Nama : Aisyah Goevara

NIM : L200180034

Kelas : B

Modul 6

Pengurutan Lanjutan

1. no 1.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 1.py (3.8.2)

File Edit Format Run Options Window Help

```
class Mahasiswa:
    keadaan = 'lapar'
    def __init__(self, nama, nim, kota, us):
        self.nama = nama
        self.nim = nim
        self.kotaTinggal = kota
        self.uangSaku = us
    def __str__(self):
        s = self.nama + ', NIM ' + str(self.nim) \
            + '. Tinggal di ' + self.kotaTinggal \
            + '. Uang saku Rp ' + str(self.uangSaku) \
            + ' perharinya.'
        return s
    def ambilNama(self):
        return self.nama
    def ambilNIM(self):
        return self.nim
    def ambilUangSaku(self):
        return self.uangSaku

def mergeSort(A):
    if len(A) > 1:
        mid = len(A) // 2
        separuhkiri = A[:mid]
        separuhkanan = A[mid:]
        mergeSort(separuhkiri)
        mergeSort(separuhkanan)
        i = 0; j = 0; k = 0
        while i < len(separuhkiri) and j < len(separuhkanan):
            if separuhkiri[i] < separuhkanan[j]:
                A[k] = separuhkiri[i]
                i += 1
            else:
                A[k] = separuhkanan[j]
                j += 1
            k += 1

        while i < len(separuhkiri):
            A[k] = separuhkiri[i]
            i += 1
            k += 1
        while j < len(separuhkanan):
```

