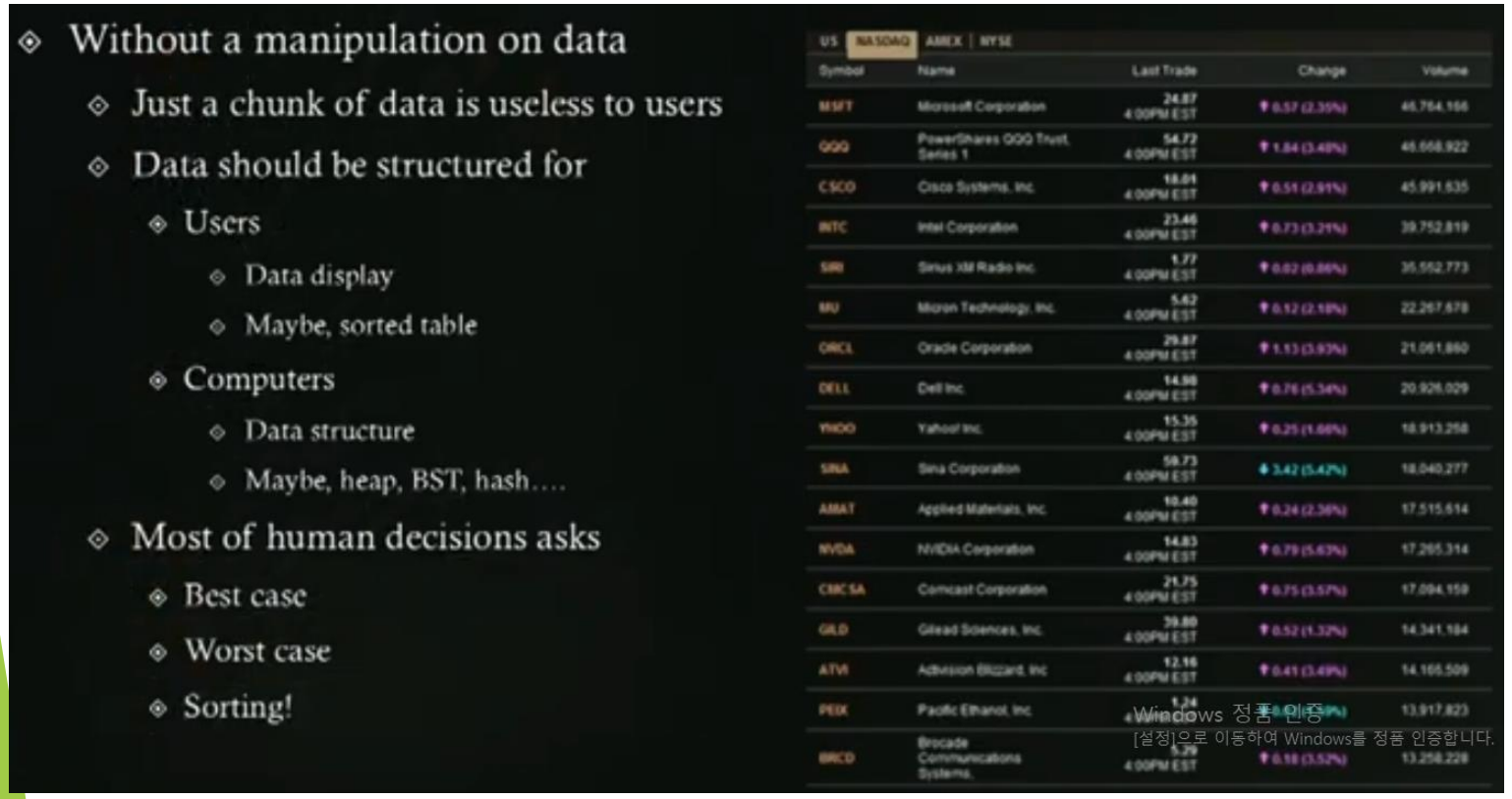
3. Data Structure and Algorithms 2

Sorting

목차

1. O(N^2) Sorting

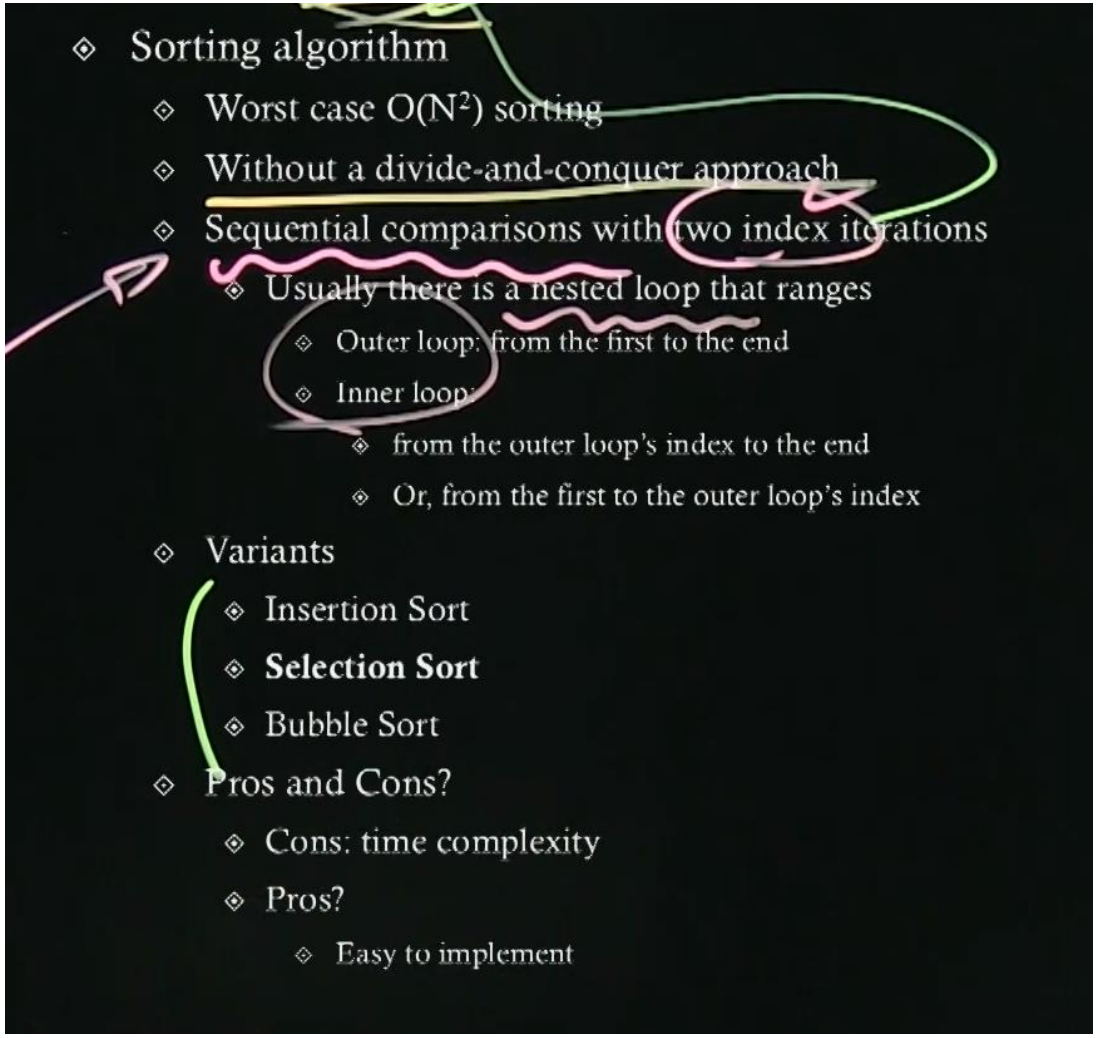
Sorting



- Sorting Algorithms Animations : https://www.toptal.com/developers/sorting-algorithms
- Sorting은 다양한 용도로 쓰인다.
- 이름별, 주식상황별 등
- 다양한 상황에 맞춰 sort algorithms 구현
- 문제는 수십만개를 수십만명이 동시에 들어와서 알아보고자 할 때 핸들링이 어렵다
- DB의 sort를 사용해도 된다. 그 이상의 sort algorithm을 구현하기 힘들다.
- 하지만 DB아니더라고 요소요소에 sort algorithm이 쓰일때가 많다.

