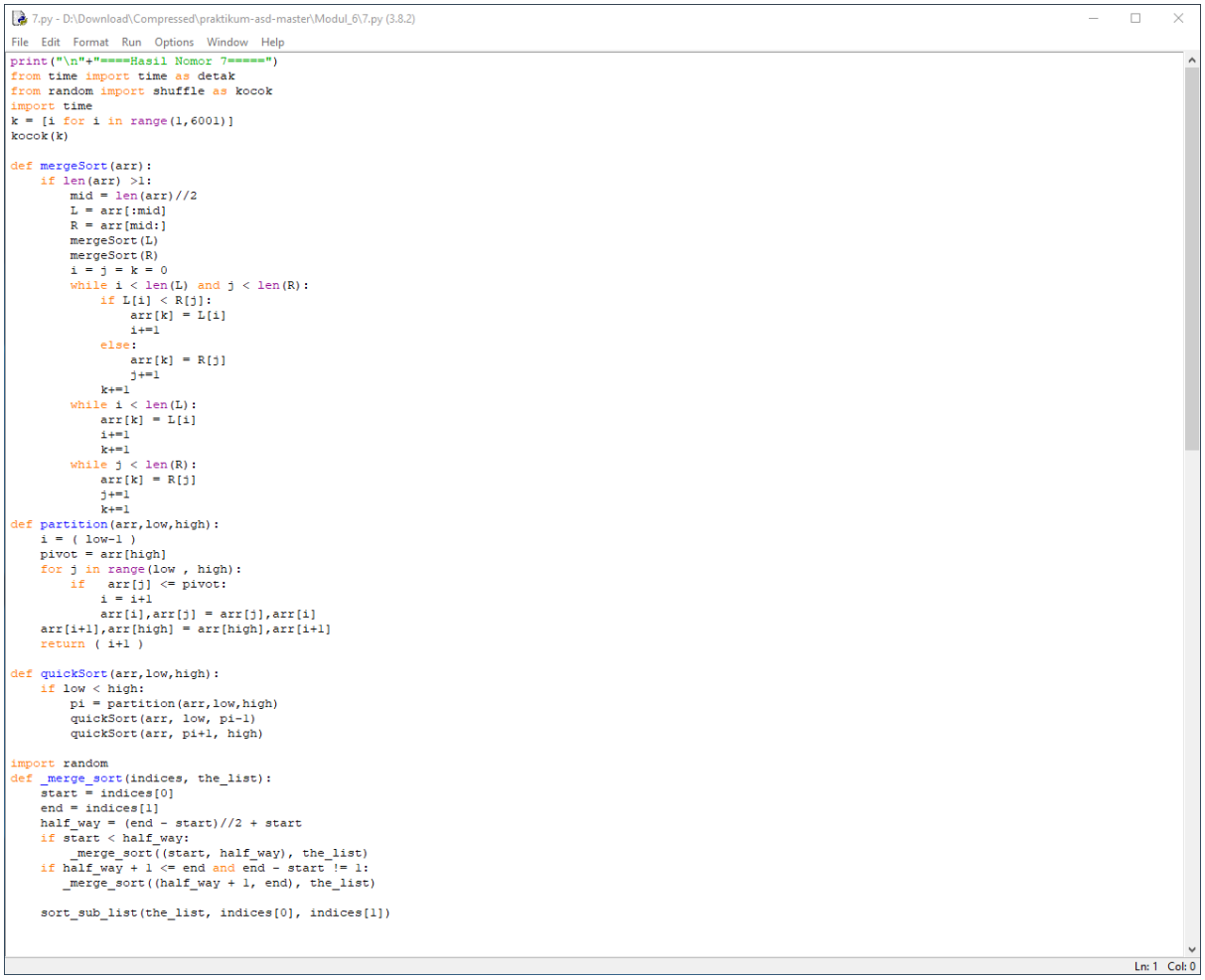
File Edit Format Run Options Window Help

```
print("\n"+"====Hasil Nomor 6=====")
def quickSort(L, ascending = True):
    quicksorthelp(L, 0, len(L), ascending)

def quicksorthelp(L, low, high, ascending = True):
    result = 0
    if low < high:
        pivot_location, result = Partition(L, low, high, ascending)
        result += quicksorthelp(L, low, pivot_location, ascending)
        result += quicksorthelp(L, pivot_location + 1, high, ascending)
    return result


def Partition(L, low, high, ascending = True):
    result = 0
    pivot, pidx = median_of_three(L, low, high)
    L[low], L[pidx] = L[pidx], L[low]
    i = low + 1
    for j in range(low+1, high, 1):
        result += 1
        if (ascending and L[j] < pivot) or (not ascending and L[j] > pivot):
            L[i], L[j] = L[j], L[i]
            i += 1
    L[low], L[i-1] = L[i-1], L[low]
    return i - 1, result

def median_of_three(L, low, high):
    mid = (low+high-1)//2
    a = L[low]
    b = L[mid]
    c = L[high-1]
    if a <= b <= c:
        return b, mid
    if c <= b <= a:
        return b, mid
    if a <= c <= b:
        return c, high-1
    if b <= c <= a:
        return c, high-1
    return a, low

listel = list([14,4,2,104,23,50])

quickSort(listel, False)  # descending order
print('sorted:')
print(listel)
```

