Nama : Malik

Muhammad

NIM : L200180072

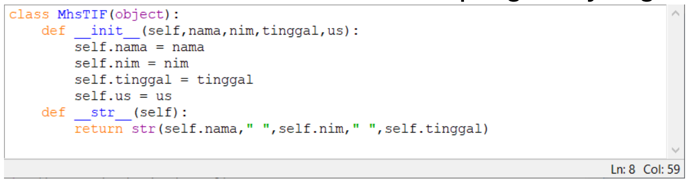
Kelas : C

# Modul 6 – Pengurutan Lanjutan

## Soal-soal untuk mahasiswa

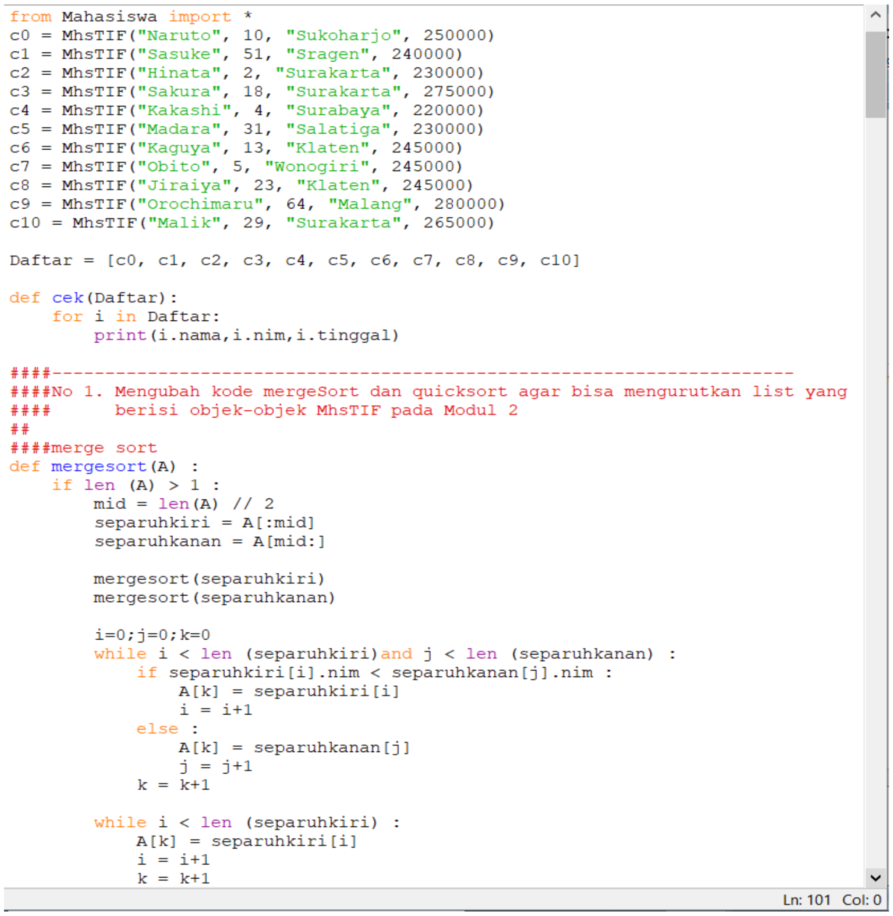
1. Mengubah kode mergeSort dan quicksort agar bisa mengurutkan list yang berisiobjek- objek MhsTIF pada Modul2

   Berikut adalah screenshoot dari program yang saya buat:



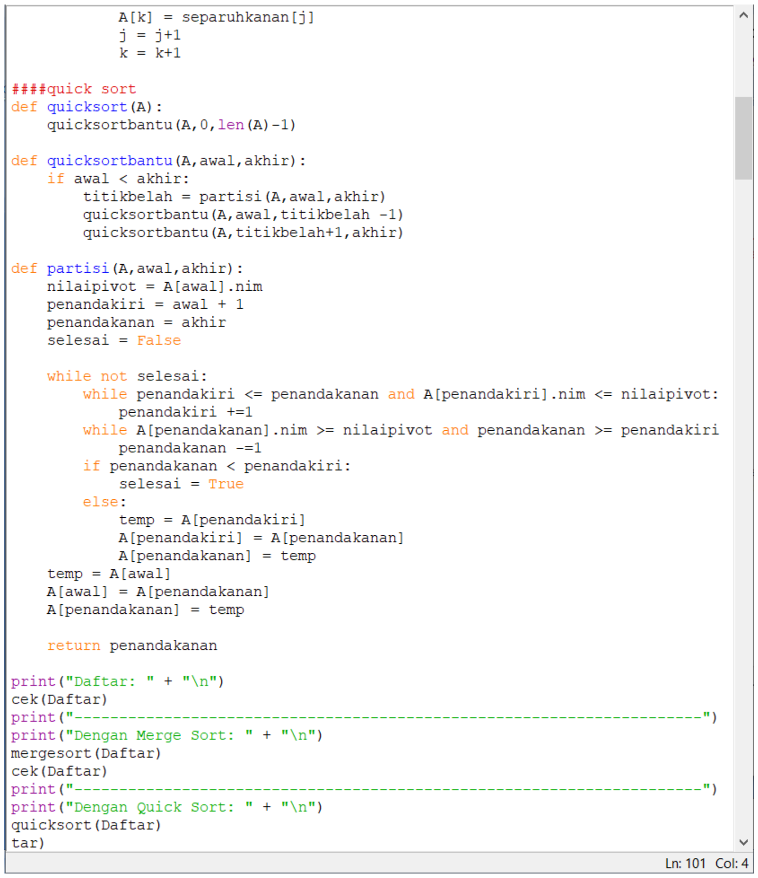
```
class MhsTIF(object):
    def __init__(self,nama,nim,tinggal,us):
        self.nama = nama
        self.nim = nim
        self.tinggal = tinggal
        self.us = us
    def __str__(self):
        return str(self.nama," ",self.nim," ",self.tinggal)
```