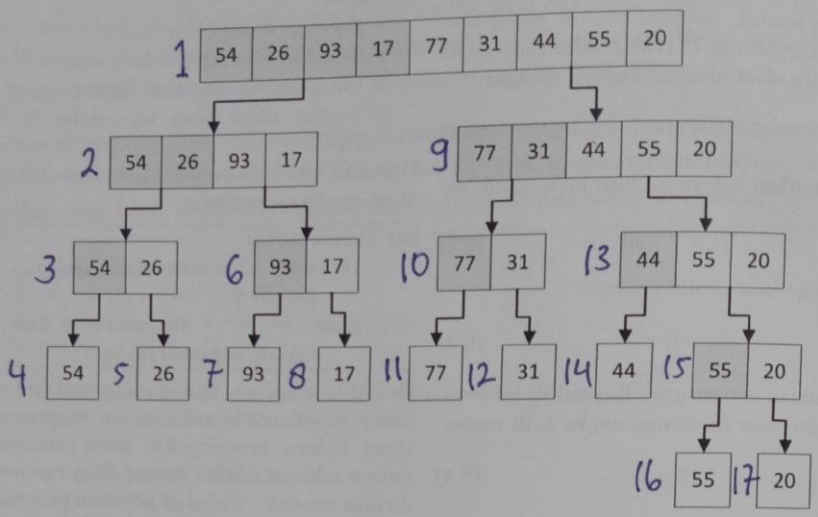
Ln: 82 Col: 39

8:37
04/04/2020

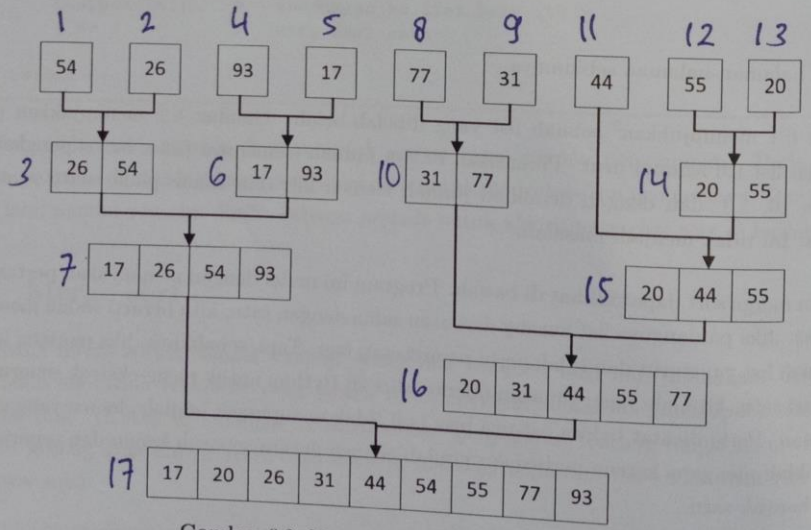

no 1.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 1.py (3.8.2)
File Edit Format Run Options Window Help
i += 1
k += 1
while j < len(separuhkanan):
A[k] = separuhkanan[j]
j += 1
k += 1
def quickSort(A):
quickSortBantu(A, 0, len(A) - 1)
def quickSortBantu(A, awal, akhir):
if awal < akhir:
titikBelah = partisi(A, awal, akhir)
quickSortBantu(A, awal, titikBelah - 1)
quickSortBantu(A, titikBelah + 1, akhir)
def partisi(A, awal, akhir):
nilaiPivot = A[awal]
penandaKiri = awal + 1
penandaKanan = akhir
selesai = False
while not selesai:
while penandaKiri <= penandaKanan and A[penandaKiri] <= nilaiPivot:
penandaKiri = penandaKiri + 1
while A[penandaKanan] >= nilaiPivot and penandaKanan >= penandaKiri:
penandaKanan -= 1
if penandaKanan < penandaKiri:
selesai = True
else:
temp = A[penandaKiri]
A[penandaKiri] = A[penandaKanan]
A[penandaKanan] = temp
temp = A[awal]
A[awal] = A[penandaKiri]
A[penandaKanan] = temp
return penandaKanan
mahas1 = Mahasiswa("Aisyah", 34, "Surakarta", 255000)
mahas2 = Mahasiswa("Goe", 36, "Yogyakarta", 650000)
mahas3 = Mahasiswa("Vara", 40, "Semarang", 450000)
mahas4 = Mahasiswa("Septia", 46, "Salatiga", 200000)
mahas5 = Mahasiswa("Waku", 50, "Purbalingga", 350000)
Ln: 82 Col: 39
8:38
04/04/2020


no 1.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 1.py (3.8.2)
File Edit Format Run Options Window Help
def quickSort(A):
quickSortBantu(A, 0, len(A) - 1)
def quickSortBantu(A, awal, akhir):
if awal < akhir:
titikBelah = partisi(A, awal, akhir)
quickSortBantu(A, awal, titikBelah - 1)
quickSortBantu(A, titikBelah + 1, akhir)
def partisi(A, awal, akhir):
nilaiPivot = A[awal]
penandaKiri = awal + 1
penandaKanan = akhir
selesai = False
while not selesai:
while penandaKiri <= penandaKanan and A[penandaKiri] <= nilaiPivot:
penandaKiri = penandaKiri + 1
while A[penandaKanan] >= nilaiPivot and penandaKanan >= penandaKiri:
penandaKanan -= 1
if penandaKanan < penandaKiri:
selesai = True
else:
temp = A[penandaKiri]
A[penandaKiri] = A[penandaKanan]
A[penandaKanan] = temp
temp = A[awal]
A[awal] = A[penandaKiri]
A[penandaKanan] = temp
return penandaKanan
mahas1 = Mahasiswa("Aisyah", 34, "Surakarta", 255000)
mahas2 = Mahasiswa("Goe", 36, "Yogyakarta", 650000)
mahas3 = Mahasiswa("Vara", 40, "Semarang", 450000)
mahas4 = Mahasiswa("Septia", 46, "Salatiga", 200000)
mahas5 = Mahasiswa("Waku", 50, "Purbalingga", 350000)
mahas6 = Mahasiswa("Fiita", 54, "Bekasi", 470000)
listin = [mahas1.nim, mahas2.nim, mahas3.nim, mahas4.nim, mahas5.nim, mahas6.nim]
mergeSort(listin)
print(listin)
Ln: 82 Col: 39
8:38
04/04/2020


Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 1.py
[34, 36, 40, 46, 50, 54]
>>>



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.

2.

3.

no 3.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 3.py (3.8.2)

File Edit Format Run Options Window Help

```
from time import time as detak
from random import shuffle as kocok

def swap(A, p, q):
    temp = A[p]
    A[p] = A[q]
    A[q] = temp
def cariposisiterkecil(A, darisini, sampaisini):
    posisiterkecil = darisini
    for i in range(darisini + 1, sampaisini):
        if A[i] < A[posisiterkecil]:
            posisiterkecil = i
    return posisiterkecil

def bubbleSort(A):
    n = len(A)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if A[j] > A[j + 1]:
                swap(A, j, j + 1)

def selectionSort(A):
    n = len(A)
    for i in range(n - 1):
        indexkecil = cariposisiterkecil(A, i, n)
        if indexkecil != i:
            swap(A, i, indexkecil)

def insertionSort(A):
    n = len(A)
    for i in range(1, n):
        nilai = A[i]
        pos = i
        while pos > 0 and nilai < A[pos - 1]:
            A[pos] = A[pos - 1]
            pos = pos - 1
        A[pos] = nilai

def mergeSort(A):
    if len(A) > 1:
        mid = len(A) // 2
        L = A[:mid]
```

