Selection Sort

- Kombinasi antara searching dan sorting.

1. Mencari elemen dengan nilai terbaik (select).

2. Menempatkan elemen terbaik ke posisi yang seharusnya (tukar dengan elemen yang menempati posisi tersebut saat ini).

Contoh :

```python
#PENGURUTAN SORTING

angka= [9, 5, 8, 6, 7, 4, 3, 1, 2, ]

#===============SELECTION SORT===============#

#MEMBUAT FUNGSI/FUNCTION YANG MENERIMA PARAMETER ARRAY
def selection_sort(array):
    #MEMBUAT FOR PERULANGAN SEBANYAK ISI/VALUE DARI ARRAY
    for i in range(len(array)-1):
        #MEMBUAT VARIABEL INDEX_AWAL DAN MENGISINYA DENGAN I
        min_idx = i
        #MEMBUAT FOR PERULANGAN DI MNA I + 1, DAN LOOP SEBANYAK ISI
        for j in range(i+1, len(array)):
            #JIKA ARRAY[J] LEBIH KECIL DARI ARRAY[INDEX_AWAL]
            if array[j] < array [min_idx]:
                #MAKA INDEX_AWAL = J
                min_idx = j
        #MENGUBAH VALUE
        array[i], array[min_idx] = array[min_idx], array[i]

```

Bubble Sort

- Diilustrasikan seperti pergerakan busa (bubble) dalam air. Busa akan bergerak ke atas karena massa jenisnya lebih ringan dibandingkan air. Membandingkan busa dengan air: yang busa naik, yang air turun.

- Dalam setiap iterasi, elemen dibandingkan dengan elemen sebelahnya: jika posisi elemen yang lebih kecil ada di kanan (di indeks yang lebih besar), maka lakukan pertukaran.

- Di akhir setiap iterasi, posisi elemen terbesar pasti akan ada di sebelah kanan (di indeks yang seharusny

Contoh :

```python
#===============BUBLE SORT===============#

def buble_sort(array):
    for i in range(len(array)-1):
        for j in range(len(array)-1-i):
            if array [j] > array [j+1]:
                array[j], array[j+1] = array[j+1], array[j]

```

Insertion Sort

1. Pengurutan dilakukan dari elemen ke-2 sampai elemen terakhir, dibandingkan dengan elemen-elemen sebelumnya.

2. Jika ditemukan data yang lebih kecil, maka data yang dibandingkan akan ditempatkan di posisi yang seharusnya.

3. Pada penyisipan elemen, elemen-elemen lain bergeser ke belakang.

Contoh :

```
30  #===============INSERTION SORT===============#
31
32  def insertion_sort(array):
33      for i in range(1, len(array)):
34          key = array[i]
35          j = i - 1
36          while j >= 0 and array[j] > key:
37              array[j+1] = array[j]
38              j = j - 1
39              array[j+1] = key
40
```

Merge Sort

1. Menggunakan pendekatan iteratif divide-and-conquer (membagi dan menyelesaikan)
2. Divide: membagi masalah menjadi dua submasalah yang lebih kecil
3. Conquer: menyelesaikan setiap masalah secara rekursif
4. Combine: menggabungkan dua submasalah yang telah diselesaikan

Contoh :

```
41  #===============MERGE SORT===============#
42
43  def merge(left, right, array):
44      i, j, k = 0, 0, 0
45      while i < len(left) and j < len(right):
46          if left[i] < right[j]:
47              array[k] = left[i]
48              i = i + 1
49          else:
50              array[k] = right[j]
51              j = j + 1
52          k = k + 1
53
54      while i < len(left):
55          array[k] = left[i]
56          k = k + 1
57          i = i + 1
58
59      while j < len(right):
60          array[k] = right[j]
61          k = k + 1
62          j = j + 1
63
64  def merge_sort(array):
65      if len(array) > 1:
66          mid = len(array) // 2
67          leftarr = array[:mid]
68          rightarr = array[mid:]
69          merge_sort(leftarr)
70          merge_sort(rightarr)
71          merge(leftarr, rightarr, array)
72
```

Quick Sort

1. Seperti Merge Sort, menggunakan pendekatan divide-and-conquer
2. Membagi kontainer menjadi dua partisi berdasarkan suatu pivot (nilai acuan) yang dipilih dari salah satu elemen pada kontainer yang akan diurutkan
3. Cara kerja partitioning: Elemen yang lebih kecil dari nilai pivot akan ditempatkan di sebelah kiri pivot, sedangkan elemen yang lebih besar dari nilai pivot akan ditempatkan di sebelah kanan pivot.
4. Di setiap akhir partitioning, elemen pivot pasti berada di indeks yang seharusnya
5. Pemilihan pivot: elemen pertama, elemen terakhir, elemen tengah, random

Contoh :

```
73  #===============QUICK SORT===============#
74
75  def quick_sort(array):
76      quick_sort_rec(array,0, len(array)-1)
77
78  def quick_sort_rec(array, start, end):
79      if start< end:
80          pivot_idx = partition(array, start, end)
81          quick_sort_rec(array, start, pivot_idx-1)
82          quick_sort_rec(array, pivot_idx+1, end)
83  def partition(array, start, end):
84      pivot= array[end]
85      i = start
86      for j in range(start, end):
87          if array[j] <= pivot:
88              array[i], array[j]= array[j], array[i]
89              i = i + 1
90      array[i], array[end]= array[end], array[i]
91      return i
92
```