1. O(N^2) Sorting

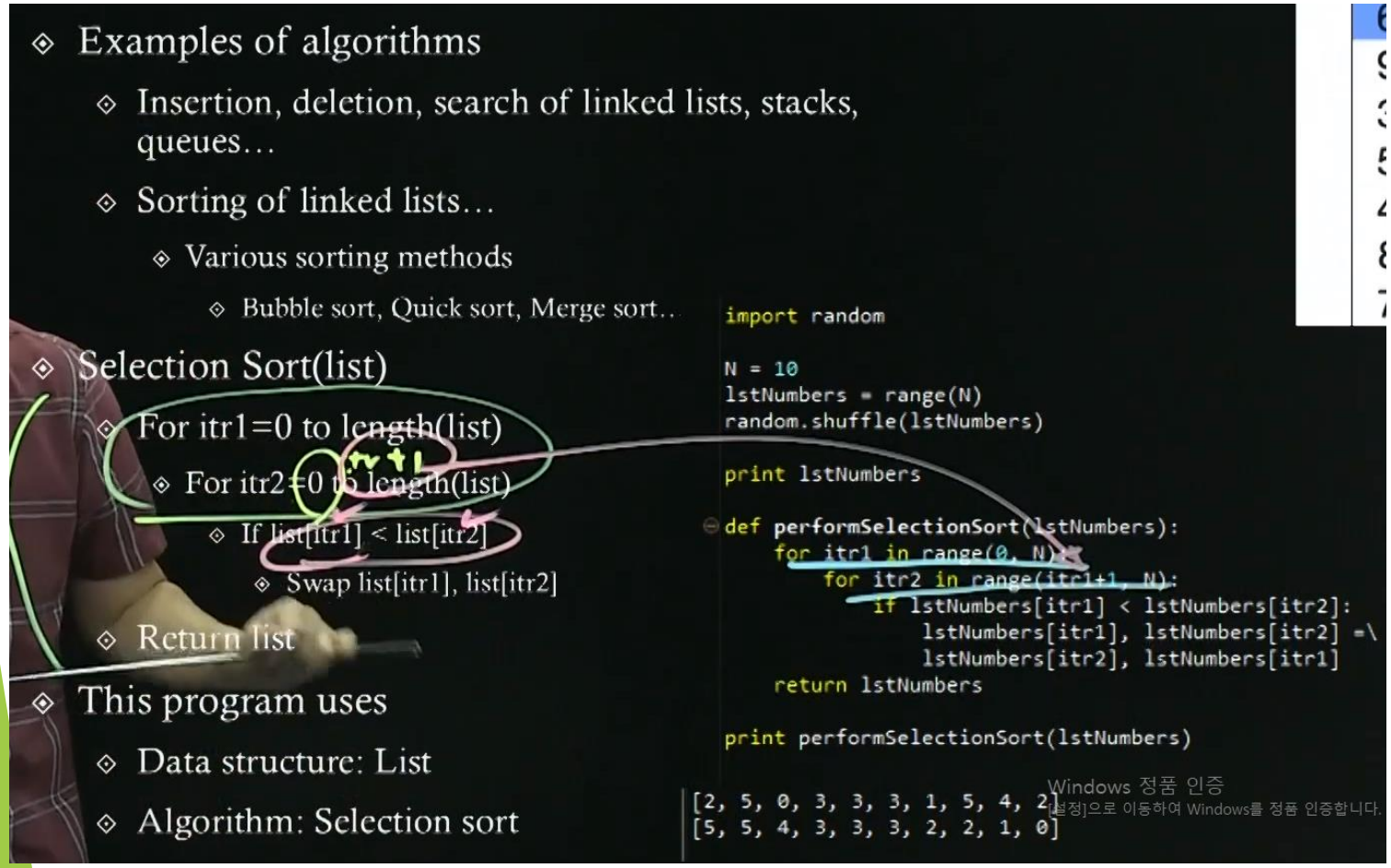
Sorting



- Order of N sqaure은 selection sorting 계열
- 기본적으로 효율적이고 쉽게 구현 가능
- 왜냐하면 tree structure같은 구조가 활용되지 않음
- Index를 두개를 돌리고 sequential comparison을 함
- Time complexity 혼자서 돌려보기에는 좋으나 논문을 쓰기위해 분석을 한다든가 현실에 적용하기에는 좋지 않다.
- 하지만 쉽게 구현이 가능

1. O(N^2) Sorting

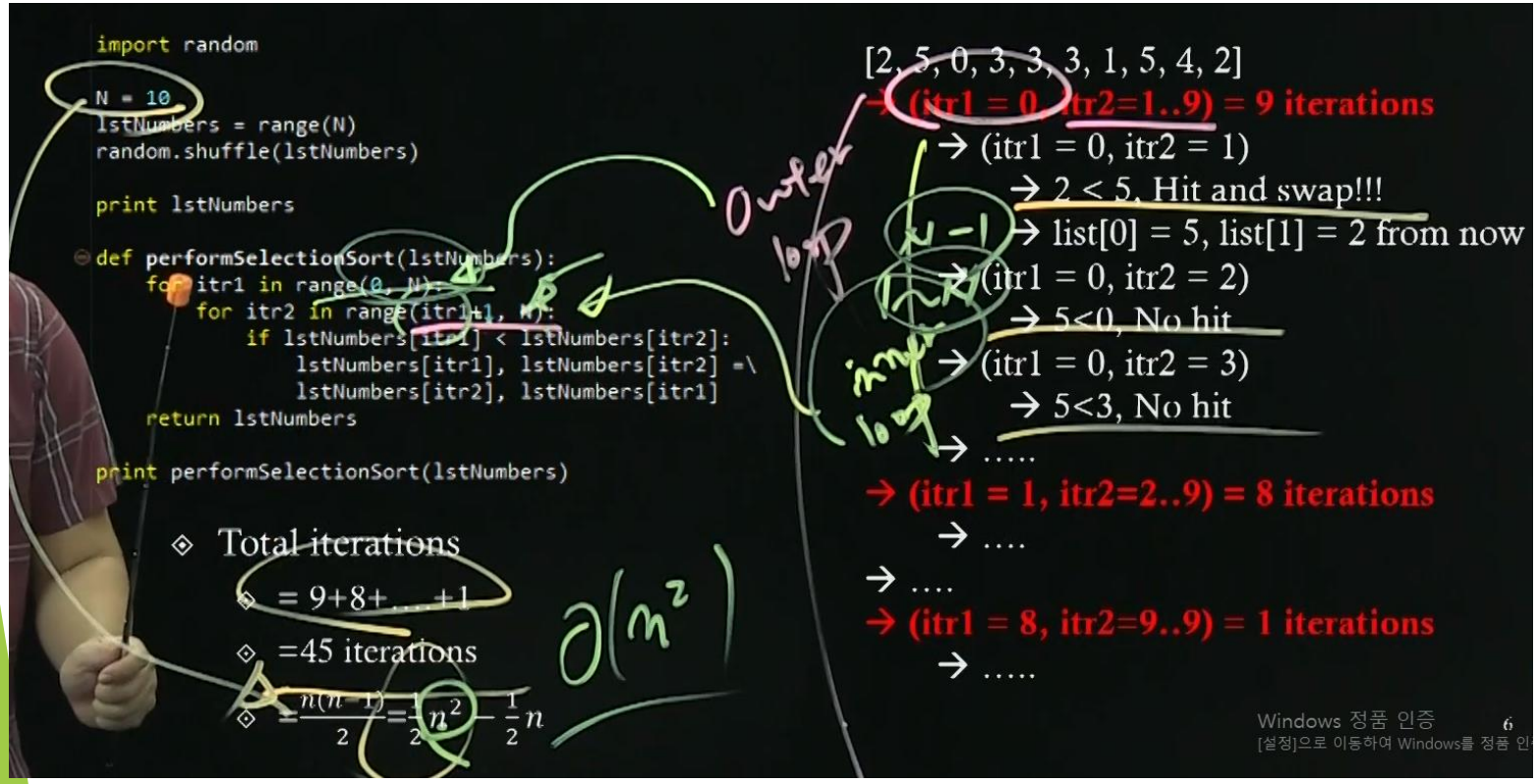
Selection sort algorithm



- Itr2 = 0 보다 효율적으로 하기 위해서 itr2 = itr + 1
- Maxmum이 항상 앞으로 가는 algorithm
- List 뿐만 아니라 array 등 다양하게 사용할수 있다.