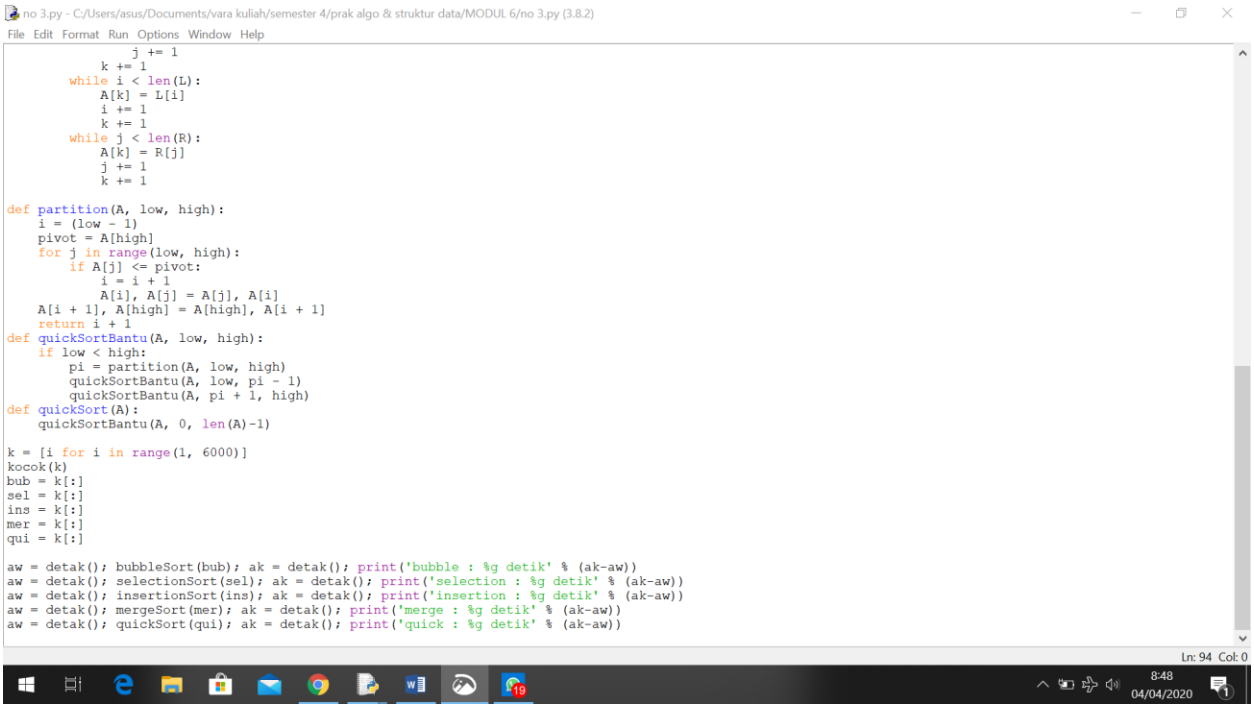
Ln: 94 Col: 0

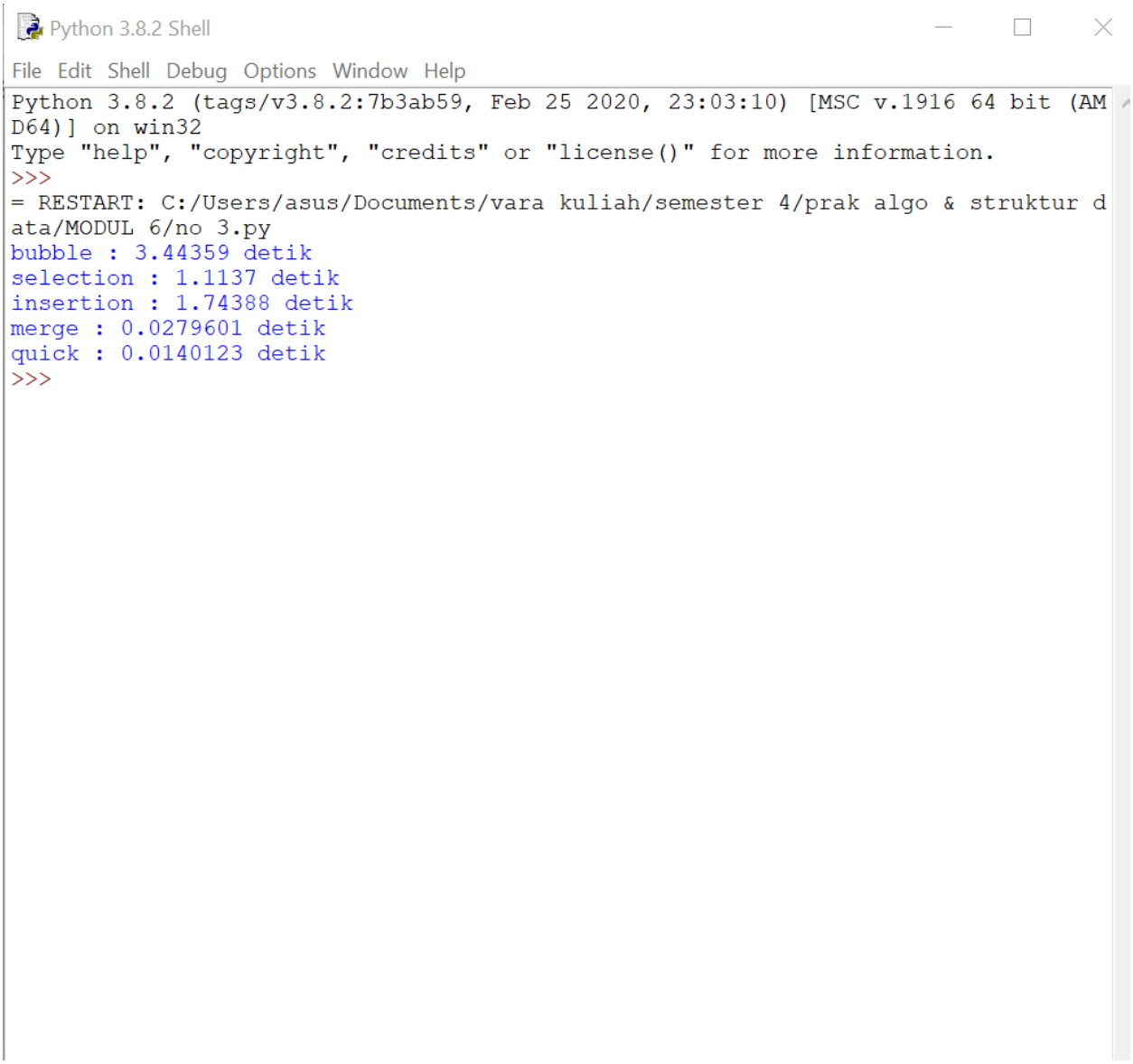
no 3.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 3.py (3.8.2)

File Edit Format Run Options Window Help

```
def mergeSort(A):
    if len(A) > 1:
        mid = len(A) // 2
        L = A[:mid]
        R = A[mid:]
        mergeSort(L)
        mergeSort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                A[k] = L[i]
                i += 1
            else:
                A[k] = R[j]
                j += 1
            k += 1
        while i < len(L):
            A[k] = L[i]
            i += 1
            k += 1
        while j < len(R):
            A[k] = R[j]
            j += 1
            k += 1

def partition(A, low, high):
    i = (low - 1)
    pivot = A[high]
    for j in range(low, high):
        if A[j] <= pivot:
            i = i + 1
            A[i], A[j] = A[j], A[i]
    A[i + 1], A[high] = A[high], A[i + 1]
    return i + 1
def quickSortBantu(A, low, high):
    if low < high:
        pi = partition(A, low, high)
        quickSortBantu(A, low, pi - 1)
        quickSortBantu(A, pi + 1, high)
def quickSort(A):
    quickSortBantu(A, 0, len(A)-1)
```

Ln: 94 Col: 0


no 3.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 3.py (3.8.2)