Ln: 8 Col: 59



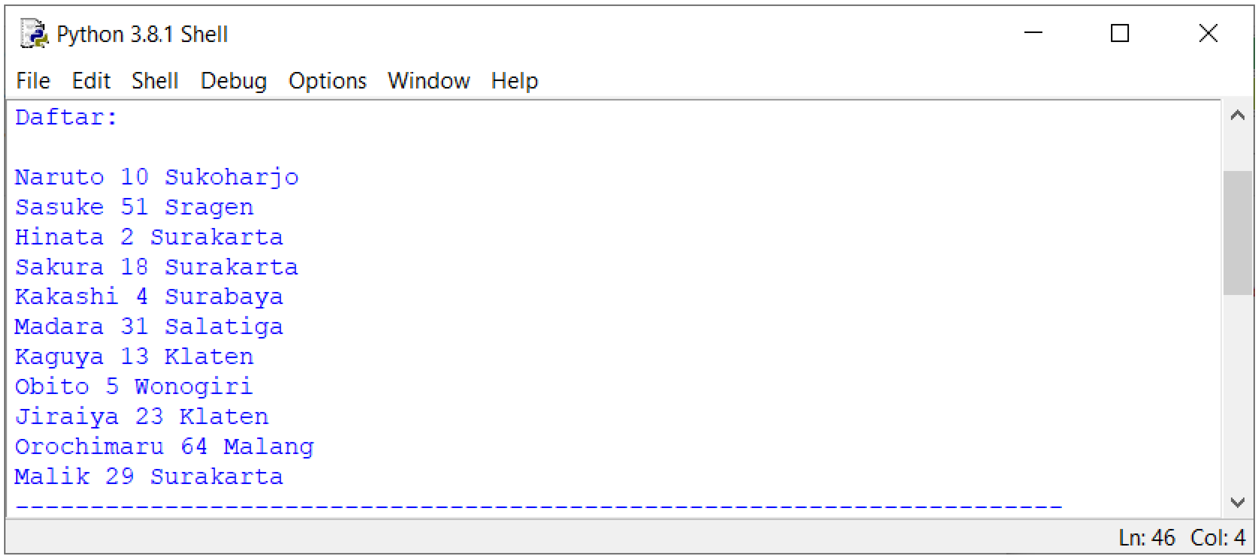
```
from Mahasiswa import *
c0 = MhsTIF("Naruto", 10, "Sukoharjo", 250000)
c1 = MhsTIF("Sasuke", 51, "Sragen", 240000)
c2 = MhsTIF("Hinata", 2, "Surakarta", 230000)
c3 = MhsTIF("Sakura", 18, "Surakarta", 275000)
c4 = MhsTIF("Kakashi", 4, "Surabaya", 220000)
c5 = MhsTIF("Madara", 31, "Salatiga", 230000)
c6 = MhsTIF("Kaguya", 13, "Klaten", 245000)
c7 = MhsTIF("Obito", 5, "Wonogiri", 245000)
c8 = MhsTIF("Jiraiya", 23, "Klaten", 245000)
c9 = MhsTIF("Orochimaru", 64, "Malang", 280000)
c10 = MhsTIF("Malik", 29, "Surakarta", 265000)

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

def cek(Daftar):
    for i in Daftar:
        print(i.nama,i.nim,i.tinggal)

####----------------------------------------------------------------------
####No 1. Mengubah kode mergeSort dan quicksort agar bisa mengurutkan list yang
####      berisi objek-objek MhsTIF pada Modul 2
##
####merge sort
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].nim < separuhkanan[j].nim :
                A[k] = separuhkiri[i]
                i = i+1
            else :
                A[k] = separuhkanan[j]
                j = j+1
            k = k+1

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

Ln: 101 Col: 0

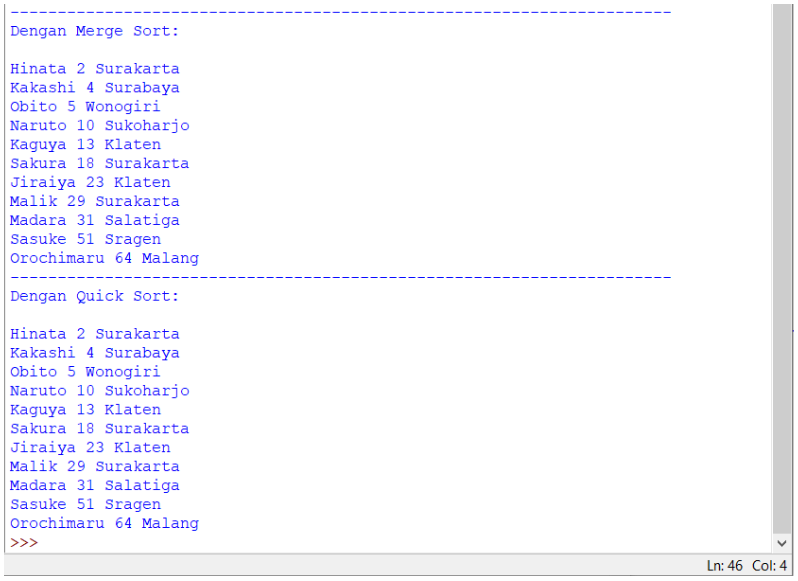


```
            A[k] = separuhkanan[j]
            j = j+1
            k = k+1

####quick sort
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].nim
    penandakiri = awal + 1
    penandakanan = akhir
    selesai = False

    while not selesai:
        while penandakiri <= penandakanan and A[penandakiri].nim <= nilaipivot:
            penandakiri +=1
        while A[penandakanan].nim >= 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[penandakanan]
    A[penandakanan] = temp

    return penandakanan

print("Daftar: " + "\n")
cek(Daftar)
print("----------------------------------------------------------------------")
print("Dengan Merge Sort: " + "\n")
mergesort(Daftar)
cek(Daftar)
print("----------------------------------------------------------------------")
print("Dengan Quick Sort: " + "\n")
quicksort(Daftar)
tar)
```

Berikut adalah screenshoot saat program dijalankan:

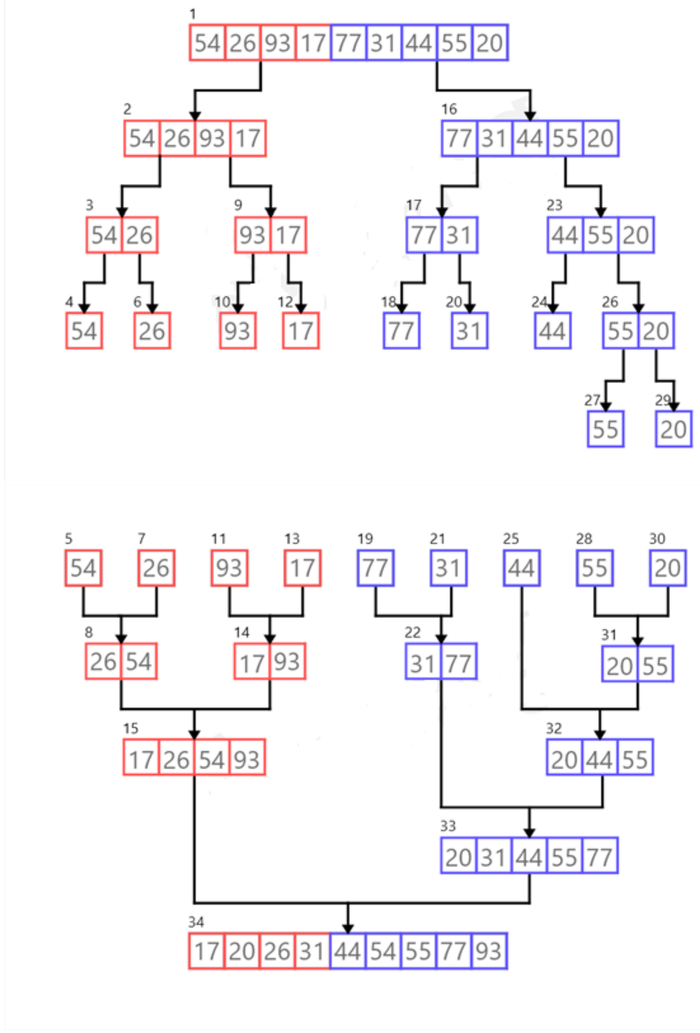


```
Daftar:

Naruto 10 Sukoharjo
Sasuke 51 Sragen
Hinata 2 Surakarta
Sakura 18 Surakarta
Kakashi 4 Surabaya
Madara 31 Salatiga
Kaguya 13 Klaten
Obito 5 Wonogiri
Jiraiya 23 Klaten
Orochimaru 64 Malang
Malik 29 Surakarta
----------------------------------------------------------------------
```



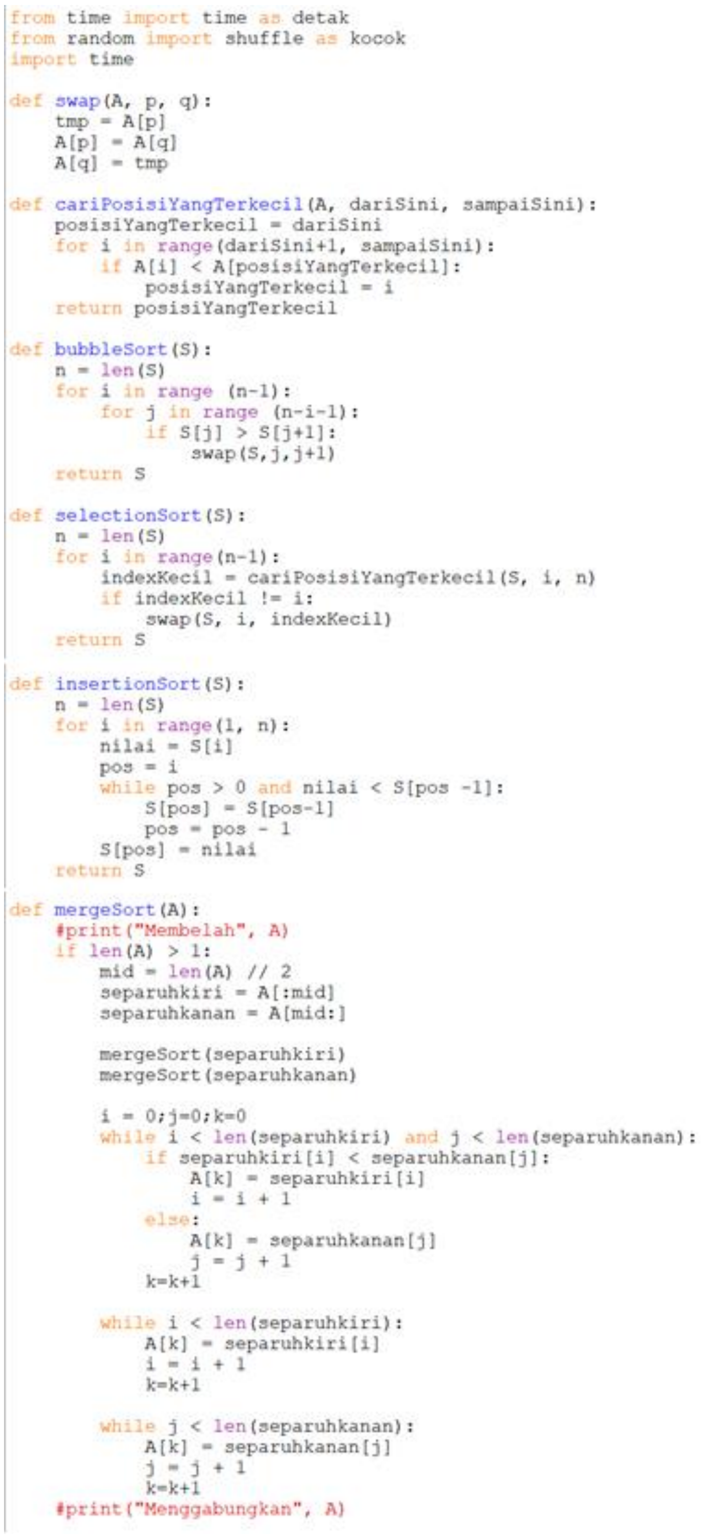
```
----------------------------------------------------------------------
Dengan Merge Sort:

Hinata 2 Surakarta
Kakashi 4 Surabaya
Obito 5 Wonogiri
Naruto 10 Sukoharjo
Kaguya 13 Klaten
Sakura 18 Surakarta
Jiraiya 23 Klaten
Malik 29 Surakarta
Madara 31 Salatiga
Sasuke 51 Sragen
Orochimaru 64 Malang
----------------------------------------------------------------------
Dengan Quick Sort:

Hinata 2 Surakarta
Kakashi 4 Surabaya
Obito 5 Wonogiri
Naruto 10 Sukoharjo
Kaguya 13 Klaten
Sakura 18 Surakarta
Jiraiya 23 Klaten
Malik 29 Surakarta
Madara 31 Salatiga
Sasuke 51 Sragen
Orochimaru 64 Malang
>>>
```

2. Menandai dan memberi nomer urut eksekusi proses pada modul halaman58

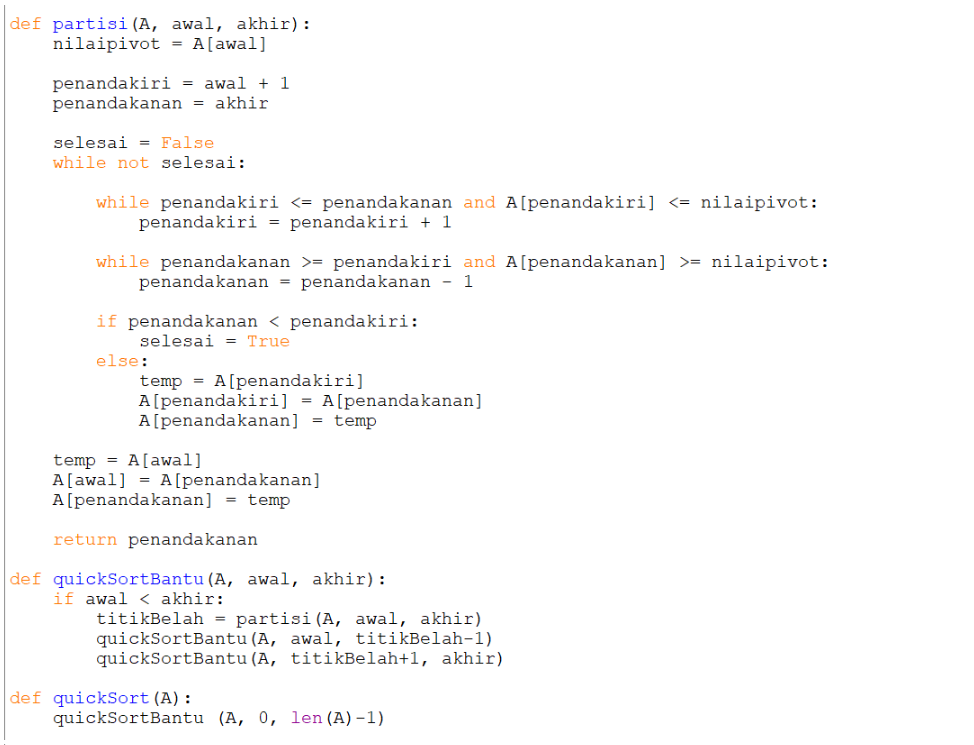


3. Menguji kecepatan mergeSort danquicksort

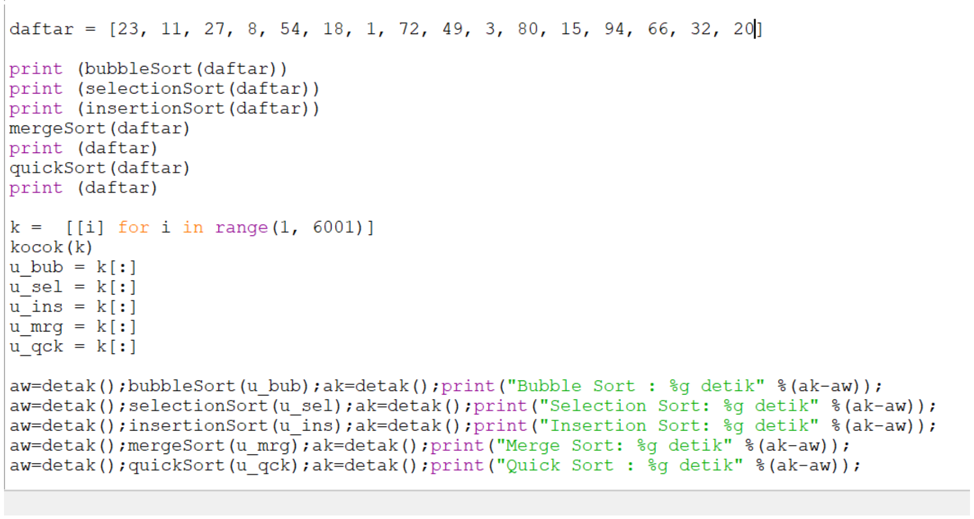
   Berikut adalah screenshoot dari program yang saya buat:



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

def swap(A, p, q):
    tmp = A[p]
    A[p] = A[q]
    A[q] = tmp

def cariPosisiYangTerkecil(A, dariSini, sampaiSini):
    posisiYangTerkecil = dariSini
    for i in range(dariSini+1, sampaiSini):
        if A[i] < A[posisiYangTerkecil]:
            posisiYangTerkecil = i
    return posisiYangTerkecil

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

def selectionSort(S):
    n = len(S)
    for i in range(n-1):
        indexKecil = cariPosisiYangTerkecil(S, i, n)
        if indexKecil != i:
            swap(S, i, indexKecil)
    return S

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

def mergeSort(A):
    #print("Membelah", 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 = i + 1
            else:
                A[k] = separuhkanan[j]
                j = j + 1
            k=k+1

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

        while j < len(separuhkanan):
            A[k] = separuhkanan[j]
            j = j + 1
            k=k+1
    #print("Menggabungkan", A)
```