1. O(N^2) Sorting

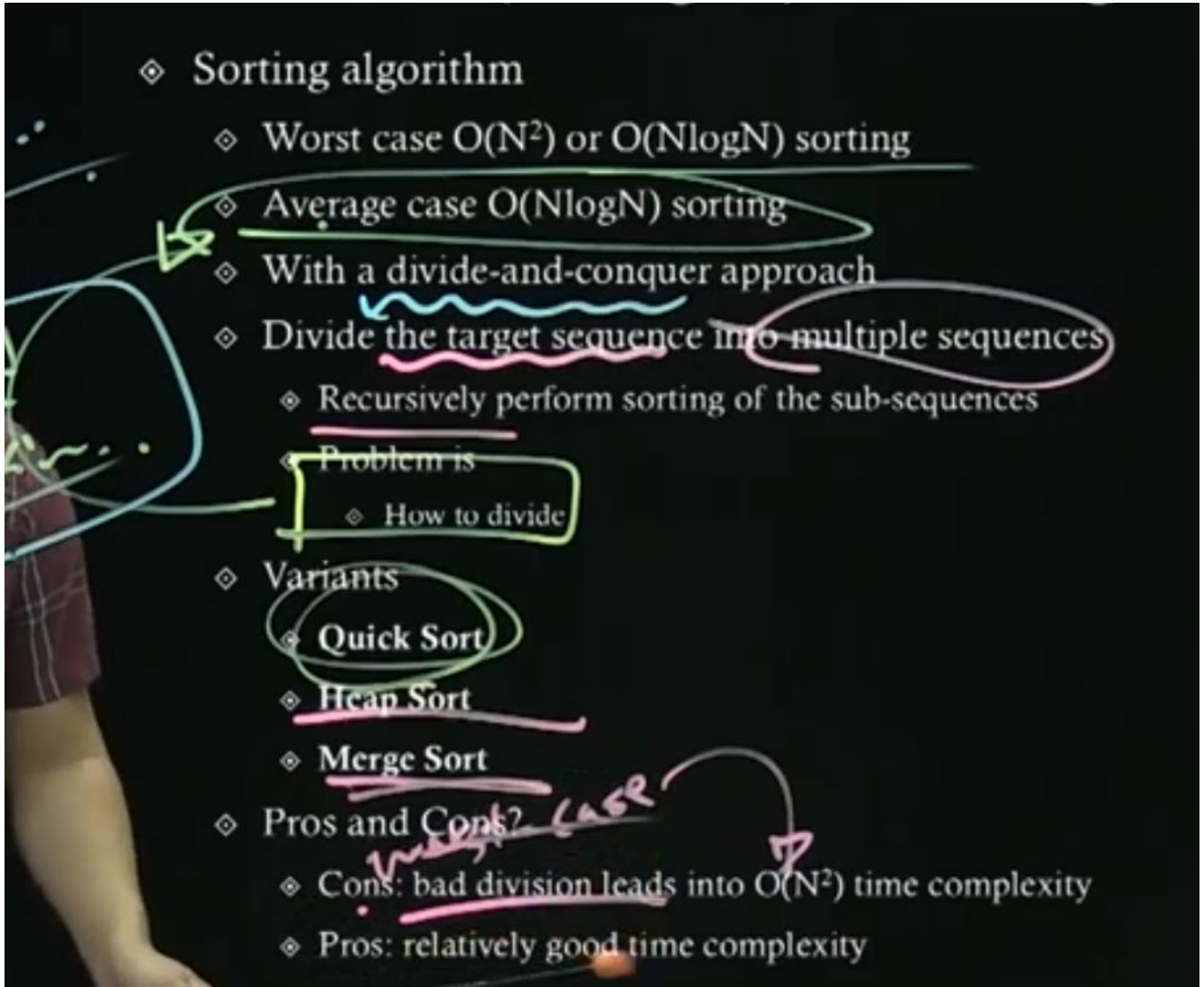
Example of selection sort execution



- irt1 = 0, itr2 = 1..9
- Outer loop는 N번만큼
- 코드 구현은 쉽지만 성능이 너무 안나온다.
- 총 반복이 45번

2. Merge Sort

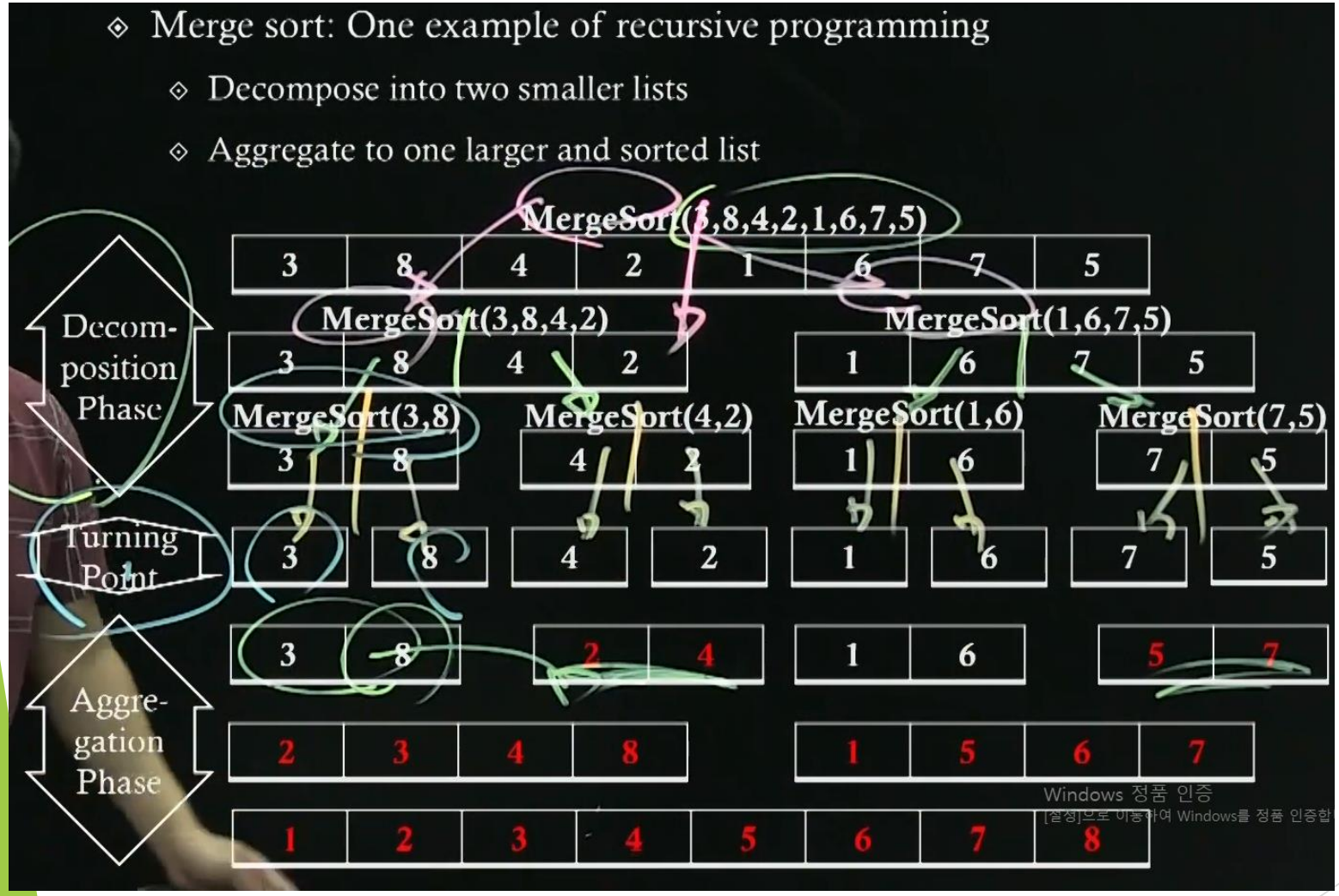
O(NlogN) Sorting



- O(N^2)로직보다 좀 더 나은 O(NlogN)이다
- 이 로직은 divide and conquer를 사용
- 기본적으로 Target sequence를 multiple sequences로 쪼개서 비교하는것
- Recursion이 필요
- Good division이면 average case를 달성
- 단점은 worstcast면 O(N^2)같은 성능을 낸다.
- Comparison에서는 O(NlogN)이 베스트이다.

