

Nama : Anang Prasetyo

NIM : L200180063

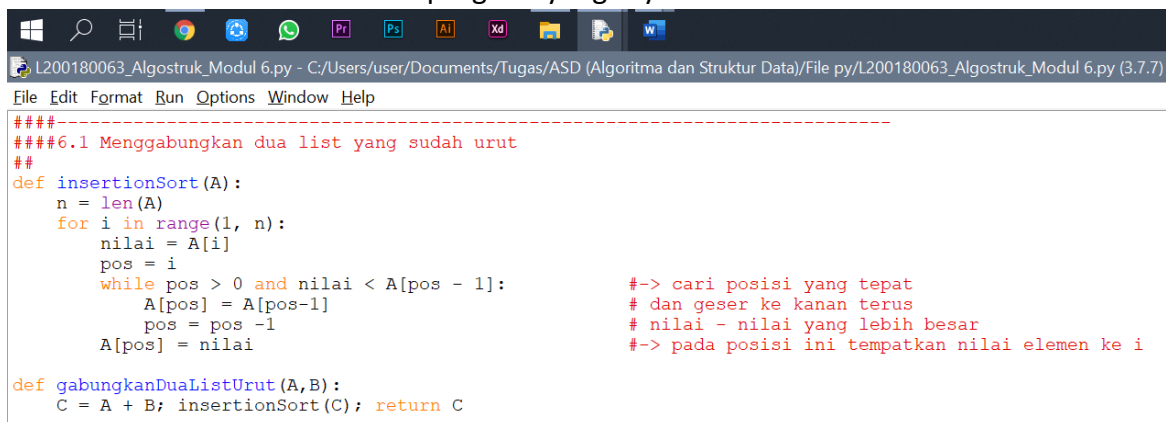
Kelas : C

## Modul 6 – Pengurutan Lanjutan

### Laporan Praktikum

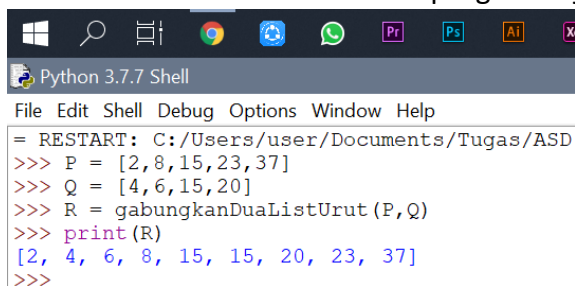
#### 6.1 Menggabungkan dua list yang sudahurut

Berikut adalah screenshoot dari program yang saya buat :



```
#####  
####6.1 Menggabungkan dua list yang sudahurut  
##  
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]          #-> cari posisi yang tepat  
            pos = pos - 1              # dan geser ke kanan terus  
        A[pos] = nilai                 # nilai - nilai yang lebih besar  
                                       #-> pada posisi ini tempatkan nilai elemen ke i  
  
def gabungkanDuaListUrut(A,B):  
    C = A + B; insertionSort(C); return C
```