File Edit Format Run Options Window Help
j += 1
k += 1
while i < len(L):
A[k] = L[i]
i += 1
k += 1
while j < len(R):
A[k] = R[j]
j += 1
k += 1
def partition(A, low, high):
i = (low - 1)
pivot = A[high]
for j in range(low, high):
if A[j] <= pivot:
i = i + 1
A[i], A[j] = A[j], A[i]
A[i + 1], A[high] = A[high], A[i + 1]
return i + 1
def quickSortBantu(A, low, high):
if low < high:
pi = partition(A, low, high)
quickSortBantu(A, low, pi - 1)
quickSortBantu(A, pi + 1, high)
def quickSort(A):
quickSortBantu(A, 0, len(A)-1)
k = [i for i in range(1, 6000)]
kocok(k)
bub = k[:]
sel = k[:]
ins = k[:]
mer = k[:]
qui = k[:]
aw = detak(); bubbleSort(bub); ak = detak(); print('bubble : %g detik' % (ak-aw))
aw = detak(); selectionSort(sel); ak = detak(); print('selection : %g detik' % (ak-aw))
aw = detak(); insertionSort(ins); ak = detak(); print('insertion : %g detik' % (ak-aw))
aw = detak(); mergeSort(mer); ak = detak(); print('merge : %g detik' % (ak-aw))
aw = detak(); quickSort(qui); ak = detak(); print('quick : %g detik' % (ak-aw))
Ln: 94 Col: 0
8:48
04/04/2020


Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur d
ata/MODUL 6/no 3.py
bubble : 3.44359 detik
selection : 1.1137 detik
insertion : 1.74388 detik
merge : 0.0279601 detik
quick : 0.0140123 detik
>>>

4.

