삽입 정렬(insertion sort)

1. 삽입 정렬(insertion sort)란?

- 삽입 정렬은 두 번째 인덱스부터 시작
- 해당 인덱스(key값)앞에 있는 데이터(B)부터 비교해서 key값이 더 작으면, B값을 뒤 인덱스로 복사
- 이를 key값이 더 큰 데이터를 만날때까지 반복, 그리고 큰 데이터를 만난 위치 바로 뒤에 key값을 이동

2. 어떻게 코드로 만들까?

- 데이터가 네 개 일때(데이터 개수에 따라 복잡도가 떨어지는 것은 아니므로, 네 개로 바로 로직을 이해해보자.)
  - 예 : data_list = [9,3,2,5]
    - 처음 한 번 실행하면, key값은 9, index(0)-1은 0보다 작으므로 끝 : [9,3,2,5]
    - 두 번째 실행하면, key값은 3, 9보다 3이 작으므로 자리 바꾸고, 끝 : [3,9,2,5]
    - 세 번째 실행하면, key값은 2, 9보다 2가 작으므로 자리 바꾸고, 다시 3보다 2가 작으므로 끝 : [2,3,9,5]
    - 네 번째 실행하면, key값은 5, 9보다 5가 작으므로 자리 바꾸고, 3보다는 5가 크므로 끝 : [2,3,5,9]

3. 알고리즘 구현

1. for stand in range(len(data_list))로 반복
2. key = data_list[stand]
3. for num in range(stand,0,-1)반복
   a. 내부 반복문안에서 data_lisdt[stand]<data_list[num-1]이면,
      i. swap

```
def insertion_sort(data):
  for index in range(len(data)-1):
    for index2 in range(index+1, 0, -1):
      if data[index2]<data[index2-1]:
        data[index2],data[index2-1] = data[index2-1],data[index2]
      else:
        break
  return data
```

```
import random

data_list = random.sample(range(100),50)
print(insertion_sort(data_list))
>>[1, 2, 3, 5, 8, 9, 10, 11, 14, 16, 17, 20, 22, 23, 32, 33, 34, 36, 40, 43, 46, 47, 49, 50, 51, 53, 56, 57, 60, 61, 62, 64, 65, 67, 6
```

---

4. 알고리즘 분석

- 반복문이 두 개 O(n^2)
  - 최악의 경우, n*(n-1)/2
- 완전 정렬이 되어 있는 상태라면 최선은 O(n)