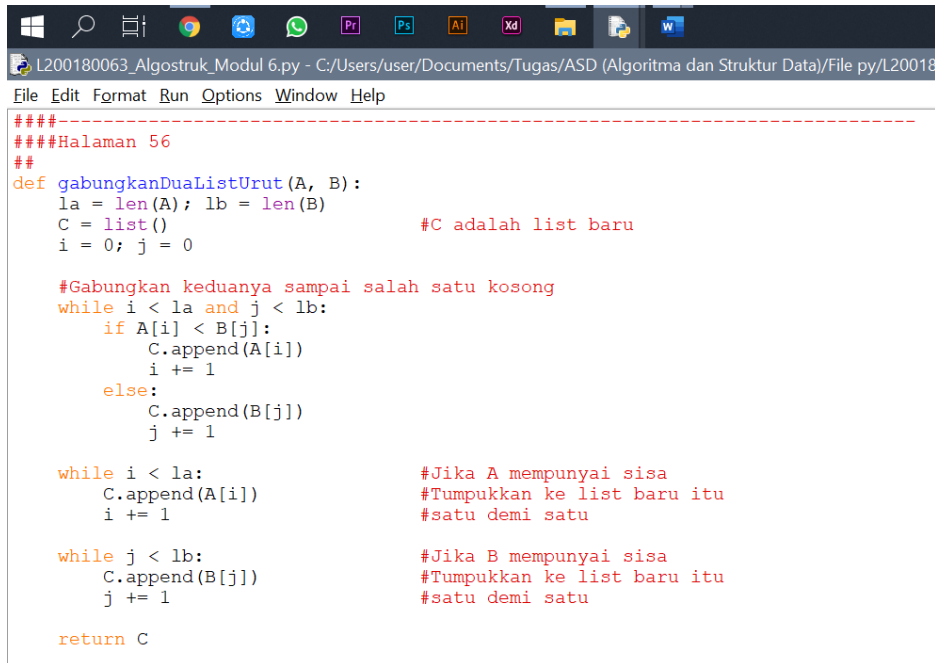
Berikut adalah screenshoot saat program dijalankan :



```
= RESTART: C:/Users/user/Documents/Tugas/ASD  
>>> P = [2,8,15,23,37]  
>>> Q = [4,6,15,20]  
>>> R = gabungkanDuaListUrut(P,Q)  
>>> print(R)  
[2, 4, 6, 8, 15, 15, 20, 23, 37]  
>>>
```

## Halaman 56

Berikut adalah screenshoot dari program yang saya buat :



```
#####
####Halaman 56
##
def gabungkanDuaListUrut(A, B):
    la = len(A); lb = len(B)
    C = list()                      #C adalah list baru
    i = 0; j = 0

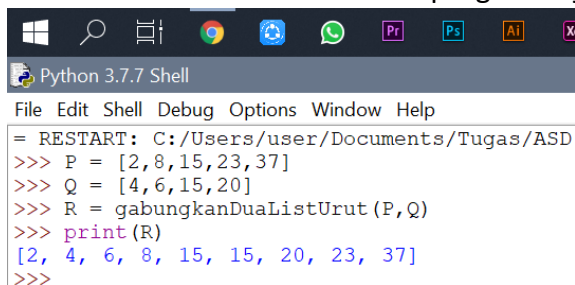
    #Gabungkan keduanya sampai salah satu kosong
    while i < la and j < lb:
        if A[i] < B[j]:
            C.append(A[i])
            i += 1
        else:
            C.append(B[j])
            j += 1

    while i < la:                   #Jika A mempunyai sisa
        C.append(A[i])              #Tumpukkan ke list baru itu
        i += 1                     #satu demi satu

    while j < lb:                   #Jika B mempunyai sisa
        C.append(B[j])              #Tumpukkan ke list baru itu
        j += 1                     #satu demi satu

    return C
```

Berikut adalah screenshoot saat program dijalankan :

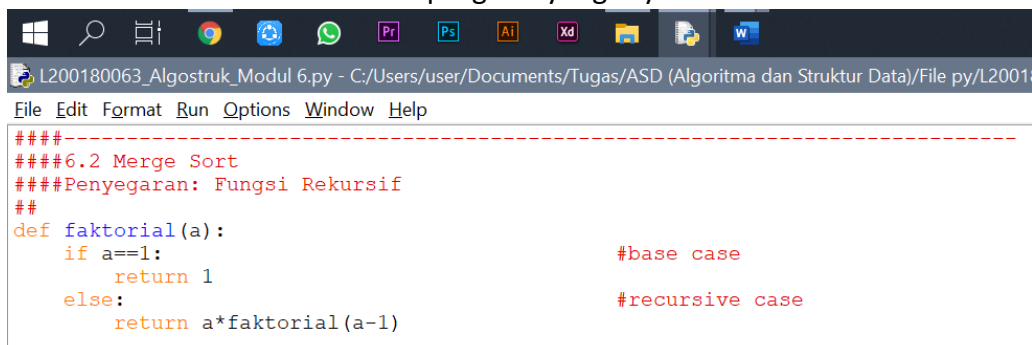


```
Python 3.7.7 Shell
File Edit Shell Debug Options Window Help
= RESTART: C:/Users/user/Documents/Tugas/ASD
>>> P = [2,8,15,23,37]
>>> Q = [4,6,15,20]
>>> R = gabungkanDuaListUrut(P,Q)
>>> print(R)
[2, 4, 6, 8, 15, 15, 20, 23, 37]
>>>
```