2. Merge Sort

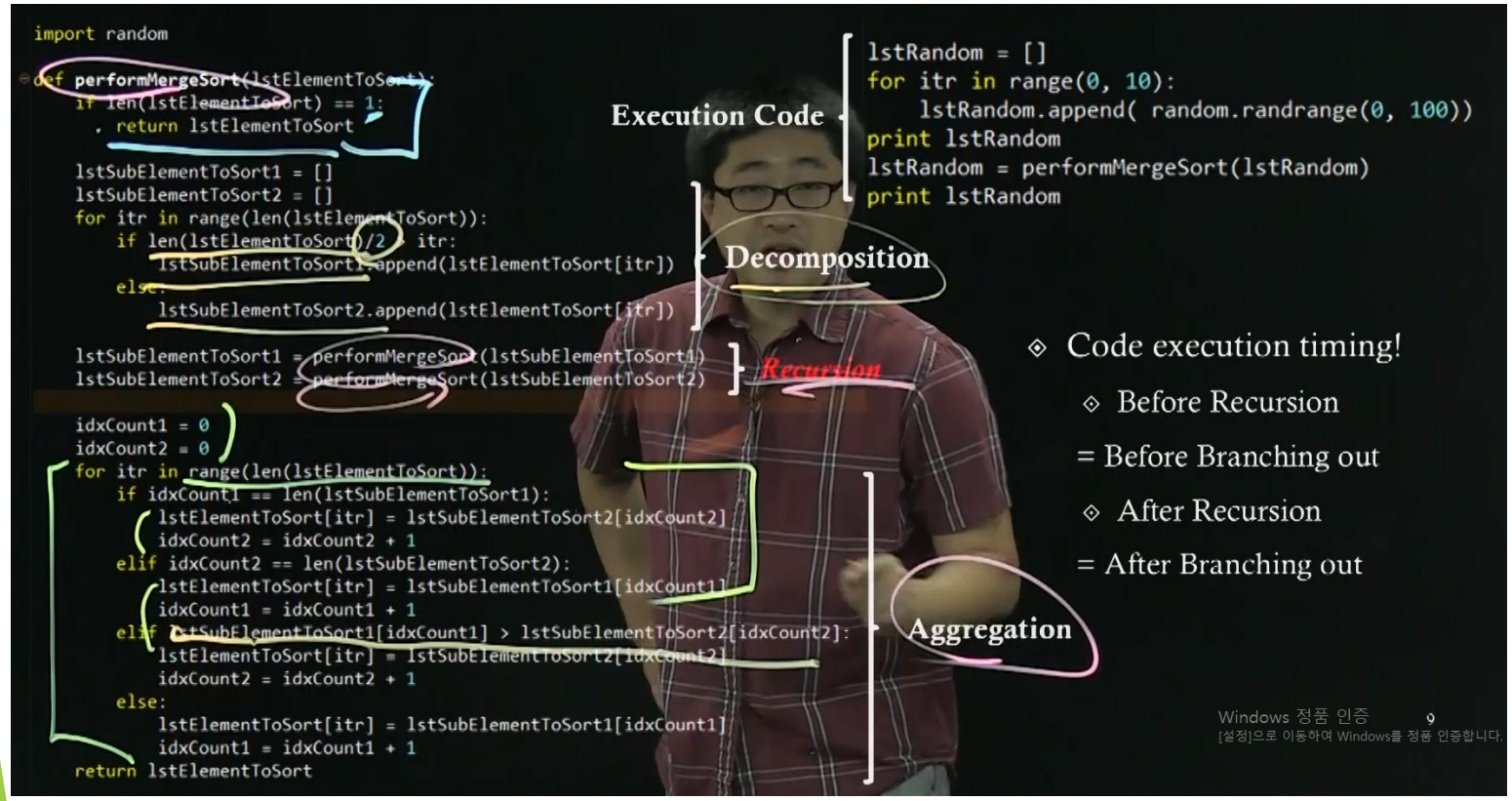
Merge Sort



- Merge Sort는 쪼갤수 없을 때까지 쪼개고 다시 aggregation하는 과정이다.
- Decom-position phase하고
- Recursion escape 으로 탈출한다.
- Sorting한다(작은것을 앞에 쓰거나 큰것을 앞에 쓴다)
- 마지막은 comparison이 필요 없다.

2. Merge Sort

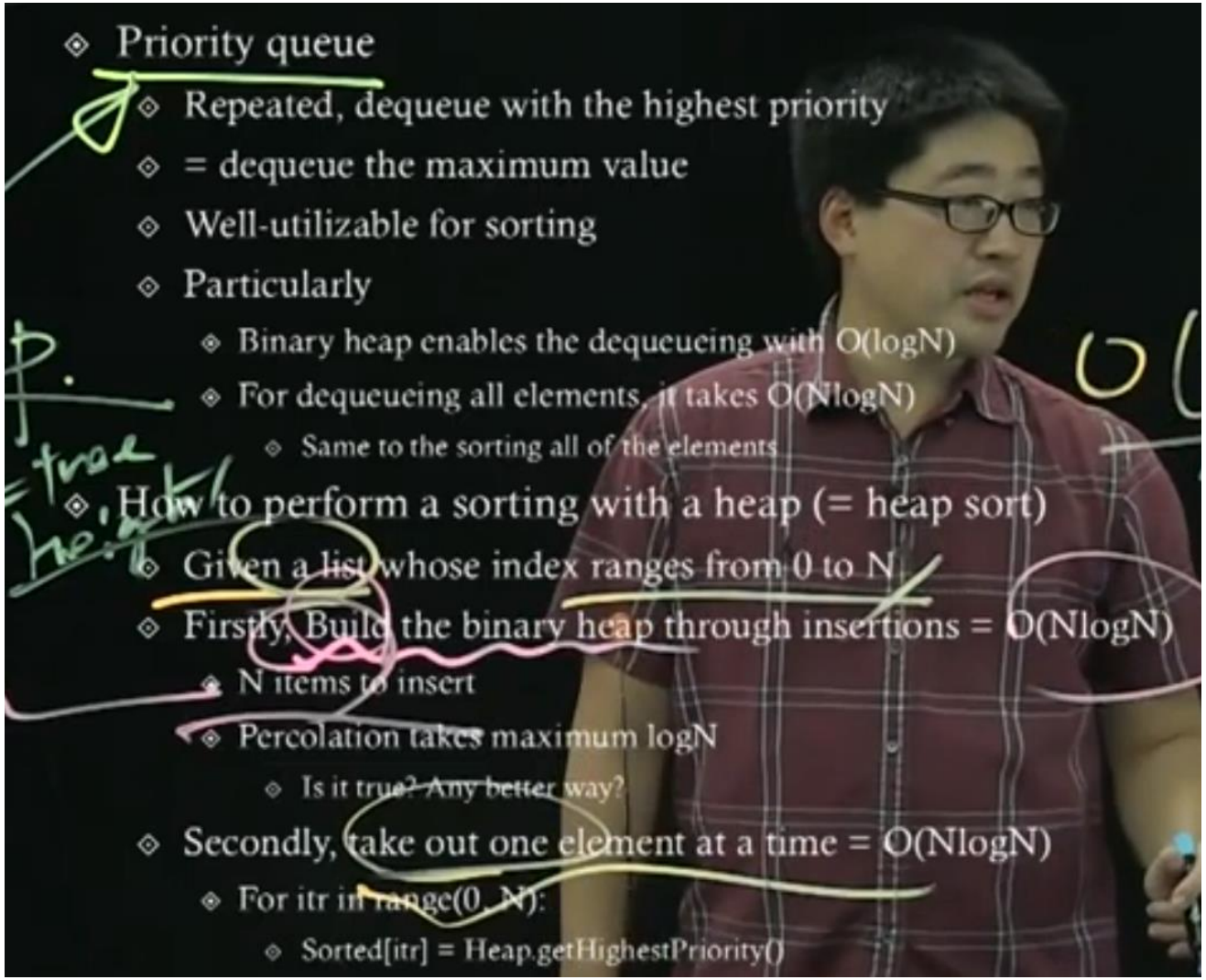
Implementation Example: Merge Sort



- Decomposition :
- 절반으로 쪼개고 append하게 된다.
- Recursion :
- Recursive call 한다.
- Escape:
- 더 이상 쪼갤수 없으면 return한다.
- Aggregation :
- For loop 돌린다.
- 각 index의 0째부터 시작
- Elif 두번째가 비교문
- If 문 첫번째 리스트가 소진되었을때
- Elif 첫번째가 두번째 리스트가 소진 되었을때

