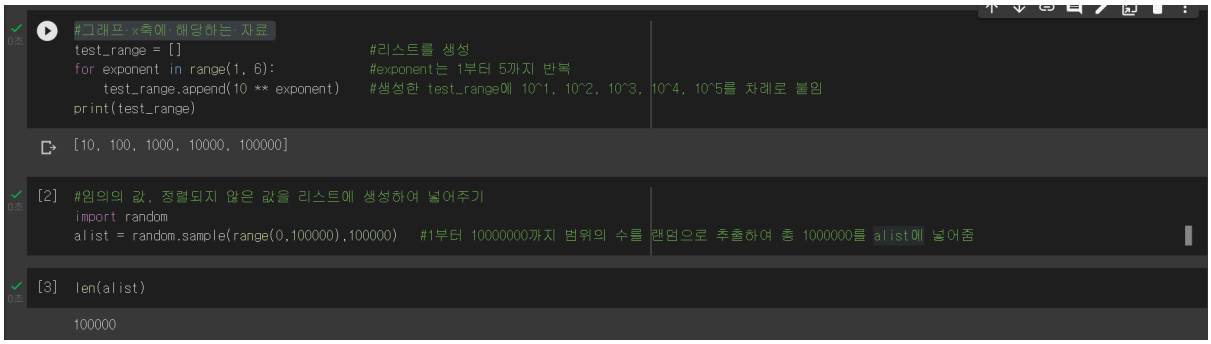
12180626 성시열 산업정보화 002분반 중간고사 대체과제

[ 코드 및 생성과정 ]



```
#그래프 x축에 해당하는 자료
test_range = []                         #리스트를 생성
for exponent in range(1, 6):            #exponent는 1부터 5까지 반복
    test_range.append(10 ** exponent)   #생성한 test_range에 10^1, 10^2, 10^3, 10^4, 10^5를 차례로 붙임
print(test_range)
```

```
[10, 100, 1000, 10000, 100000]
```

```
[2] #임의의 값, 정렬되지 않은 값을 리스트에 생성하여 넣어주기
    import random
    alist = random.sample(range(0,100000),100000)   #1부터 10000000까지 범위의 수를 랜덤으로 추출하여 총 1000000를 alist에 넣어줌
```