## 6.2 Merge Sort

Penyegaran: Fungsi Rekursif

Berikut adalah screenshoot dari program yang saya buat :



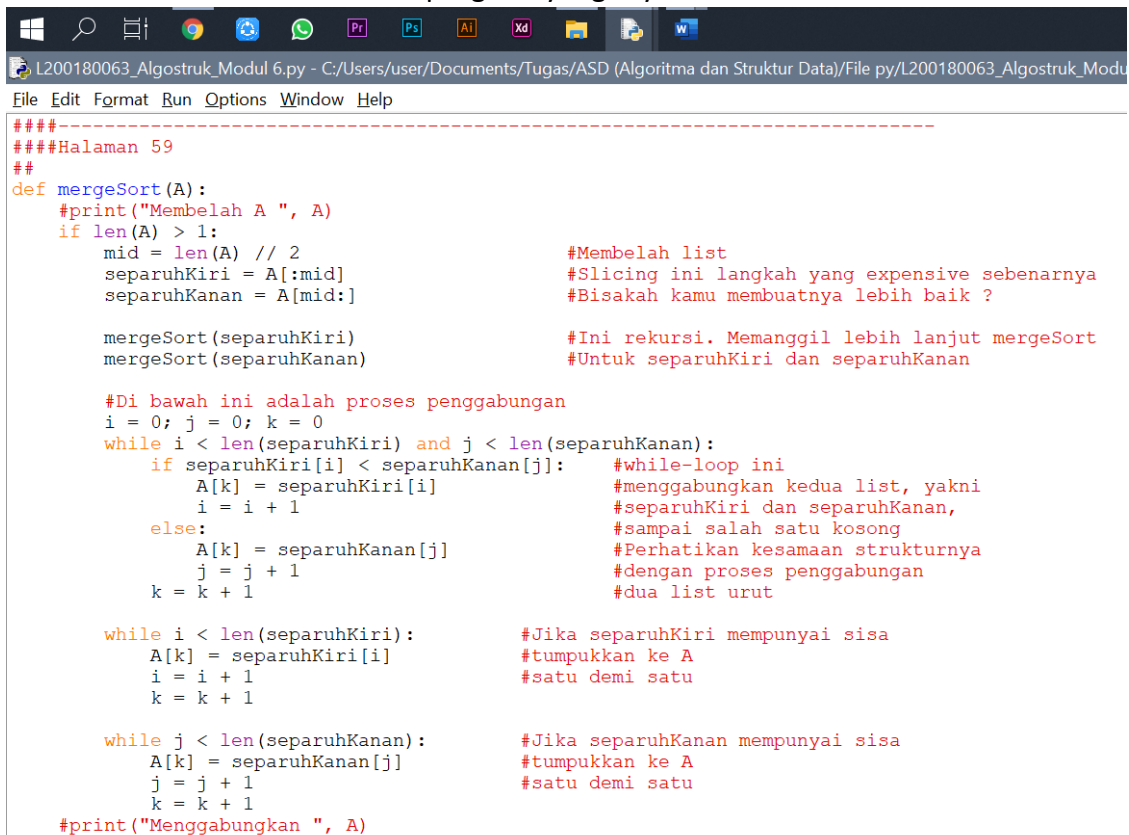
```
#####
####6.2 Merge Sort
####Penyegaran: Fungsi Rekursif
##
def faktorial(a):
    if a==1:                        #base case
        return 1
    else:                           #recursive case
        return a*faktorial(a-1)
```

Berikut adalah screenshoot saat program dijalankan :

```
===== RESTART: C:
>>> faktorial(1)
1
>>> faktorial(7)
5040
>>> faktorial(10)
3628800
>>> faktorial(4)
24
>>>
```