3. Heap Sort

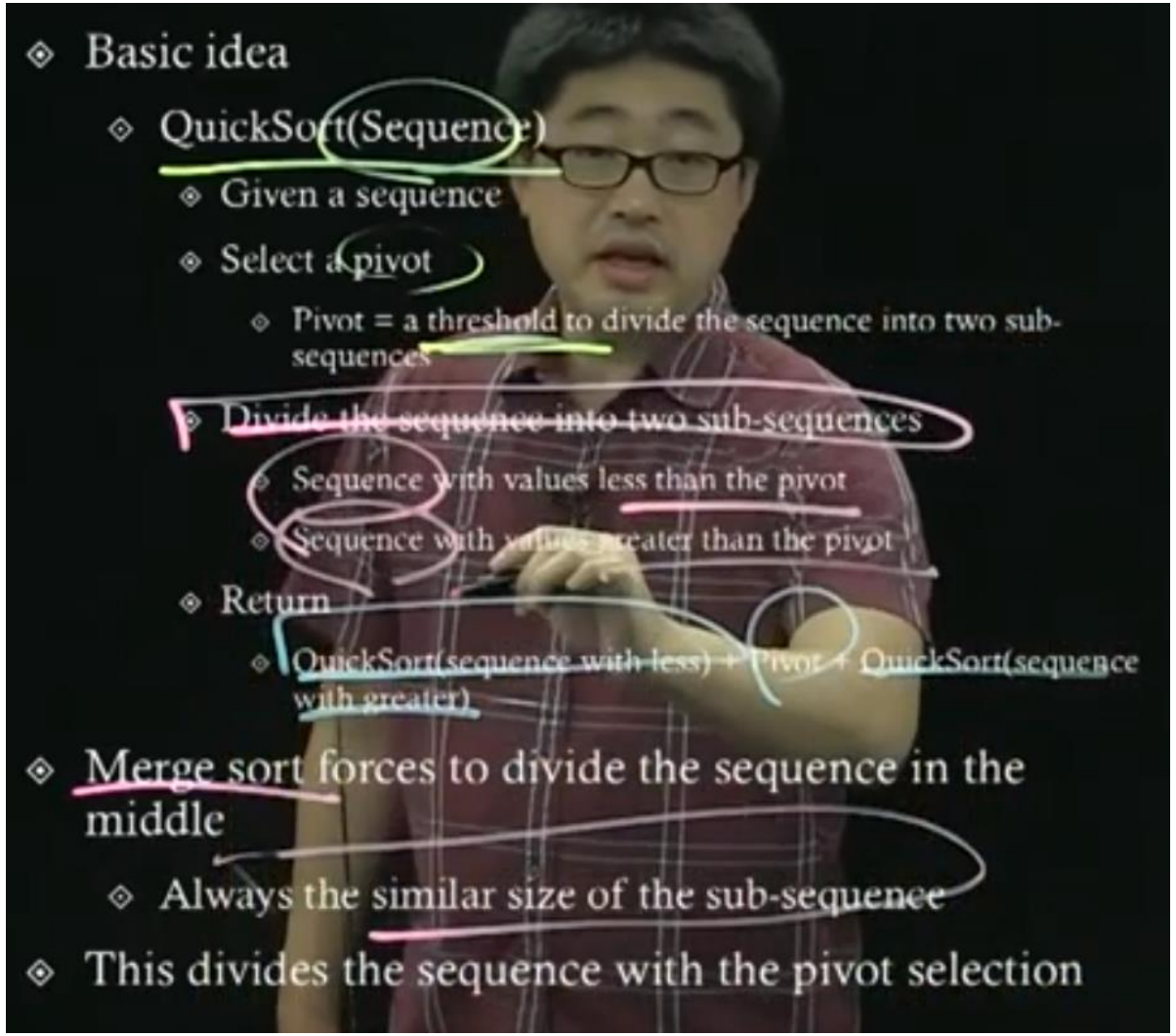
Heap sort



- Heap sort는 Merge sort와 동일한 성능을 내지만 structure만 좀 다르다.
- Priority queue를 활용
- Binary Heap 구조를 활용해서 Heap sort라 함
- Insert를 함으로써 binary heap을 build한다.
- Build하는 과정이 사실상 enqueue하는 것
- 이 때 percorlation-up이 일어난다.
- Build가 되면 하나씩 빼면 된다.
- 빼는 과정은 percorlation-down이 일어난다.

4. Quick Sort

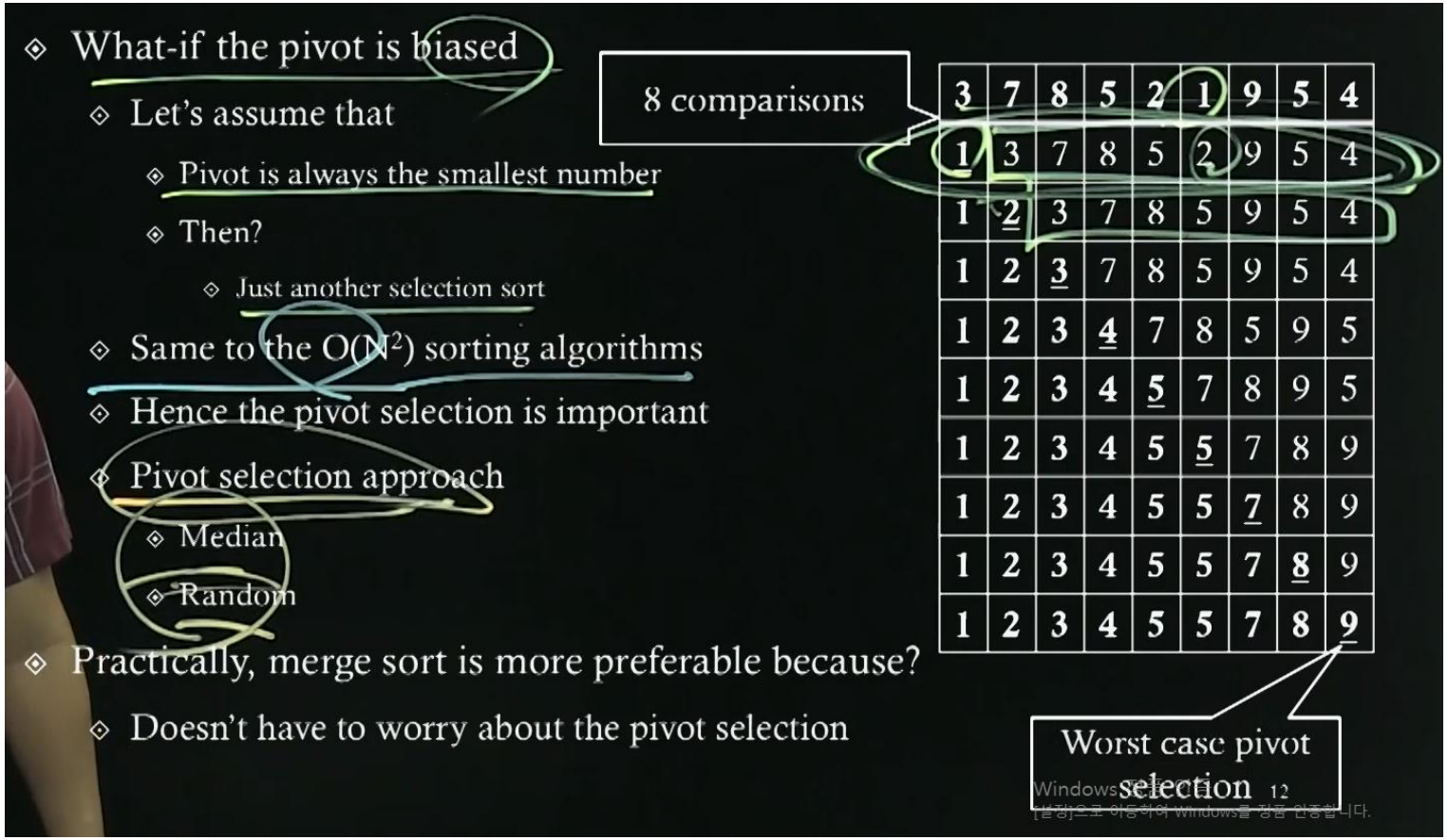
Quick sort



- Quick sort의 기본 매커니즘은 sequence가 들어오면 pivot을 선택함
- Pivot은 왼쪽과 오른쪽을 분리하는 기준점
- Threshold를 넘으면 오른쪽, 작으면 왼쪽
- Sequence을 split하는것
- 그 다음 recursion을 하게 됨
- Pivot은 중간 값을 잘 설정하는게 중요
- Merge sort는 항상 중간
- Pivot이 잘못 설정되면 양쪽의 sequence의 길이차이가 많이 나게 된다.