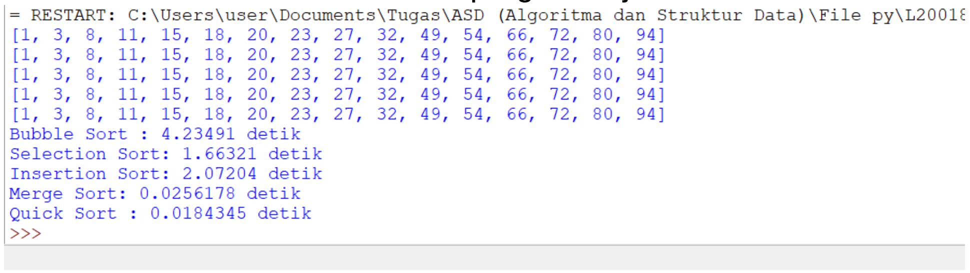


```
def partisi(A, awal, akhir):
    nilaipivot = A[awal]

    penandakiri = awal + 1
    penandakanan = akhir

    selesai = False
    while not selesai:

        while penandakiri <= penandakanan and A[penandakiri] <= nilaipivot:
            penandakiri = penandakiri + 1

        while penandakanan >= penandakiri and A[penandakanan] >= nilaipivot:
            penandakanan = penandakanan - 1

        if penandakanan < penandakiri:
            selesai = True
        else:
            temp = A[penandakiri]
            A[penandakiri] = A[penandakanan]
            A[penandakanan] = temp

    temp = A[awal]
    A[awal] = A[penandakanan]
    A[penandakanan] = temp

    return penandakanan

def quickSortBantu(A, awal, akhir):
    if awal < akhir:
        titikBelah = partisi(A, awal, akhir)
        quickSortBantu(A, awal, titikBelah-1)
        quickSortBantu(A, titikBelah+1, akhir)

def quickSort(A):
    quickSortBantu (A, 0, len(A)-1)
```



```
daftar = [23, 11, 27, 8, 54, 18, 1, 72, 49, 3, 80, 15, 94, 66, 32, 20]

print (bubbleSort(daftar))
print (selectionSort(daftar))
print (insertionSort(daftar))
mergeSort(daftar)
print (daftar)
quickSort(daftar)
print (daftar)

k =  [[i] for i in range(1, 6001)]
kocok(k)
u_bub = k[:]
u_sel = k[:]
u_ins = k[:]
u_mrg = k[:]
u_qck = k[:]

aw=detak();bubbleSort(u_bub);ak=detak();print("Bubble Sort : %g detik" %(ak-aw));
aw=detak();selectionSort(u_sel);ak=detak();print("Selection Sort: %g detik" %(ak-aw));
aw=detak();insertionSort(u_ins);ak=detak();print("Insertion Sort: %g detik" %(ak-aw));
aw=detak();mergeSort(u_mrg);ak=detak();print("Merge Sort: %g detik" %(ak-aw));
aw=detak();quickSort(u_qck);ak=detak();print("Quick Sort : %g detik" %(ak-aw));
```

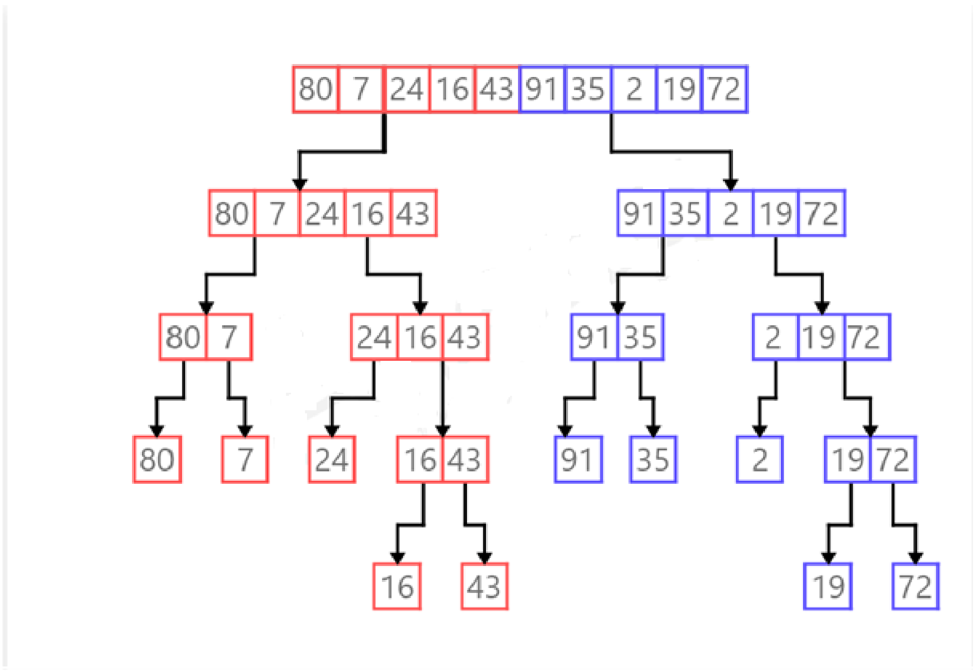
Berikut adalah screenshoot saat program dijalankan:

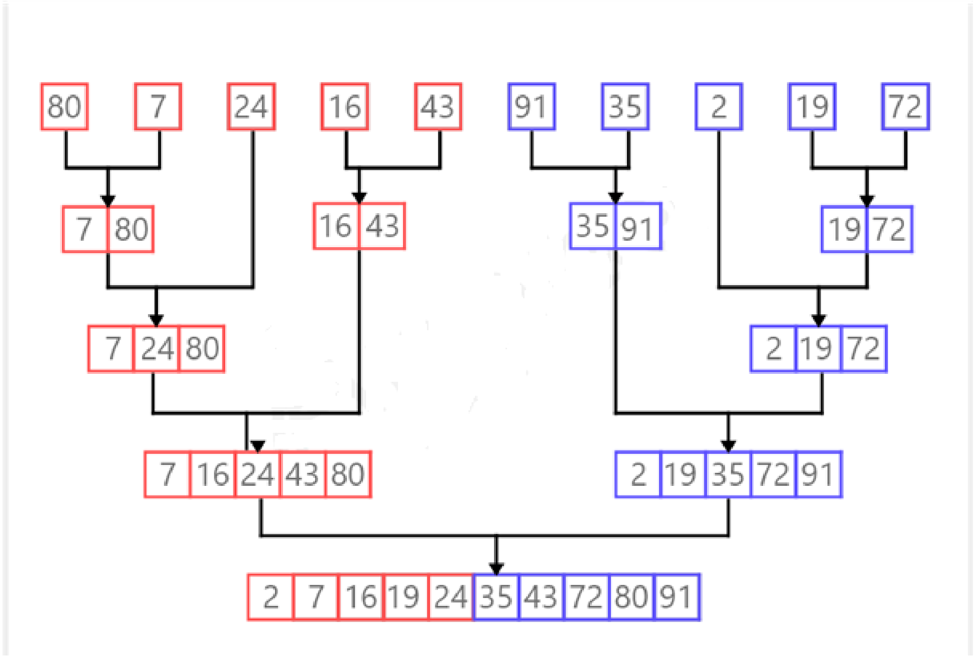


```
= RESTART: C:\Users\user\Documents\Tugas\ASD (Algoritma dan Struktur Data)\File py\L2001
[1, 3, 8, 11, 15, 18, 20, 23, 27, 32, 49, 54, 66, 72, 80, 94]
[1, 3, 8, 11, 15, 18, 20, 23, 27, 32, 49, 54, 66, 72, 80, 94]
[1, 3, 8, 11, 15, 18, 20, 23, 27, 32, 49, 54, 66, 72, 80, 94]
[1, 3, 8, 11, 15, 18, 20, 23, 27, 32, 49, 54, 66, 72, 80, 94]
[1, 3, 8, 11, 15, 18, 20, 23, 27, 32, 49, 54, 66, 72, 80, 94]
Bubble Sort : 4.23491 detik
Selection Sort: 1.66321 detik
Insertion Sort: 2.07204 detik
Merge Sort: 0.0256178 detik
Quick Sort : 0.0184345 detik
>>>
```