4(a). Merge Sort

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

80	7	24	16	43	91	35	2	19	72

Proses 1

7	80		26	24		43	91		2	35		19	72

Proses 2

7	16	24	80		2	35	43	91		19	72

Proses 3

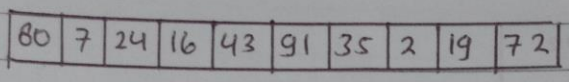
2	7	16	24	35	43	80	91		19	72

Proses 4

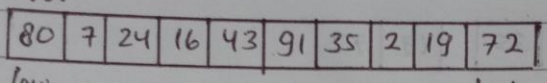
2	7	16	19	24	35	43	72	80	91

4(b). Quick Sort

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

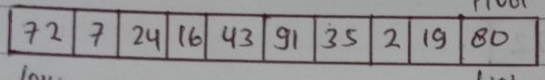


Pivot



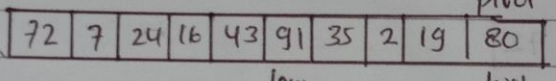
low high

Pivot



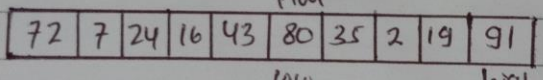
low high

pivot



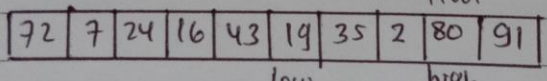
low high

Pivot



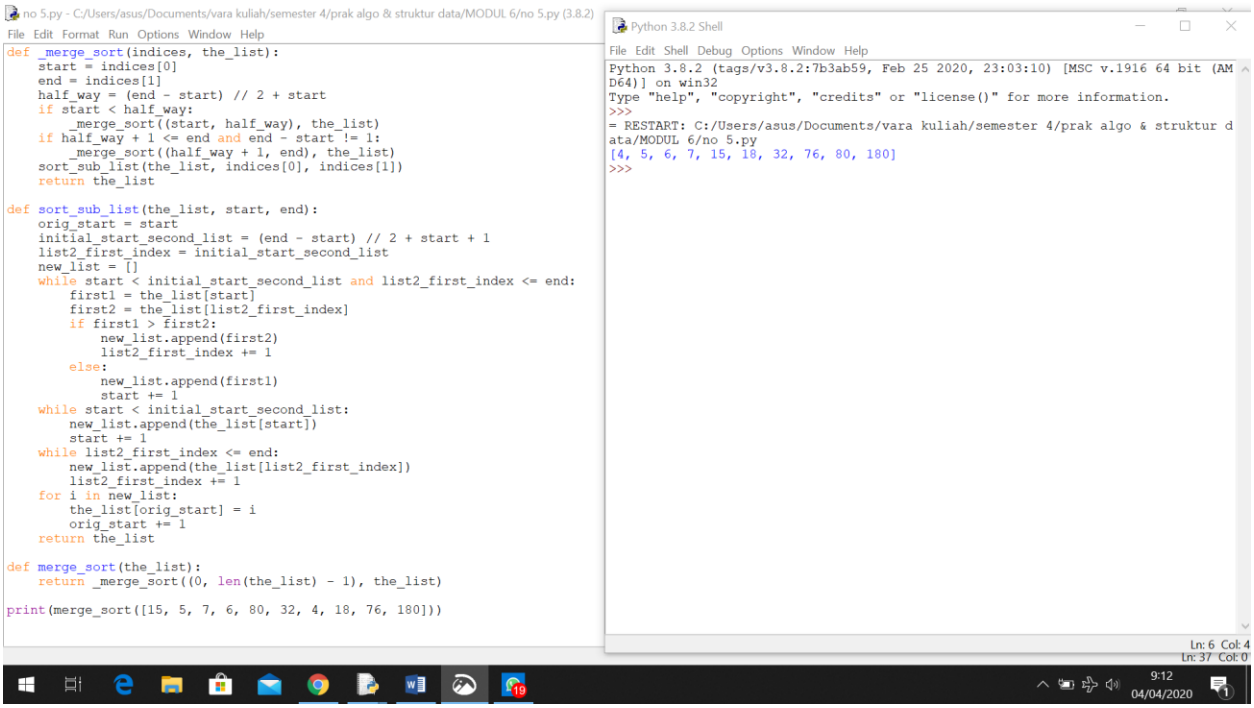
low high

Pivot



low high

5.