4. Quick Sort

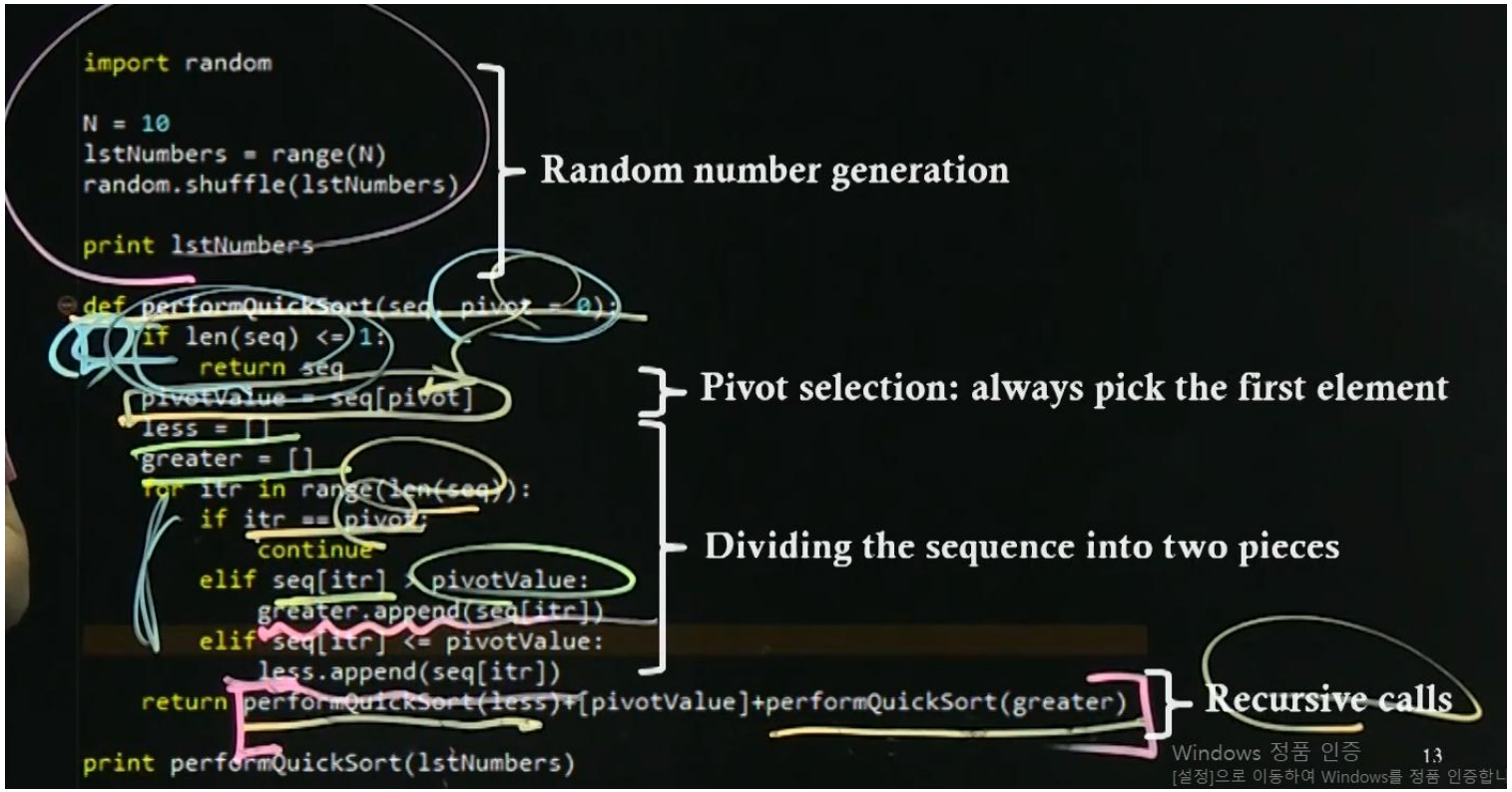
Importance of pivot in quick sort



- Pivot을 가장 작은 숫자를 선택한다면?
- Selection sort랑 동일하게 됨
- Selection의 inner와 outer loop와 같음
- O(N^2) sorting algorithms랑 같음
- 이럴경우 quick sort를 안하는게 낫다
- Median과 Random의 접근 방식이 있다.
- Merge sort가 기준점을 정하는데 걱정할 필요가 없어서 더 선호하기도 하다

4. Quick Sort

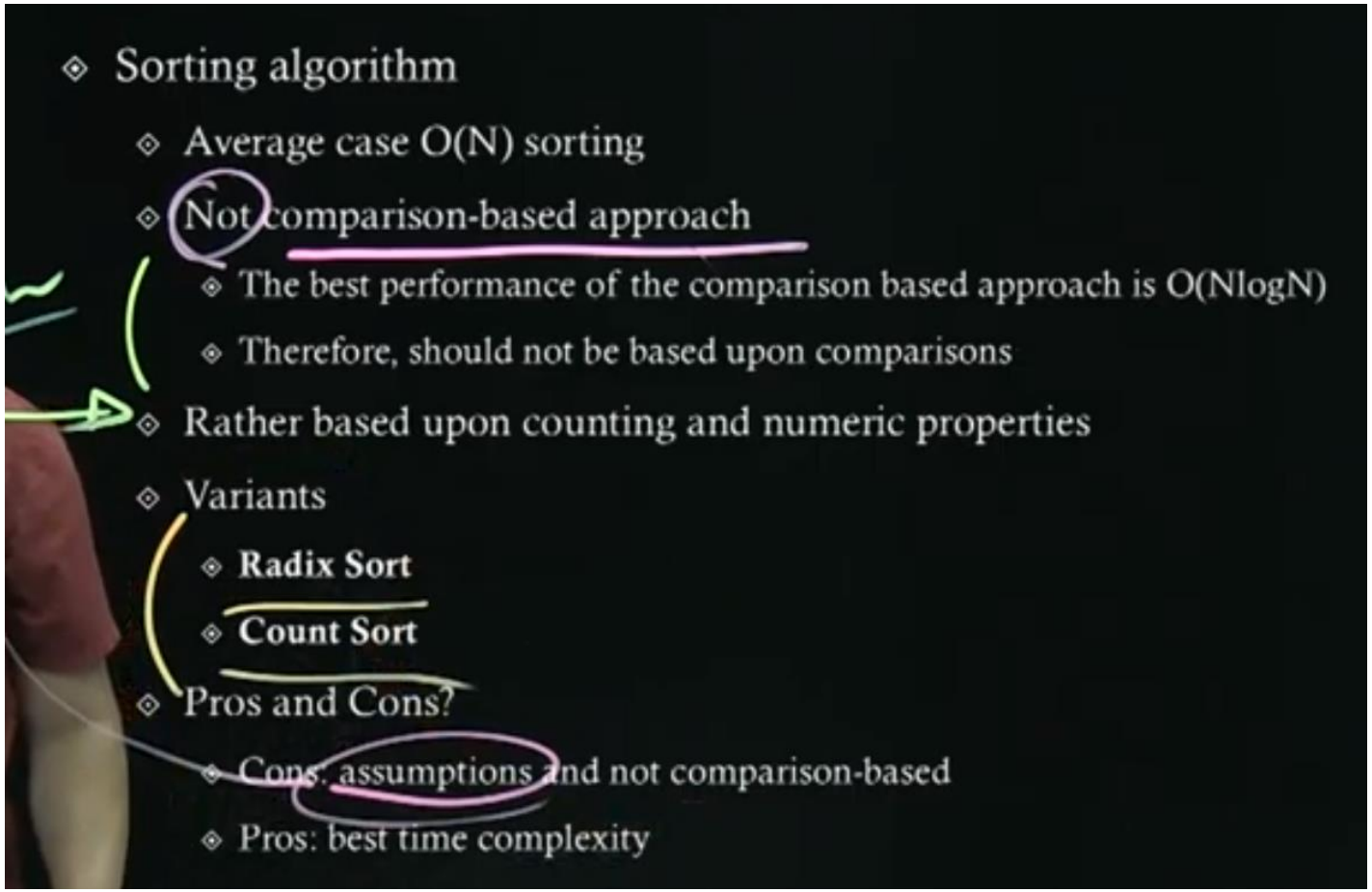
Implementation of quick sort



- Pivot은 0로 설정(idx를 맨 처음으로 설정)
- Recursion을 해야하기 때문에 escape문이 있다.
- Pivotvalue는 항상 앞에 있는 값
- Pivotvalue보다 작은거=less, 큰거=greater 만듦
- Pivotvalue보다 크면 greater
- Pivotvalue보다 작으면 less
- Recursion이 두 파트가 일어남
- Return은 less + pivotvalue + greater로 sort가 됨
- Merge sort는 division 할때 sort 안했지만 quick sort는 division 할때 sort를 하게 됨
- Aggregation하기 전에 sort하는 것

