퀵정렬(quick sort)

1. 퀵 정렬 (quick sort)이란?

- 정렬 알고리즘의 꽃
- 기준점(pivot이라고 부름)을 정해서, 기준점보다 작은 데이터는 왼쪽(left), 큰 데이터는 오른쪽(right)으로 모으는 함수를 작성함
- 각 왼쪽(left), 오른쪽(right)은 재귀용법을 사용해서 다시 동일 함수를 호출하여 위 작업을 반복함
- 함수는 왼쪽(left) + 기준점(pivot) + 오른쪽(right)을 리턴함

2. 알고리즘 구현

- quicksort 함수 만들기
  - 만약 리스트 개수가 한개이면 해당 리스트 리턴
  - 그렇지 않으면, 리스트 맨 앞의 데이터를 기준점(pivot)으로 놓기
  - left, right리스트 변수를 만들고,
  - 맨 앞의 데이터를 뺀 나머지 데이터를 기준점과 비교(pivot)
    - 기준점보다 작으면 left.append(해당 데이터)
    - 기준점보다 크면 right.append(해당 데이터)
  - return quicksort(left) + pivot + quicksort(right)로 재귀 호출

  >> 리스트로 만들어서 리턴하기 : return quick_sort(left) +[pivot] + quick_sort(right)

```
def qsort(data):
  if len(data)<=1:
    return data
  left, right = list(),list()
  pivot = data[0]

  for index in range(1,len(data)):
    if pivot > data[index]:
```

```
      left.append(data[index])
    else:
      right.append(data[index])
  return qsort(left) + [pivot] + qsort(right)
```

```
imort random

data_list = random.sample(range(100),10)
qsort(data_list)
>>[11, 16, 20, 32, 35, 37, 44, 68, 70, 75]
```

```
def qsort(data):
  if len(data)<=1:
    return data
  pivot = data[0]

  left = [item for item in data[1:] if pivot> item]
  right = [item for item in data[1:] if pivot <=item]

  return qsort(left) +[pivot] + qsort(right)
```

```
import random

data_list = random.sample(range(100),10)

qsort(data_list)
>>[5, 7, 9, 62, 72, 84, 85, 89, 90, 91]
```

3. 알고리즘 분석

- 병합정렬과 유사, 시간복잡도는 O(nlogn)
  - 단, 최악의 경우
    - 맨 처음 pivot이 가장 크거나, 가장 작으면
    - 모든 데이터를 비교하는 상황이 나옴
    - O(n^2)

Worst Case

아래의 배열을 가장 왼쪽 값을 피벗으로 하여 오름차순 정렬해보자