no 5.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 5.py (3.8.2)

File Edit Format Run Options Window Help

```
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([15, 5, 7, 6, 80, 32, 4, 18, 76, 180]))
```

Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur d
ata/MODUL 6/no 5.py
[4, 5, 6, 7, 15, 18, 32, 76, 80, 180]
>>>
```

Ln: 6 Col: 4

Ln: 37 Col: 0

9:12

04/04/2020

6.

no 6.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 6.py (3.8.2)

File Edit Format Run Options Window Help

```
def quickSort(S):
    quicksorthelp(S, 0, len(S))

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

def Partition(S, low, high):
    result = 0
    pivot, pidx = median_of_three(S, low, high)
    S[low], S[pidx] = S[pidx], S[low]
    i = low + 1
    for j in range(low + 1, high, 1):
        result += 1
        if S[j] < pivot:
            S[i], S[j] = S[j], S[i]
            i += 1
    S[low], S[i - 1] = S[i - 1], S[low]
    return i - 1, result

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

listin = [15, 5, 7, 6, 80, 32, 4, 18, 76, 180]

quickSort(listin)
```

Ln: 40 Col: 45

no 6.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 6.py (3.8.2)

File Edit Format Run Options Window Help

```
def quicksorthelp(S, low, high):
    result = 0
    if low < high:
        pivot_location, result = Partition(S, low, high)
        result += quicksorthelp(S, low, pivot_location)
        result += quicksorthelp(S, pivot_location + 1, high)
    return result

def Partition(S, low, high):
    result = 0
    pivot, pidx = median_of_three(S, low, high)
    S[low], S[pidx] = S[pidx], S[low]
    i = low + 1
    for j in range(low + 1, high, 1):
        result += 1
        if S[j] < pivot:
            S[i], S[j] = S[j], S[i]
            i += 1
    S[low], S[i - 1] = S[i - 1], S[low]
    return i - 1, result

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

listin = [15, 5, 7, 6, 80, 32, 4, 18, 76, 180]

quickSort(listin)
print(listin)
```

Ln: 40 Col: 45

Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur d
ata/MODUL 6/no 5.py
[4, 5, 6, 7, 15, 18, 32, 76, 80, 180]
>>>
= RESTART: C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur d
ata/MODUL 6/no 6.py
[4, 5, 6, 7, 15, 18, 32, 76, 80, 180]
>>>
```

Ln: 9 Col: 4

7.

no 7.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 7.py (3.8.2)

File Edit Format Run Options Window Help

```
from time import time as detak
from random import shuffle as kocok
import time
k = [i for i in range(1,6000)]
kocok(k)

def mergeSort(S):
    if len(S) >1:
        mid = len(S)//2
        L = S[:mid]
        R = S[mid:]
        mergeSort(L)
        mergeSort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                S[k] = L[i]
                i+=1
            else:
                S[k] = R[j]
                j+=1
            k+=1
        while i < len(L):
            S[k] = L[i]
            i+=1
            k+=1
        while j < len(R):
            S[k] = R[j]
            j+=1
            k+=1
def partition(S,low,high):
    i = ( low-1 )
    pivot = S[high]
    for j in range(low , high):
        if   S[j] <= pivot:
            i = i+1
            S[i],S[j] = S[j], S[i]
    S[i+1],S[high] = S[high], S[i+1]
    return ( i+1 )

def quickSort(S,low,high):
    if low < high:
```

Ln: 98 Col: 1

no 7.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 7.py (3.8.2)

File Edit Format Run Options Window Help

```
def quickSort(S,low,high):
    if low < high:
        pi = partition(S,low,high)
        quickSort(S, low, pi-1)
        quickSort(S, 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])

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
```