5. Counting Sort

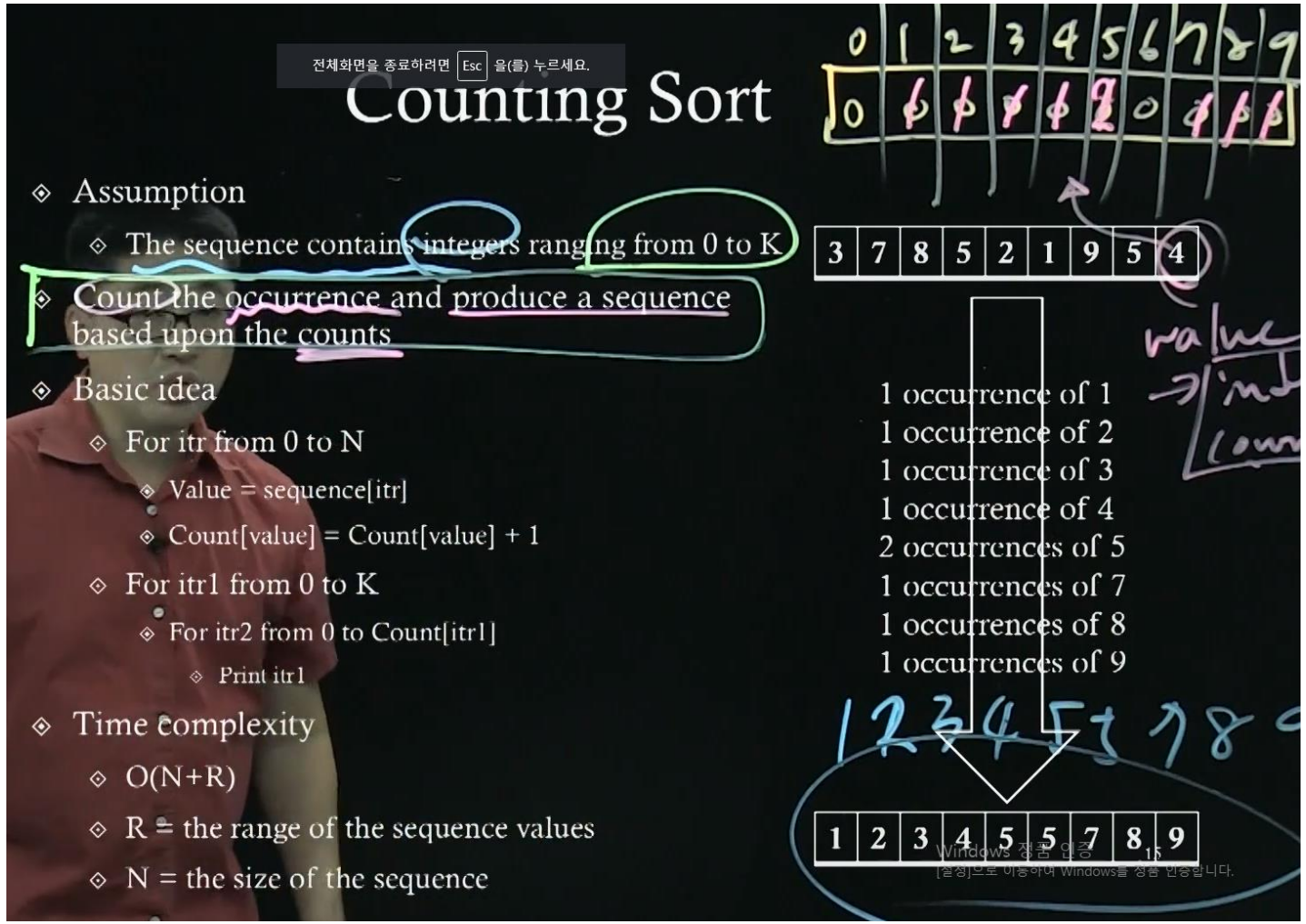
O(N) Sorting



- O(N) Sorting은 비교를 하지 않는다
- 비교를 하지 않았지만 비교를 한것과 같은 결과를 만들어 내야 O(NlogN)의 성능을 낼 수 있다.
- Counting and numeric propeties기반하에 assumption 하는것
- 장점으로 가장 빠르다

5. Counting Sort

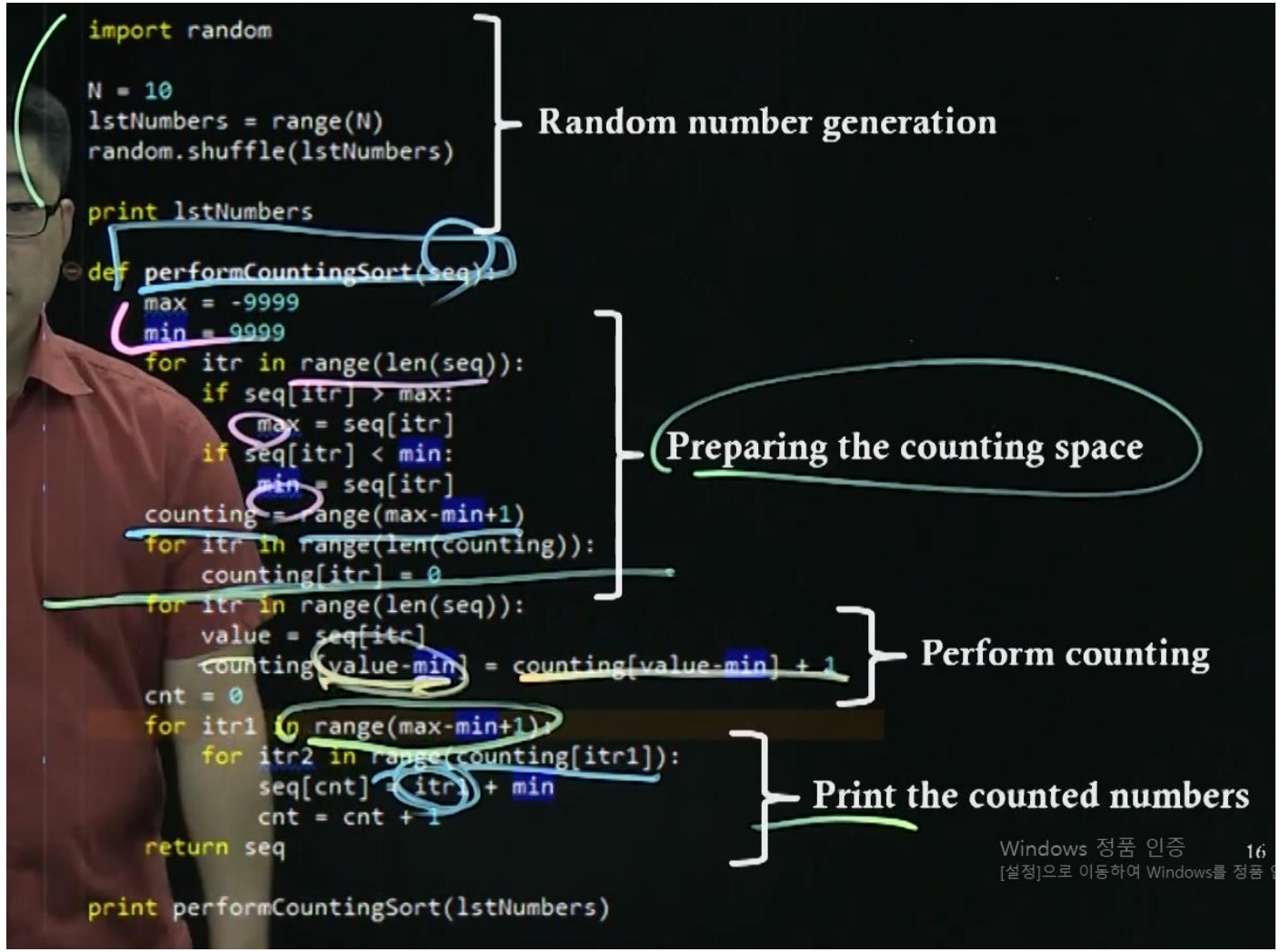
Counting Sort



- Counting sort는 integers만 가능
- 0 from K로 맥스 K를 알아야 함
- 각 숫자가 몇번 일어나는지 count함
- Count에 맞춰 seqeunce를 produce함
- Array를 max까지 만든다.
- 0으로 초기화한다.
- Value가 index가 돼 버린다.