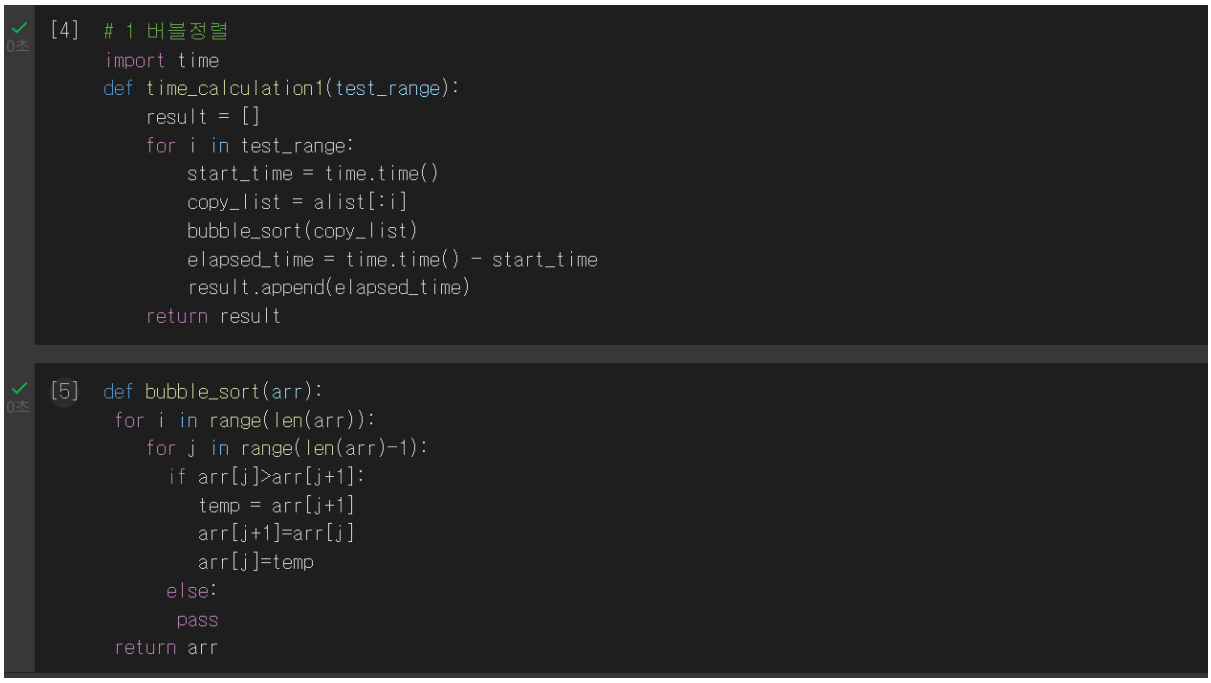
```
[3] len(alist)

100000
```

- 그래프로 시각화할 때 x축에 해당하는 값 10, 100, 1000, 10000, 100000값을 생성하고 출력해보며 결과를 확인.

- 이후 정렬되지 않은 값으로 구성된 리스트를 만들기 위해 1부터 100,000사이 범위의 임의의 값 100,000개를 alist에 생성.

- alist의 길이를 확인.

1. 버블정렬



```
[4] # 1 버블정렬
    import time
    def time_calculation1(test_range):
        result = []
        for i in test_range:
            start_time = time.time()
            copy_list = alist[:i]
            bubble_sort(copy_list)
            elapsed_time = time.time() - start_time
            result.append(elapsed_time)
        return result
```

```
[5] def bubble_sort(arr):
     for i in range(len(arr)):
        for j in range(len(arr)-1):
          if arr[j]>arr[j+1]:
             temp = arr[j+1]
             arr[j+1]=arr[j]
             arr[j]=temp
          else:
           pass
     return arr
```

- 버블정렬 알고리즘 구현 및 버블정렬이 실행되는 시간을 계산하는 함수 생성.

```
[6]  output_bubble = time_calculation1(test_range)
```

29분

```
[7]  output_bubble
```

0초

```
[3.552436828613281e-05,
 0.008541584014892578,
 0.504798173904419,
 21.01521611213684,
 1762.1225230693817]
```

- 버블 정렬이 실행되는 시간이 계산된 값을 output_bubble에 저장.

- output_bubble 값 확인.

2. 선택정렬

```
[8]  #2 선택정렬
     def time_calculation2(test_range):
         result = []
         for i in test_range:
             start_time = time.time()
             copy_list = alist[:i]
             selection_sort(copy_list)
             elapsed_time = time.time() - start_time
             result.append(elapsed_time)
         return result
```

0초

```
[9]  def find_smallest(arr):
         smallest = arr[0]
         smallest_index = 0
         for i in range(1,len(arr)):
             if arr[i] < smallest:
                 smallest = arr[i]
                 smallest_index = i
         return smallest_index
```

0초

```
[10] def selection_sort(arr):
         newArr = []
         for i in range(len(arr)):
             smallest = find_smallest(arr)
             newArr.append(arr.pop(smallest))
         return newArr
```

- 선택정렬 알고리즘 구현 및 선택정렬이 실행되는 시간을 계산하는 함수 생성.

```
[ ]  output_selection = time_calculation2(test_range)
```

```
[ ]  output_selection
```

```
[2.8133392333984375e-05,
 0.0009121894836425781,
 0.03536868095397949,
 3.7280654907226562,
 456.4719979763031]
```

- 선택 정렬이 실행되는 시간이 계산된 값을 output_selection에 저장.

- output_selection 값 확인.

3. 합병정렬

```
[ ]  #3 합병정렬
     import time

     def time_calculation3(test_range):
         result = []
         for i in test_range:
             start_time = time.time()
             copy_list = alist[:i]
             merge_sort(copy_list)
             elapsed_time = time.time() - start_time
             result.append(elapsed_time)
         return result
```

```
[ ]  def merge_sort(list):
         if len(list) <= 1:
             return list

         mid = len(list) // 2
         leftList = list[:mid]
         rightList = list[mid:]

         leftList = merge_sort(leftList)
         rightList = merge_sort(rightList)
         return merge(leftList, rightList)
```

```
[ ]  def merge(left, right):
         result = []
         while len(left) > 0 or len(right) > 0:
             if len(left) > 0 and len(right) > 0:
                 if left[0] <= right[0]:
                     result.append(left[0])
                     left = left[1:]
                 else:
                     result.append(right[0])
                     right = right[1:]
             elif len(left) > 0:
                 result.append(left[0])
                 left = left[1:]
             elif len(right) > 0:
                 result.append(right[0])
                 right = right[1:]
         return result
```