Ln: 98 Col: 1

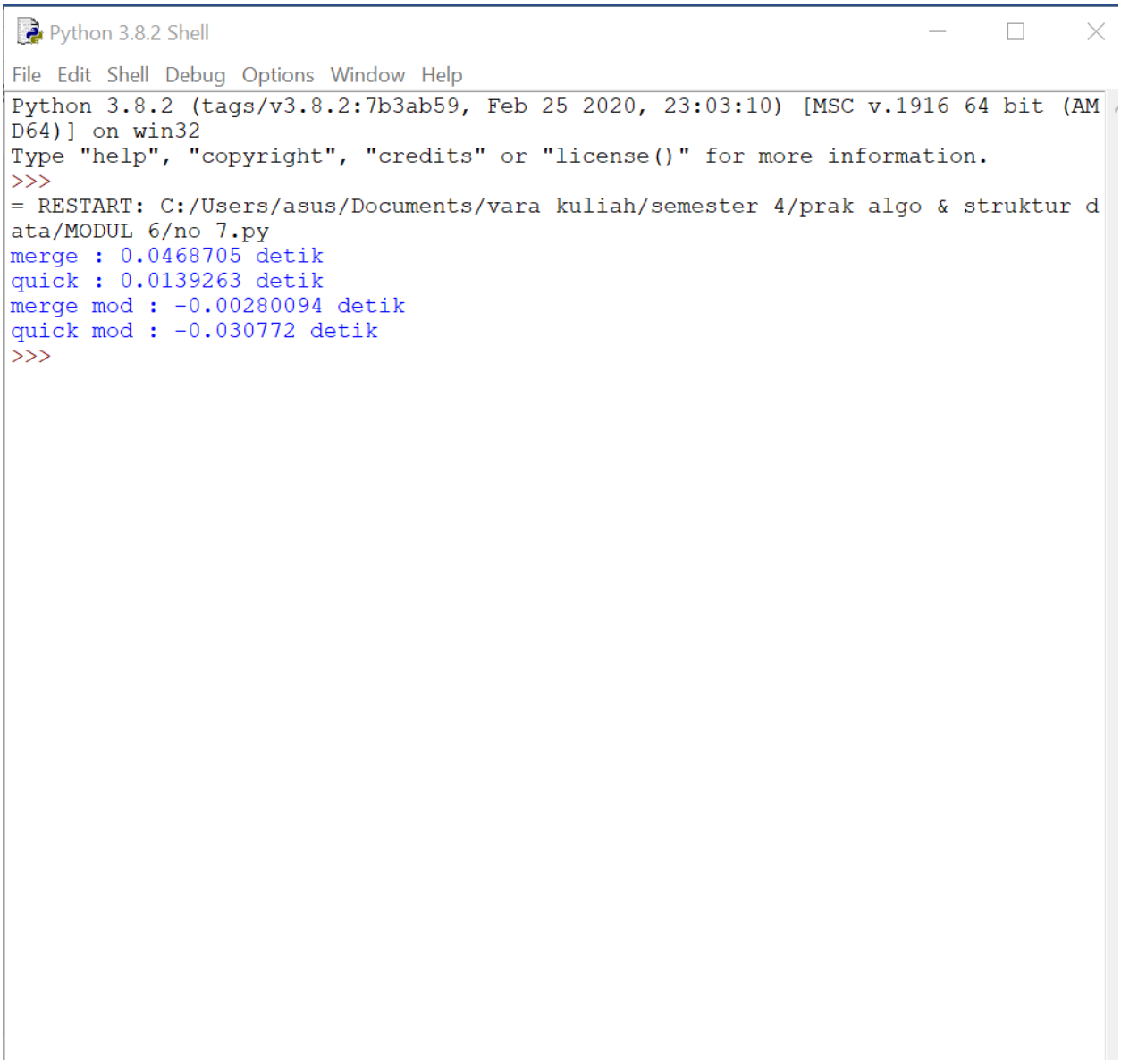
no 7.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 7.py (3.8.2)

File Edit Format Run Options Window Help

```
        list2_first_index += 1
    for i in new_list:
        the_list[orig_start] = i
        orig_start += 1

def merge_sort(the_list):
    return _merge_sort((0, len(the_list) - 1), the_list)

def quickSortMOD(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:
```