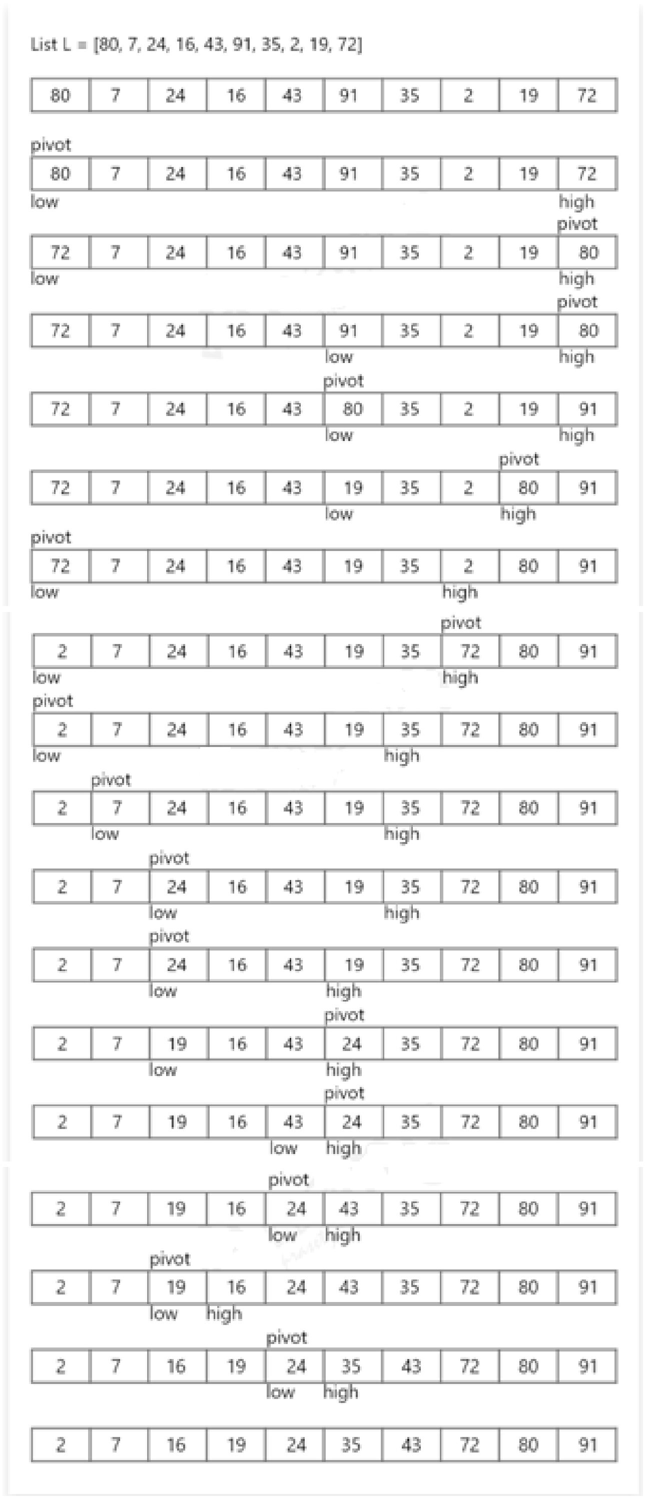
4. Menggambar trace pengurutan untuk algortima berdasarkan listL

   a. Mergesort





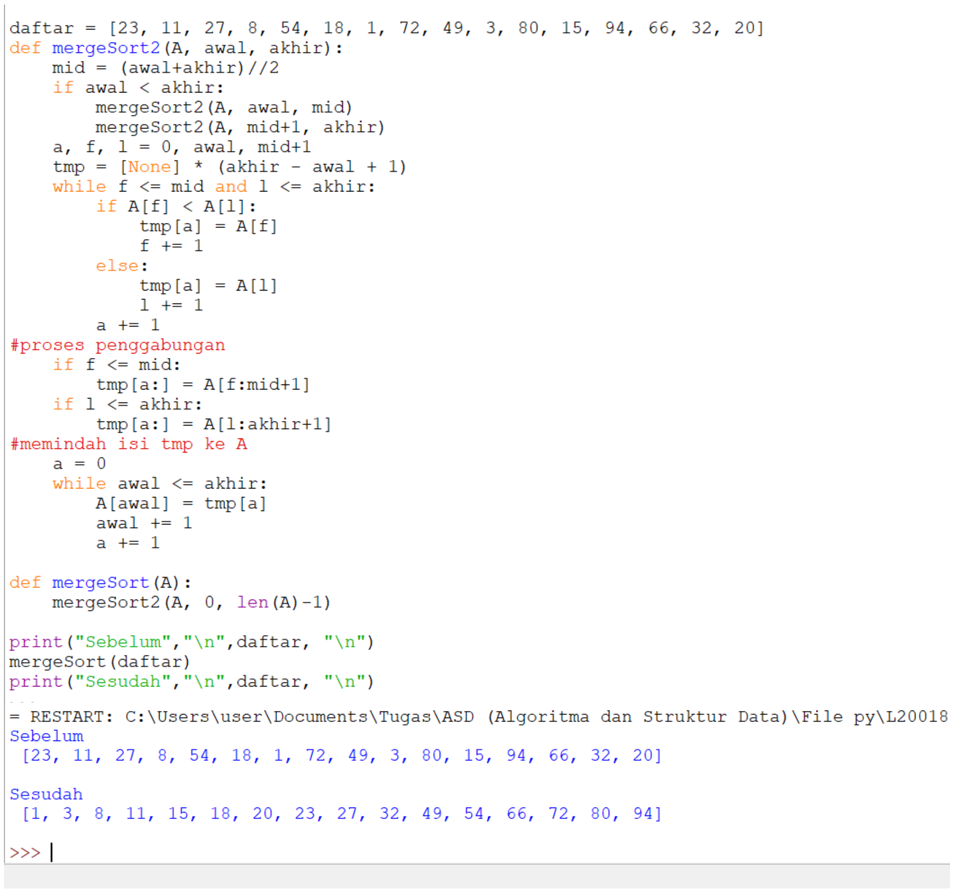
b. Quicksort


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

5. Meningkatkan efisiensi program mergeSort dengan tidak memakai operator slice dan lalu mem-pass index awal dan index akhir Bersama list-nya saat memanggil mergeSort secararekusif.