5. Counting Sort

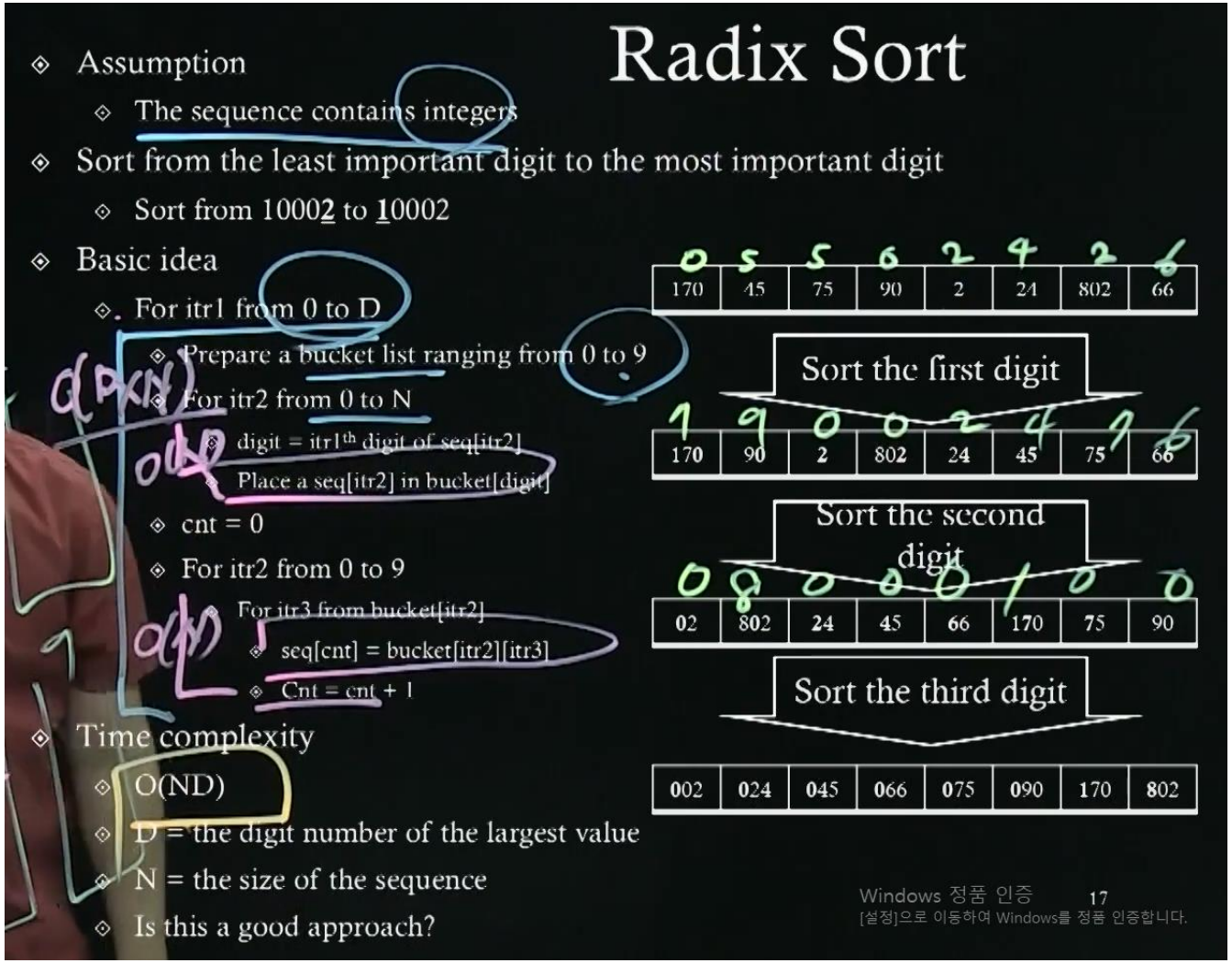
Implementation of counting sort



- Seq의 길이 만큼 한번 훑어서 min과 max을 알아낸다.
- 0부터 k까지의 저장소를 만들기 위함
- Counting을 0으로 초기화 한다.
- Value-min = min이 꼭 0이 아니어도 0부터 만들수 있게한 code
- 그 다음 counting 함
- Counting된 index값을 sequence로 print

6. Radix Sort

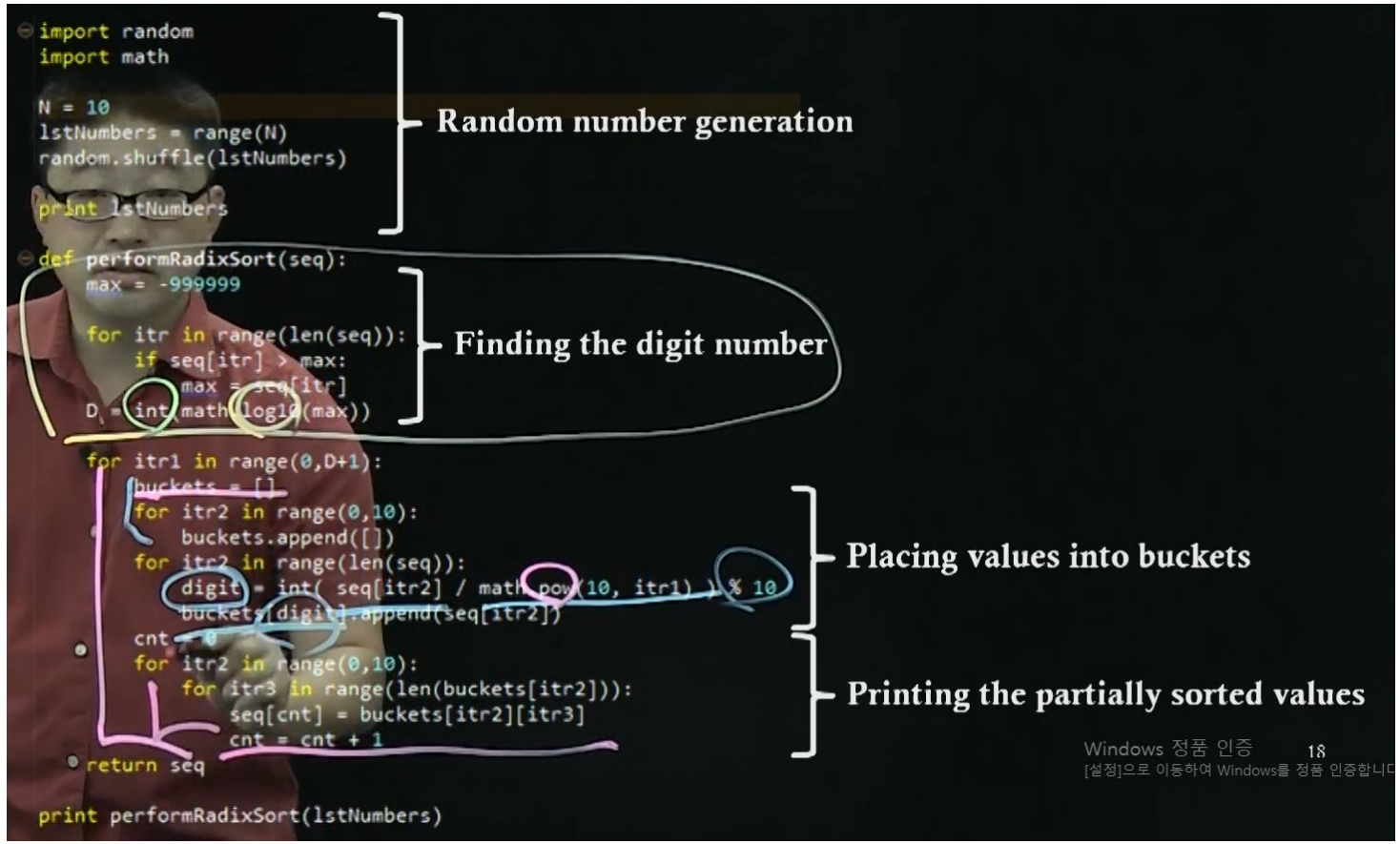
Radix Sort



- Sequence는 integer여야 한다.
- Counting은 array의 길이가 길면 비효율적인데 이것을 보완하기 위해 만든것이 Radex Sort다
- 마지막 자리에 있는 digit부터 첫번째 digit까지 count하면서 sort
- Bucket에 넣고 순서대로 print하면 된다.

6. Radix Sort

Implementation of radix sort



- Digit의 number를 알아보는 과정
- Log10을 활용
- Log10을 활용하면 소수점 자리가 나올수 있어서 int로 처리
- Outer loop는 digit를 돈다
- Buckets을 만든다.
- Pow가 10의 몇승의 자리인이 알아보는것
- Bucket에 자리수 별로 집어 넣고
- 다음 loop에서 bucket에 있는 값을 print한다.

6. Radix Sort

Performance of sorting algorithms

	Average Case	Worst Case
Selection Sort	$O(N^2)$	$O(N^2)$
Merge Sort	O(NlogN)	O(NlogN)
Heap Sort	O(NlogN)	O(NlogN)
Quick Sort	O(NlogN)	$O(N^2)$
Counting Sort	O(N+R)	O(N+R)
Radix Sort	O(ND)	O(ND)

- In the real world
  - Many people do not concern the time complexity of the sorting
  - Why?
    - Most of time, people rely on the database and "DESC" and "ASC"
    - Most of time, people do not give too much thought on this issue
      - Not a good idea
- You need to consider the cost of your system
  - Development
  - Maintenance