Halaman 59

Berikut adalah screenshoot dari program yang saya buat:



```
L200180063_Algostruk_Modul 6.py - C:/Users/user/Documents/Tugas/ASD (Algoritma dan Struktur Data)/File py/L200180063_Algostruk_Modul
File Edit Format Run Options Window Help
#####
####Halaman 59
##
def mergeSort(A):
    #print("Membelah A ", A)
    if len(A) > 1:
        mid = len(A) // 2
        separuhKiri = A[:mid]
        separuhKanan = A[mid:]

        mergeSort(separuhKiri)
        mergeSort(separuhKanan)

        #Di bawah ini adalah proses penggabungan
        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

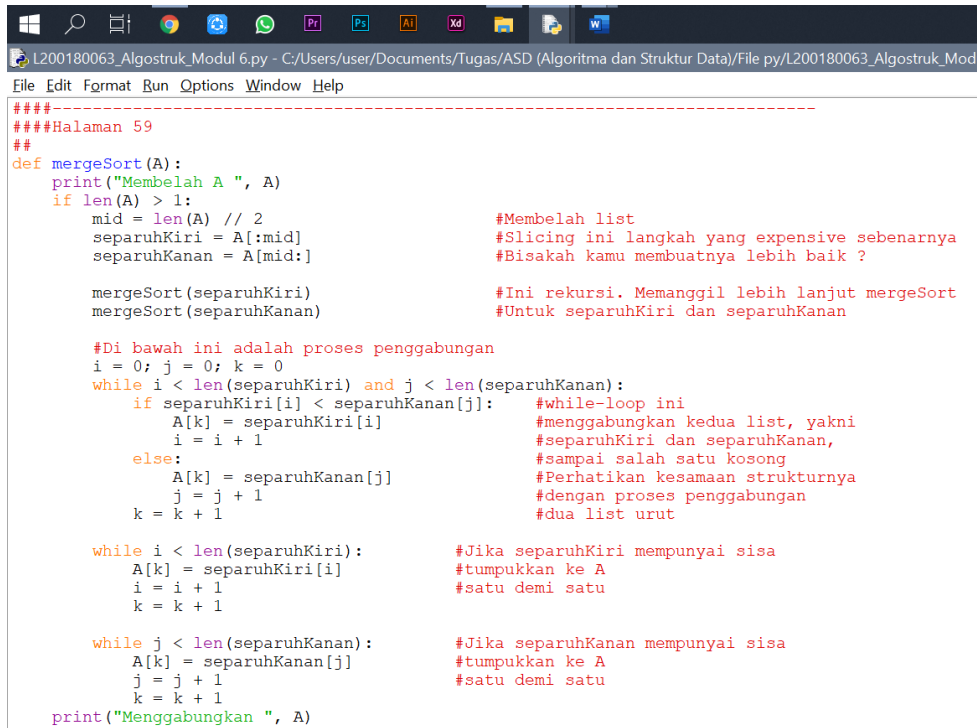
        while j < len(separuhKanan):
            A[k] = separuhKanan[j]
            j = j + 1
            k = k + 1

    #print("Menggabungkan ", A)
```

Berikut adalah screenshoot saat program dijalankan :

```
===== RESTART: C:/Users/u
>>> alist = [54,26,93,17,77,31,44,55,20]
>>> mergeSort(alist)
>>> print(alist)
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>> |
```

Berikut adalah screenshot program saat perintah print("Membelah ", A) dan print("Menggabungkan ", A) sudah diaktifkan :



```
#####
####Halaman 59
##
def mergeSort(A):
    print("Membelah A ", A)
    if len(A) > 1:
        mid = len(A) // 2
        separuhKiri = A[:mid]
        separuhKanan = A[mid:]

        mergeSort(separuhKiri)
        mergeSort(separuhKanan)

    #Di bawah ini adalah proses penggabungan
    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)
```

#####

####Halaman 59

##

def mergeSort(A):

print("Membelah A ", A)

if len(A) > 1:

mid = len(A) // 2

separuhKiri = A[:mid]

separuhKanan = A[mid:]

mergeSort(separuhKiri)

mergeSort(separuhKanan)

#Membelah list

#Slicing ini langkah yang expensive sebenarnya

#Bisakah kamu membuatnya lebih baik ?