   Berikut adalah screenshoot dari program yang saya buat:

   Berikut adalah screenshoot saat program dijalankan:



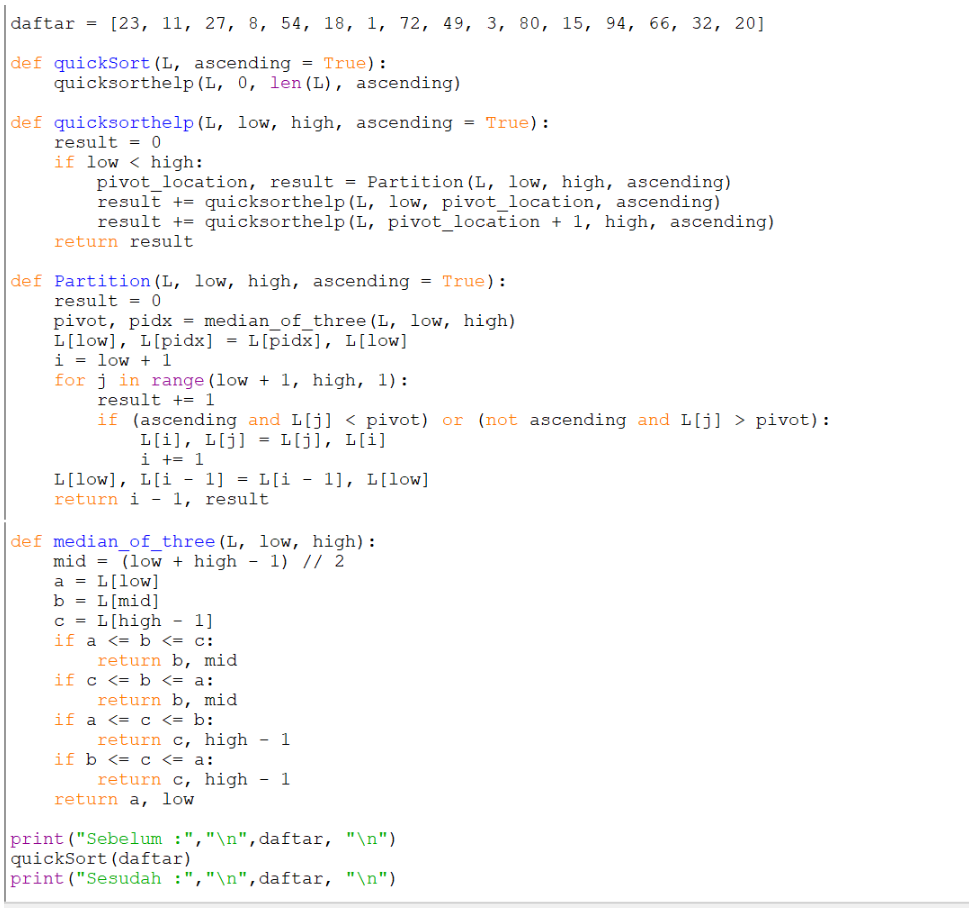
```
daftar = [23, 11, 27, 8, 54, 18, 1, 72, 49, 3, 80, 15, 94, 66, 32, 20]
def mergeSort2(A, awal, akhir):
    mid = (awal+akhir)//2
    if awal < akhir:
        mergeSort2(A, awal, mid)
        mergeSort2(A, mid+1, akhir)
    a, f, l = 0, awal, mid+1
    tmp = [None] * (akhir - awal + 1)
    while f <= mid and l <= akhir:
        if A[f] < A[l]:
            tmp[a] = A[f]
            f += 1
        else:
            tmp[a] = A[l]
            l += 1
        a += 1
#proses penggabungan
    if f <= mid:
        tmp[a:] = A[f:mid+1]
    if l <= akhir:
        tmp[a:] = A[l:akhir+1]
#memindah isi tmp ke A
    a = 0
    while awal <= akhir:
        A[awal] = tmp[a]
        awal += 1
        a += 1

def mergeSort(A):
    mergeSort2(A, 0, len(A)-1)

print("Sebelum","\n",daftar, "\n")
mergeSort(daftar)
print("Sesudah","\n",daftar, "\n")
```

```
= RESTART: C:\Users\user\Documents\Tugas\ASD (Algoritma dan Struktur Data)\File py\L20018
Sebelum
 [23, 11, 27, 8, 54, 18, 1, 72, 49, 3, 80, 15, 94, 66, 32, 20]

Sesudah
 [1, 3, 8, 11, 15, 18, 20, 23, 27, 32, 49, 54, 66, 72, 80, 94]

>>> |
```

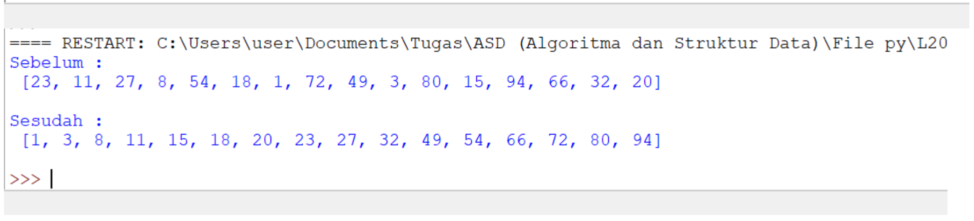
6. Meningkatkan efisien program quicksort dengan memakai metode median-dari-tiga untuk memilihpivotnya

   Berikut adalah screenshoot dari program yang saya buat:

   Berikut adalah screenshoot saat program dijalankan:



```
daftar = [23, 11, 27, 8, 54, 18, 1, 72, 49, 3, 80, 15, 94, 66, 32, 20]

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

print("Sebelum :","\n",daftar, "\n")
quickSort(daftar)
print("Sesudah :","\n",daftar, "\n")
```

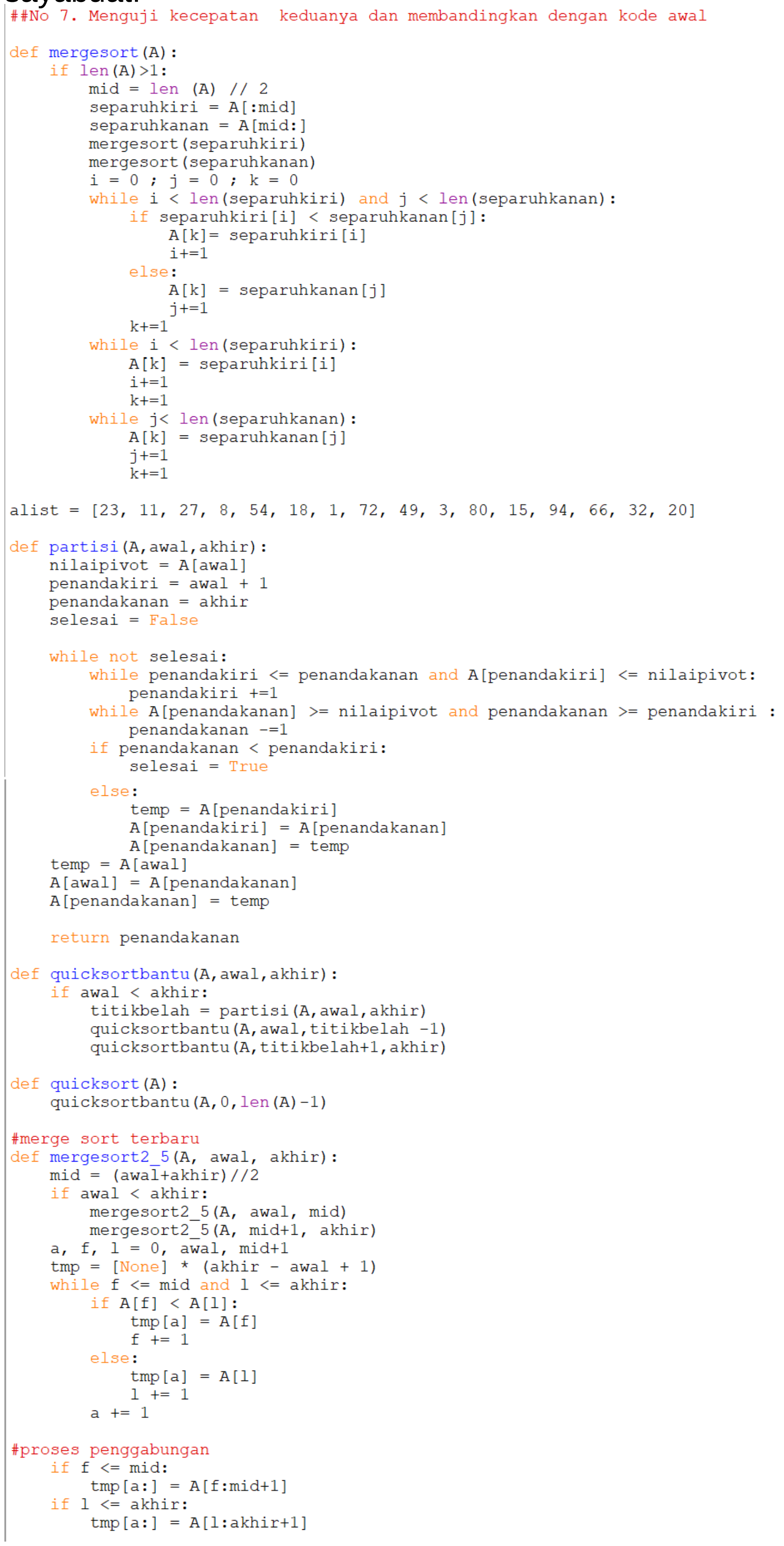


```
==== RESTART: C:\Users\user\Documents\Tugas\ASD (Algoritma dan Struktur Data)\File py\L20
Sebelum :
 [23, 11, 27, 8, 54, 18, 1, 72, 49, 3, 80, 15, 94, 66, 32, 20]

Sesudah :
 [1, 3, 8, 11, 15, 18, 20, 23, 27, 32, 49, 54, 66, 72, 80, 94]

>>> |
```