- 합병정렬 알고리즘 구현 및 합병정렬이 실행되는 시간을 계산하는 함수 생성.

```
[ ]  output_merge = time_calculation3(test_range)
```

```
[ ]  output_merge
```

```
[6.151199340820312e-05,
 0.0012385845184326172,
 0.010023355484008789,
 0.36121201515197754,
 39.017786741256714]
```

- 합병 정렬이 실행되는 시간이 계산된 값을 output_merge에 저장.

- output_merge 값 확인.

4. 퀵정렬

```
[ ]  # 4 퀵정렬
     def time_calculation4(test_range):
         result = []
         for i in test_range:
             start_time = time.time()
             copy_list = alist[:i]
             quicksort(copy_list)
             elapsed_time = time.time() - start_time
             result.append(elapsed_time)
         return result
```

```
[ ]  def quicksort(array):
         if len(array) < 2:
             return array

         else:
             pivot = array[0]

             less = []
             greater = []
             for i in array[1:]:
                 if i <= pivot:
                     less.append(i)
                 else:
                     greater.append(i)

             return quicksort(less) + [pivot] + quicksort(greater)
```

- 퀵정렬 알고리즘 구현 및 퀵정렬이 실행되는 시간을 계산하는 함수 생성.

```
[ ]  output_quick = time_calculation4(test_range)

[ ]  output_quick

     [2.4557113647460938e-05,
      0.0002641677856445312,
      0.004910707473754883,
      0.03719735145568848,
      0.3243069648742676]
```

- 퀵 정렬이 실행되는 시간이 계산된 값을 output_quick에 저장.

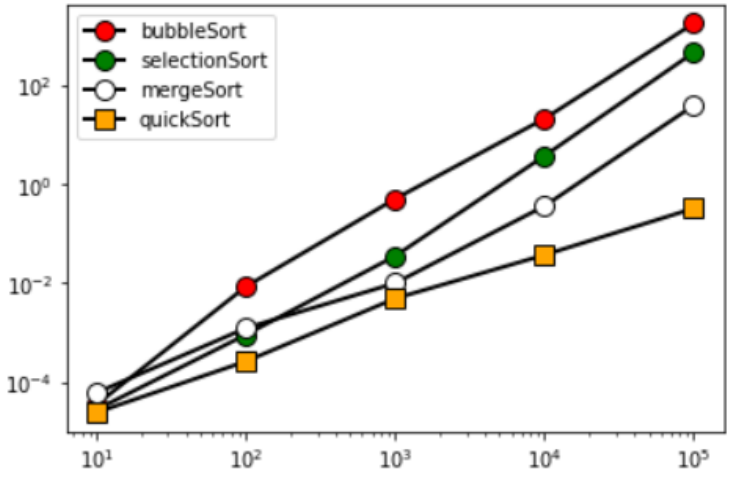
- output_quick 값 확인.

시각화

```
[ ]  # libraries
     import matplotlib.pyplot as plt
     import numpy as np
     import pandas as pd

[ ]  #data
     df=pd.DataFrame({'x': test_range, 'bubbleSort': output_bubble, 'selectionSort': output_selection, 'mergeSort': output_merge, 'quickSort': output_quick})
```

- 시각화를 위해 3가지 라이브러리를 가져오고 df에 데이터를 저장.

```
# 4가지 정렬시간 비교 그래프
plt.xscale('log')
plt.yscale('log')

plt.plot( 'x', 'bubbleSort', data=df, marker="o", markerfacecolor='red', markersize=10, color='black', linewidth=2)
plt.plot( 'x', 'selectionSort', data=df, marker='o', markerfacecolor='green', markersize=10, color='black', linewidth=2)
plt.plot( 'x', 'mergeSort', data=df, marker="o", markerfacecolor='white', markersize=10, color='black', linewidth=2)
plt.plot( 'x', 'quickSort', data=df, marker="s", markerfacecolor='orange', markersize=10, color='black', linewidth=2)

plt.legend()
```




- 4가지 정렬의 시간을 비교하는 그래프 시각화.