Ln: 98 Col: 1

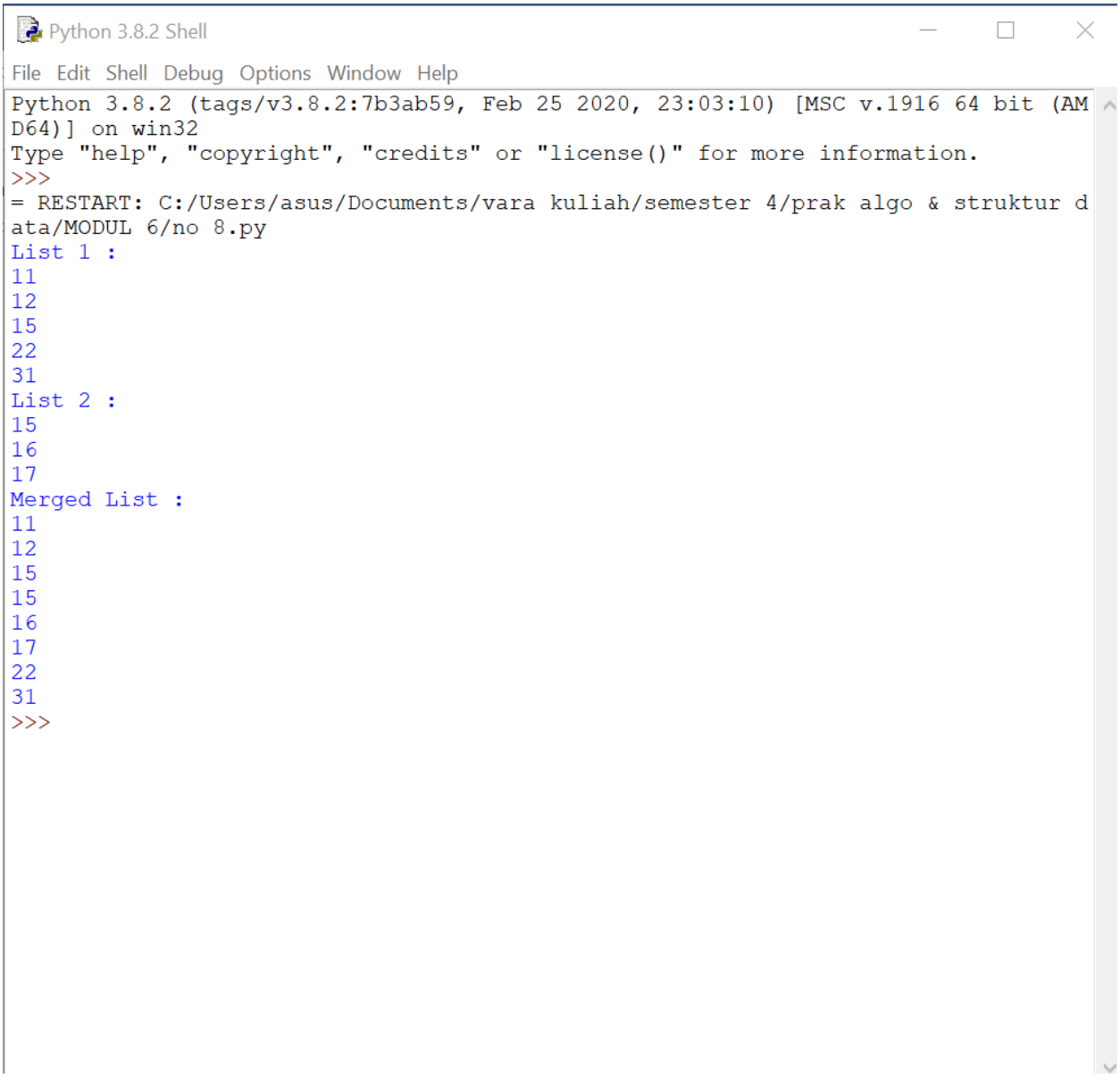

no 7.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 7.py (3.8.2)
File Edit Format Run Options Window Help
pivot_location, result = Partition(L, low, high, ascending)
result += quicksorthelp(L, low, pivot_location, ascending)
result += quicksorthelp(L, pivot_location + 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
mer = k[:]
qui = k[:]
mer2 = k[:]
qui2 = k[:]
aw=detak();mergeSort(mer);ak=detak();print('merge : %g detik' %(ak-aw));
aw=detak();quickSort(qui,0,len(qui)-1);ak=detak();print('quick : %g detik' %(ak-aw));
aw=detak();merge_sort(mer2);print('merge mod : %g detik' %(ak-aw));
aw=detak();quickSortMOD(qui2, False);print('quick mod : %g detik' %(ak-aw));


Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur d
ata/MODUL 6/no 7.py
merge : 0.0468705 detik
quick : 0.0139263 detik
merge mod : -0.00280094 detik
quick mod : -0.030772 detik
>>>

8.


no 8.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 8.py (3.8.2)
File Edit Format Run Options Window Help
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):
Ln: 79 Col: 0


no 8.py - C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 8.py (3.8.2)
File Edit Format Run Options Window Help
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(15)
list1.appendSorted(11)
list1.appendSorted(31)
list1.appendSorted(12)
list1.appendSorted(22)
print("List 1 :"),
list1.printList()
list2 = LinkedList()
list2.appendSorted(16)
list2.appendSorted(17)
list2.appendSorted(15)
print("List 2 :"),
list2.printList()
list3 = LinkedList()
list3.head = list3.mergeSorted(list1.head, list2.head)
print("Merged List :"),
list3.printList()
Ln: 79 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, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/asus/Documents/vara kuliah/semester 4/prak algo & struktur data/MODUL 6/no 8.py
List 1 :
11
12
15
22
31
List 2 :
15
16
17
Merged List :
11
12
15
15
16
17
22
31
>>>