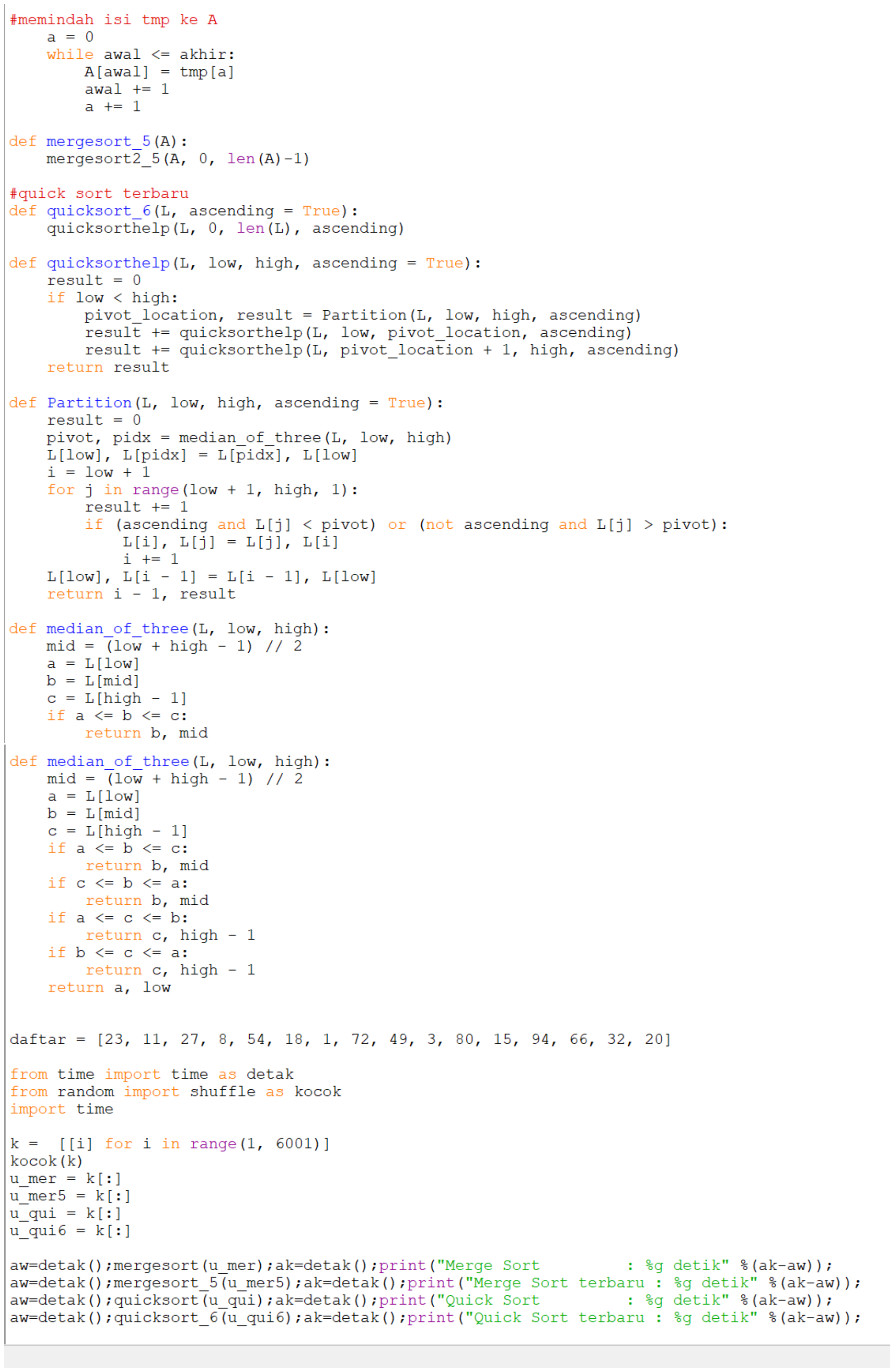
7. Menguji kecepatan keduanya dan membandingkan dengan

   kode awal Berikut adalah screenshoot dari program yang

   sayabuat:

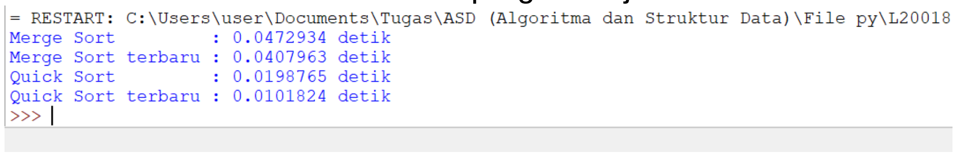


```
##No 7. Menguji kecepatan  keduanya dan membandingkan dengan kode awal

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):
            A[k] = separuhkanan[j]
            j+=1
            k+=1

alist = [23, 11, 27, 8, 54, 18, 1, 72, 49, 3, 80, 15, 94, 66, 32, 20]

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 +=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[penandakanan]
    A[penandakanan] = temp

    return penandakanan

def quicksortbantu(A,awal,akhir):
    if awal < akhir:
        titikbelah = partisi(A,awal,akhir)
        quicksortbantu(A,awal,titikbelah -1)
        quicksortbantu(A,titikbelah+1,akhir)

def quicksort(A):
    quicksortbantu(A,0,len(A)-1)

#merge sort terbaru
def mergesort2_5(A, awal, akhir):
    mid = (awal+akhir)//2
    if awal < akhir:
        mergesort2_5(A, awal, mid)
        mergesort2_5(A, mid+1, akhir)
    a, f, l = 0, awal, mid+1
    tmp = [None] * (akhir - awal + 1)
    while f <= mid and l <= akhir:
        if A[f] < A[l]:
            tmp[a] = A[f]
            f += 1
        else:
            tmp[a] = A[l]
            l += 1
        a += 1

#proses penggabungan
    if f <= mid:
        tmp[a:] = A[f:mid+1]
    if l <= akhir:
        tmp[a:] = A[l:akhir+1]
```



```
#memindah isi tmp ke A
    a = 0
    while awal <= akhir:
        A[awal] = tmp[a]
        awal += 1
        a += 1

def mergesort_5(A):
    mergesort2_5(A, 0, len(A)-1)

#quick sort terbaru
def quicksort_6(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
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


daftar = [23, 11, 27, 8, 54, 18, 1, 72, 49, 3, 80, 15, 94, 66, 32, 20]

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

k =  [[i] for i in range(1, 6001)]
kocok(k)
u_mer = k[:]
u_mer5 = k[:]
u_qui = k[:]
u_qui6 = k[:]

aw=detak();mergesort(u_mer);ak=detak();print("Merge Sort         : %g detik" %(ak-aw));
aw=detak();mergesort_5(u_mer5);ak=detak();print("Merge Sort terbaru : %g detik" %(ak-aw));
aw=detak();quicksort(u_qui);ak=detak();print("Quick Sort         : %g detik" %(ak-aw));
aw=detak();quicksort_6(u_qui6);ak=detak();print("Quick Sort terbaru : %g detik" %(ak-aw));
```