#Ini rekursi. Memanggil lebih lanjut mergeSort

#Untuk separuhKiri dan separuhKanan

#Di bawah ini adalah proses penggabungan

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-loop ini

#menggabungkan kedua list, yakni

#separuhKiri dan separuhKanan,

#sampai salah satu kosong

#Perhatikan kesamaan strukturnya

#dengan proses penggabungan

#dua list urut

while i < len(separuhKiri):

A[k] = separuhKiri[i]

i = i + 1

k = k + 1

#Jika separuhKiri mempunyai sisa

#tumpukkan ke A

#satu demi satu

while j < len(separuhKanan):

A[k] = separuhKanan[j]

j = j + 1

k = k + 1

#Jika separuhKanan mempunyai sisa

#tumpukkan ke A

#satu demi satu

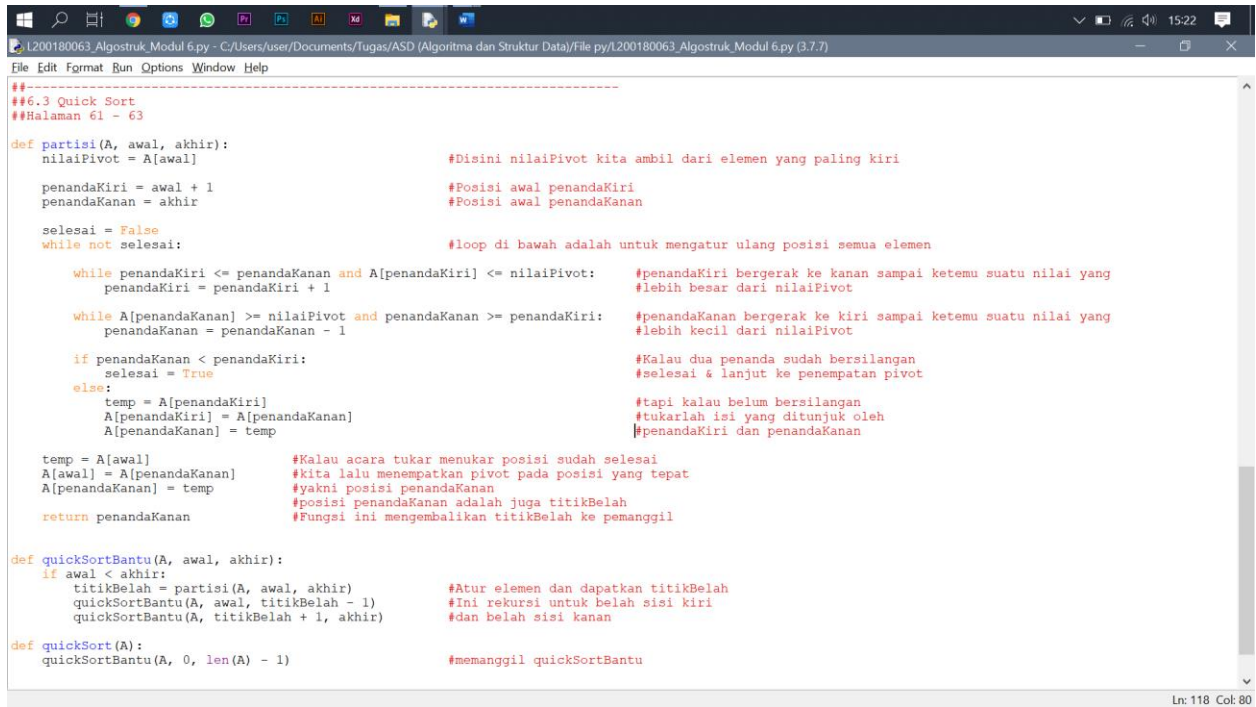
print("Menggabungkan ", A)