Ln: 1 Col: 0

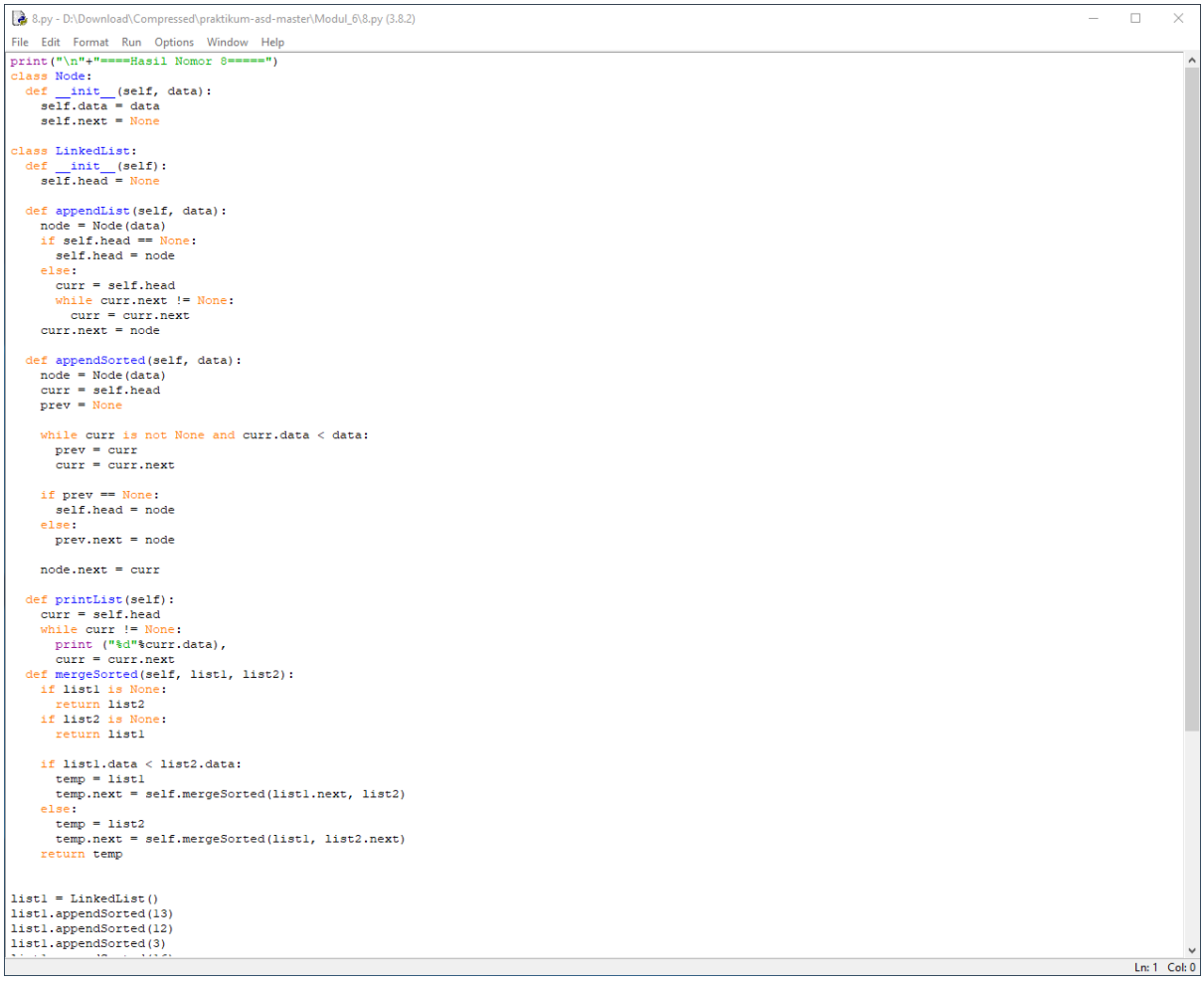

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: D:\Download\Compressed\praktikum-asd-master\Modul_6\6.py ======
====Hasil Nomor 6=====
sorted:
[104, 50, 23, 14, 4, 2]
>>>
Ln: 9 Col: 4

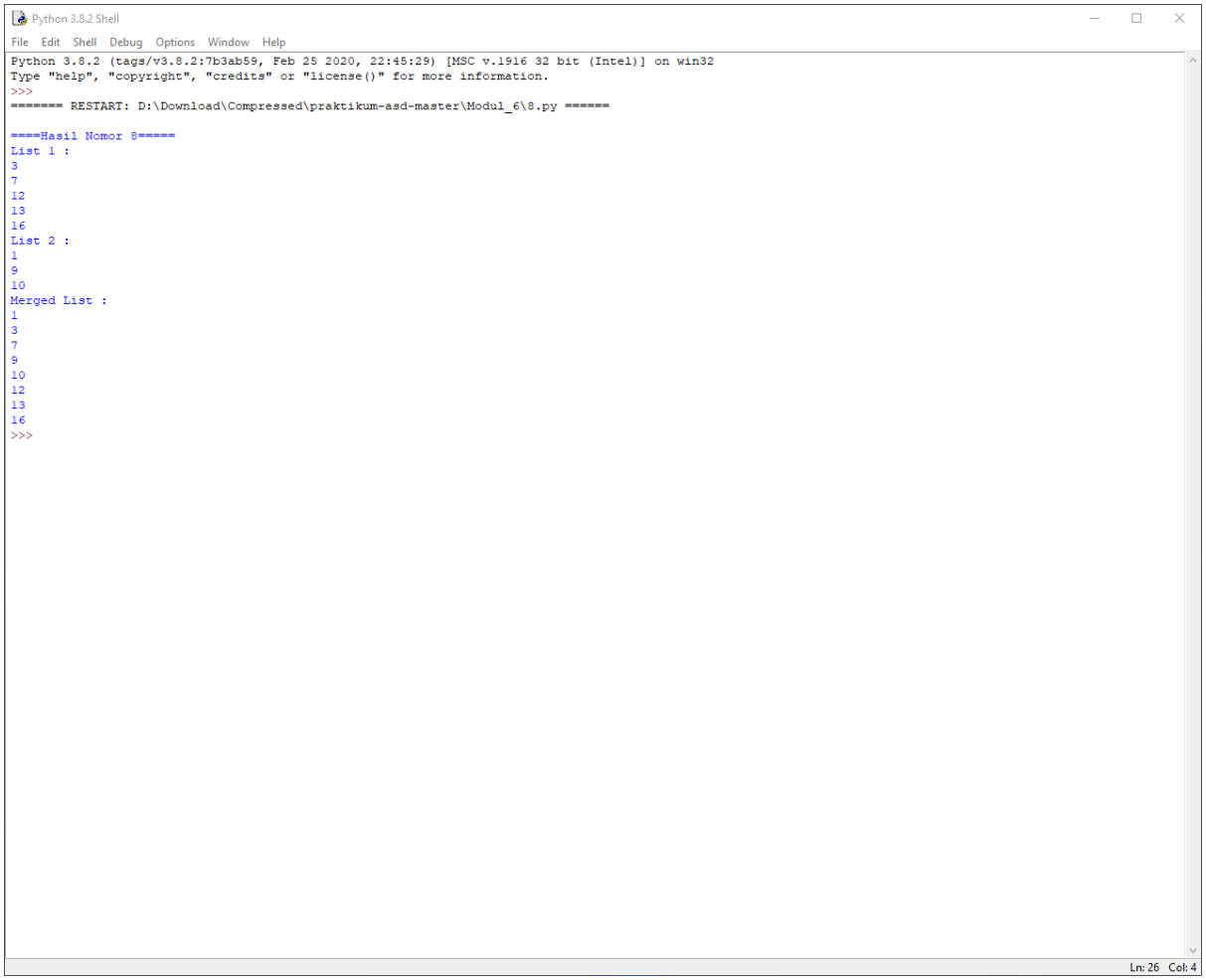
7.