Berikut adalah screenshoot saat program dijalankan:

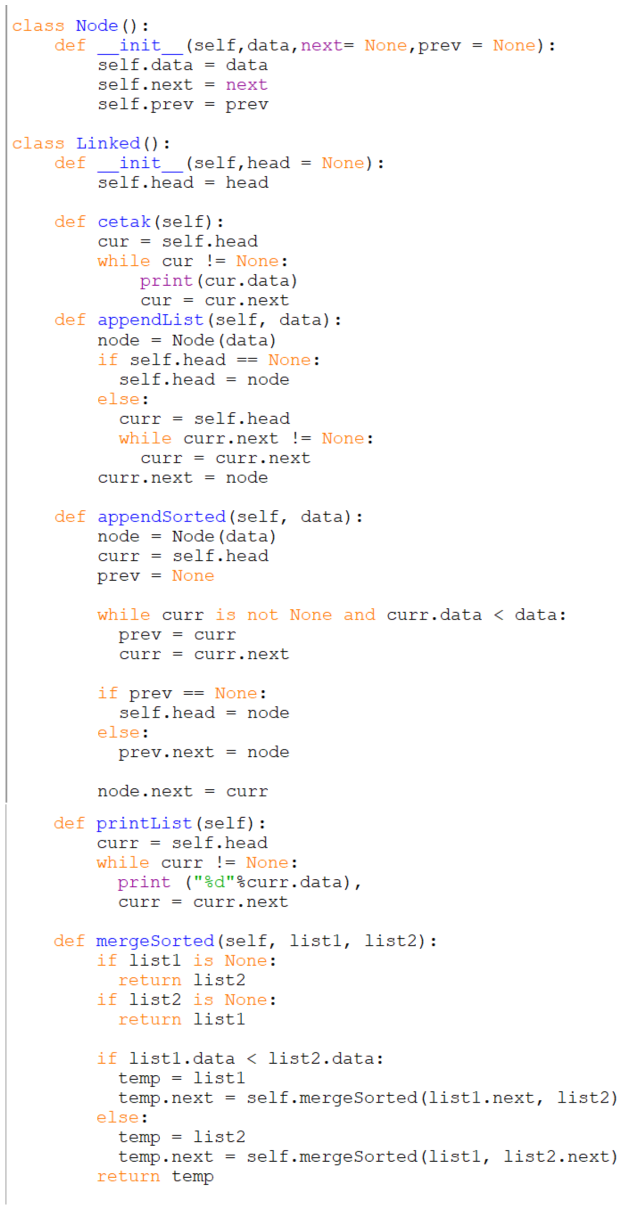


```
= RESTART: C:\Users\user\Documents\Tugas\ASD (Algoritma dan Struktur Data)\File py\L20018
Merge Sort         : 0.0472934 detik
Merge Sort terbaru : 0.0407963 detik
Quick Sort         : 0.0198765 detik
Quick Sort terbaru : 0.0101824 detik
>>> 
```

8. Membuat versi linked-list dari program mergeSort

   diatas Berikut adalah screenshoot dari program

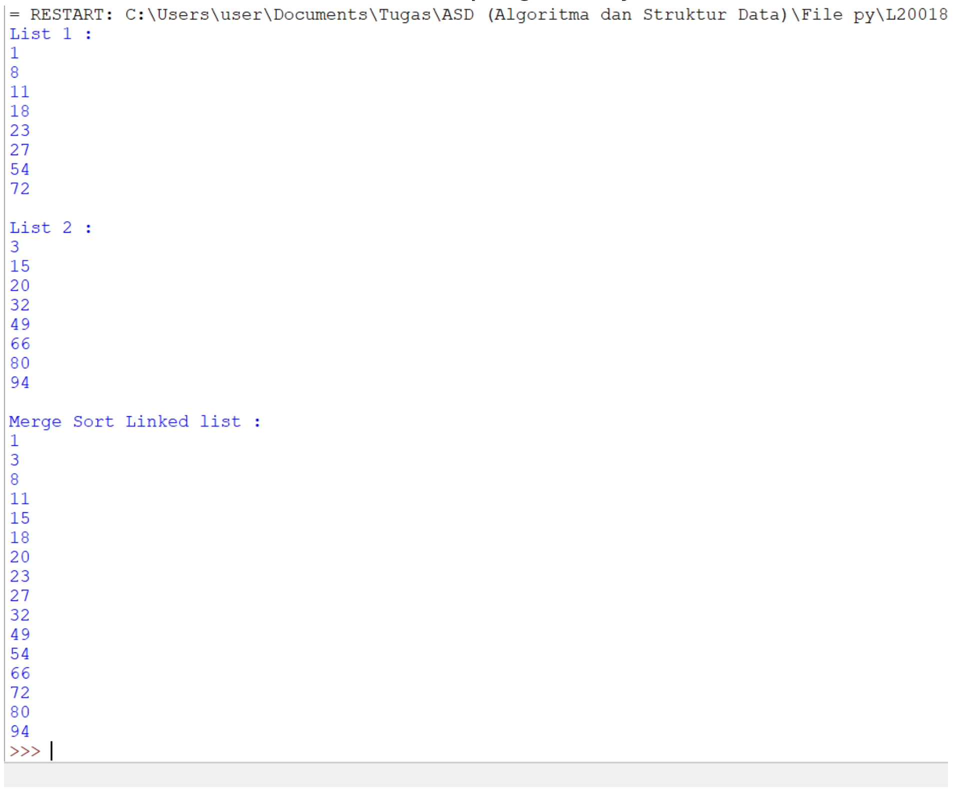
   yang sayabuat:



```
class Node():
    def __init__(self,data,next= None,prev = None):
        self.data = data
        self.next = next
        self.prev = prev

class Linked():
    def __init__(self,head = None):
        self.head = head

    def cetak(self):
        cur = self.head
        while cur != None:
            print(cur.data)
            cur = cur.next
    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 = Linked()
list1.appendSorted(23)
list1.appendSorted(11)
list1.appendSorted(27)
list1.appendSorted(8)
list1.appendSorted(54)
list1.appendSorted(18)
list1.appendSorted(1)
list1.appendSorted(72)

print("List 1 :"),
list1.printList()
print()

list2 = Linked()
list2.appendSorted(49)
list2.appendSorted(3)
list2.appendSorted(80)
list2.appendSorted(15)
list2.appendSorted(94)
list2.appendSorted(66)
list2.appendSorted(32)
list2.appendSorted(20)

print("List 2 :"),
list2.printList()
print()

list3 = Linked()
list3.head = list3.mergeSorted(list1.head, list2.head)

print("Merge Sort Linked list :"),
list3.printList()
```

Berikut adalah screenshoot saat program dijalankan:



```
= RESTART: C:\Users\user\Documents\Tugas\ASD (Algoritma dan Struktur Data)\File py\L20018
List 1 :
1
8
11
18
23
27
54
72

List 2 :
3
15
20
32
49
66
80
94

Merge Sort Linked list :
1
3
8
11
15
18
20
23
27
32
49
54
66
72
80
94
>>> |
```