Berikut adalah screenshoot saat program dijalankan :

```
===== RESTART: C:/Users/user/Document
>>> alist = [54,26,93,17,77,31,44,55,20]
>>> mergeSort(alist)
Membelah A [54, 26, 93, 17, 77, 31, 44, 55, 20]
Membelah A [54, 26, 93, 17]
Membelah A [54, 26]
Membelah A [54]
Menggabungkan [54]
Membelah A [26]
Menggabungkan [26]
Menggabungkan [26, 54]
Membelah A [93, 17]
Membelah A [93]
Menggabungkan [93]
Membelah A [17]
Menggabungkan [17]
Menggabungkan [17, 93]
Menggabungkan [17, 26, 54, 93]
Membelah A [77, 31, 44, 55, 20]
Membelah A [77, 31]
Membelah A [77]
Menggabungkan [77]
Membelah A [31]
Menggabungkan [31]
Menggabungkan [31, 77]
Membelah A [44, 55, 20]
Membelah A [44]
Menggabungkan [44]
Membelah A [55, 20]
Membelah A [55]
Menggabungkan [55]
Membelah A [20]
Menggabungkan [20]
Menggabungkan [20, 55]
Menggabungkan [20, 44, 55]
Menggabungkan [20, 31, 44, 55, 77]
Menggabungkan [17, 20, 26, 31, 44, 54, 55, 77, 93]
>>> print(alist)
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>>
```

## 6.3 Quick Sort

Berikut adalah screenshoot dari program yang saya buat:



```
##-----
##6.3 Quick Sort
##Halaman 61 - 63

def partisi(A, awal, akhir):
    nilaiPivot = A[awal]                                #Disini nilaiPivot kita ambil dari elemen yang paling kiri
    penandaKiri = awal + 1                               #Posisi awal penandaKiri
    penandaKanan = akhir                                 #Posisi awal penandaKanan
    selesai = False
    while not selesai:                                   #loop di bawah adalah untuk mengatur ulang posisi semua elemen
        while penandaKiri <= penandaKanan and A[penandaKiri] <= nilaiPivot: #penandaKiri bergerak ke kanan sampai ketemu suatu nilai yang
            penandaKiri = penandaKiri + 1                 #lebih besar dari nilaiPivot
        while A[penandaKanan] >= nilaiPivot and penandaKanan >= penandaKiri: #penandaKanan bergerak ke kiri sampai ketemu suatu nilai yang
            penandaKanan = penandaKanan - 1               #lebih kecil dari nilaiPivot
        if penandaKanan < penandaKiri:                   #Kalau dua penanda sudah bersilangan
            selesai = True                                #selesai & lanjut ke penempatan pivot
        else:
            temp = A[penandaKiri]                         #tapi kalau belum bersilangan
            A[penandaKiri] = A[penandaKanan]              #tukarlah isi yang ditunjuk oleh
            A[penandaKanan] = temp                       #penandaKiri dan penandaKanan

    temp = A[awal]                                       #Kalau acara tukar menukar posisi sudah selesai
    A[awal] = A[penandaKanan]                           #kita lalu menempatkan pivot pada posisi yang tepat
    A[penandaKanan] = temp                             #yakni posisi penandaKanan
    #posisi penandaKanan adalah juga titikBelah
    #Fungsi ini mengembalikan titikBelah ke pemanggil
    return penandaKanan

def quickSortBantu(A, awal, akhir):
    if awal < akhir:
        titikBelah = partisi(A, awal, akhir)             #Atur elemen dan dapatkan titikBelah
        quickSortBantu(A, awal, titikBelah - 1)          #Ini rekursi untuk belah sisi kiri
        quickSortBantu(A, titikBelah + 1, akhir)          #dan belah sisi kanan

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

Berikut adalah screenshoot saat program dijalankan :

```
===== RESTART: C:/Users/use
>>> A = [54,26,93,17,77,31,44,55,20]
>>> partisi(A,0,8)
5
>>> quickSort(A)
>>> print(A)
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>> |
```