7.py - D:\Download\Compressed\praktikum-asd-master\Modul_6\7.py (3.8.2)
File Edit Format Run Options Window Help
print("\n"+"====Hasil Nomor 7=====")
from time import time as detak
from random import shuffle as kocok
import time
k = [i for i in range(1,6001)]
kocok(k)
def mergeSort(arr):
if len(arr) >1:
mid = len(arr)//2
L = arr[:mid]
R = arr[mid:]
mergeSort(L)
mergeSort(R)
i = j = k = 0
while i < len(L) and j < len(R):
if L[i] < R[j]:
arr[k] = L[i]
i+=1
else:
arr[k] = R[j]
j+=1
k+=1
while i < len(L):
arr[k] = L[i]
i+=1
k+=1
while j < len(R):
arr[k] = R[j]
j+=1
k+=1
def partition(arr,low,high):
i = (low-1)
pivot = arr[high]
for j in range(low , high):
if arr[j] <= pivot:
i = i+1
arr[i],arr[j] = arr[j],arr[i]
arr[i+1],arr[high] = arr[high],arr[i+1]
return (i+1)
def quickSort(arr,low,high):
if low < high:
pi = partition(arr,low,high)
quickSort(arr, low, pi-1)
quickSort(arr, pi+1, high)
import random
def _merge_sort(indices, the_list):
start = indices[0]
end = indices[1]
half_way = (end - start)//2 + start
if start < half_way:
_merge_sort((start, half_way), the_list)
if half_way + 1 <= end and end - start != 1:
_merge_sort((half_way + 1, end), the_list)
sort_sub_list(the_list, indices[0], indices[1])
Ln: 1 Col: 0


Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: D:\Download\Compressed\praktikum-asd-master\Modul_6\7.py ======
====Hasil Nomor 7=====
merge : 0.0429671 detik
quick : 0.0269887 detik
merge mod : -0.00199866 detik
quick mod : -0.0509512 detik
>>>
Ln: 11 Col: 4

8.


8.py - D:\Download\Compressed\praktikum-asd-master\Modul_6\8.py (3.8.2)
File Edit Format Run Options Window Help
print("\n"+"====Hasil Nomor 8=====")
class Node:
def __init__(self, data):
self.data = data
self.next = None
class LinkedList:
def __init__(self):
self.head = None
def appendList(self, data):
node = Node(data)
if self.head == None:
self.head = node
else:
curr = self.head
while curr.next != None:
curr = curr.next
curr.next = node
def appendSorted(self, data):
node = Node(data)
curr = self.head
prev = None
while curr is not None and curr.data < data:
prev = curr
curr = curr.next
if prev == None:
self.head = node
else:
prev.next = node
node.next = curr
def printList(self):
curr = self.head
while curr != None:
print ("%d"%curr.data),
curr = curr.next
def mergeSorted(self, list1, list2):
if list1 is None:
return list2
if list2 is None:
return list1
if list1.data < list2.data:
temp = list1
temp.next = self.mergeSorted(list1.next, list2)
else:
temp = list2
temp.next = self.mergeSorted(list1, list2.next)
return temp
list1 = LinkedList()
list1.appendSorted(13)
list1.appendSorted(12)
list1.appendSorted(3)
Ln: 1 Col: 0


Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: D:\Download\Compressed\praktikum-asd-master\Modul_6\8.py ======
====Hasil Nomor 8=====
List 1 :
3
7
12
13
16
List 2 :
1
9
10
Merged List :
1
3
7
9
10
12
13
16
>>>
Ln: 26 Col: 4