CS101 –리스트 활용 예제: 정렬과 소수 구하기

Lecture 16

School of Computing
KAIST

학습 목표:

- 2중 반복문을 통해 리스트를 정렬할 수 있다.
- 소수 (prime number)를 구하는 알고리즘을 구현할 수 있다.

다시보기 : 정렬

리스트는 sort 함수를 이용해 원소들을 정렬할 수 있습니다.

```
>>> ta = [ "JinYeong", "Jeongmin", "Minsuk",
    "Dohoo", "Sangjae", "Byung-Jun" ]
>>> ta.sort()
>>> ta
['Byung-Jun', 'Dohoo', 'Jeongmin', 'JinYeong',
    'Minsuk', 'Sangjae']
```

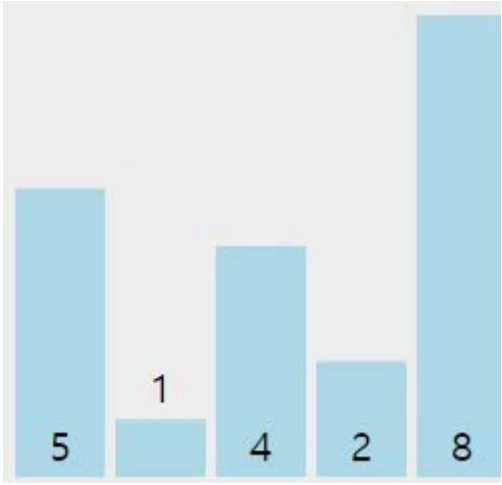
totals 리스트를 정렬해봅시다.

```
>>> totals.sort()
>>> totals
[(1, 'Croatia'), (1, 'Kazakhstan'), (1, 'Slovakia'),
    (2, 'Ukraine'), (3, 'Australia'), ..., (8,
    'Japan'), (8, 'Slovenia'), (8, 'South Korea'),
    ..., (33, 'Russia')]
```

버블 정렬

Sort 시각화 참고
https://visualgo.net/en/sorting

```
# 리스트 a의 원소들을 크기 순으로 정렬
def bubbleSort(a):
   sorted = False
   while(not sorted):
      sorted = True
      for i in range(1,len(a)):
         if(a[i-1] > a[i]):
            a[i-1],a[i]=a[i],a[i-1]
            sorted = False
```



예. bubbleSort는
[5,1,4,2,8]을
[1,2,4,5,8]로
정렬합니다.

버블 정렬

i	a [i-1]	a [i]	a[i-1] > a[i]
1	5	1	True

첫번째 반복

(**5 1** 4 2 8) -> (**1 5** 4 2 8)

```
# 리스트 a의 원소들을 크기 순으로 정렬
def bubbleSort(a):
  sorted = False
  while(not sorted):
    sorted = True
    for i in range(1,len(a)):
      if(a[i-1] > a[i]):
        a[i-1],a[i]=a[i],a[i-1]
        sorted = False
```

예. bubbleSort는 [5,1,4,2,8]을 [1,2,4,5,8]로 정렬합니다.

버블 정렬

i	a [i-1]	a [i]	a[i-1] > a[i]
1	5	1	True
2	5	4	True

첫번째 반복

(**5 1** 4 2 8) -> (**1 5** 4 2 8)

(1 **5 4** 2 8) -> (1 **4 5** 2 8)

```
# 리스트 a의 원소들을 크기 순으로 정렬
def bubbleSort(a):
  sorted = False
  while(not sorted):
    sorted = True
    for i in range(1,len(a)):
      if(a[i-1] > a[i]):
        a[i-1],a[i]=a[i],a[i-1]
        sorted = False
```

예. bubbleSort는 [5,1,4,2,8]을 [1,2,4,5,8]로 정렬합니다.

버블 정렬

i	a [i-1]	a [i]	a[i-1] > a[i]
1	5	1	True
2	5	4	True
3	5	2	True

첫번째 반복

(**5 1** 4 2 8) -> (**1 5** 4 2 8)

(1 **5 4** 2 8) -> (1 **4 5** 2 8)

(1 4 **5 2** 8) -> (1 4 **2 5** 8)

```
# 리스트 a의 원소들을 크기 순으로 정렬
def bubbleSort(a):
  sorted = False
  while(not sorted):
    sorted = True
    for i in range(1,len(a)):
      if(a[i-1] > a[i]):
        a[i-1],a[i]=a[i],a[i-1]
        sorted = False
```

예. bubbleSort는 [5,1,4,2,8]을 [1,2,4,5,8]로 정렬합니다.

버블 정렬

i	a [i-1]	a [i]	a[i-1] > a[i]
1	5	1	True
2	5	4	True
3	5	2	True
4	5	8	False

첫번째 반복

(**5 1** 4 2 8) -> (**1 5** 4 2 8)

(1 **5 4** 2 8) -> (1 **4 5** 2 8)

(1 4 **5 2** 8) -> (1 4 **2 5** 8)

(1 4 2 **5 8**) = (1 4 2 **5 8**)

sorted = False

```
# 리스트 a의 원소들을 크기 순으로 정렬
def bubbleSort(a):
  sorted = False
  while(not sorted):
    sorted = True
    for i in range(1,len(a)):
      if(a[i-1] > a[i]):
        a[i-1],a[i]=a[i],a[i-1]
        sorted = False
```

예. bubbleSort는 [5,1,4,2,8]을 [1,2,4,5,8]로 정렬합니다.

버블 정렬

```
# 리스트 a의 원소들을 크기 순으로 정렬
def bubbleSort(a):
    sorted = False
    while(not sorted):
        sorted = True
        for i in range(1,len(a)):
            if(a[i-1] > a[i]):
                a[i-1],a[i]=a[i],a[i-1]
                sorted = False
```

예. bubbleSort는 [5,1,4,2,8]을 [1,2,4,5,8]로 정렬합니다.

i	a [i-1]	a [i]	a[i-1] > a[i]
1	5	1	True
2	5	4	True
3	5	2	True
4	5	8	False

i	a [i-1]	a [i]	a[i-1] > a[i]
1	1	4	False
2	4	2	True
3	4	5	False
4	5	8	False

첫번째 반복

(**5 1** 4 2 8) -> (**1 5** 4 2 8)

(1 **5 4** 2 8) -> (1 **4 5** 2 8)

(1 4 **5 2** 8) -> (1 4 **2 5** 8)

(1 4 2 **5 8**) = (1 4 2 **5 8**)

sorted = False

↓

두번째 반복

(**1 4** 2 5 8) = (**1 4** 2 5 8)

(1 **4 2** 5 8) -> (1 **2 4** 5 8)

(1 2 **4 5** 8) = (1 2 **4 5** 8)

(1 2 4 **5 8**) = (1 2 4 **5 8**)

sorted = False

버블 정렬

```
# 리스트 a의 원소들을 크기 순으로 정렬
def bubbleSort(a):
    sorted = False
    while(not sorted):
        sorted = True
        for i in range(1,len(a)):
            if(a[i-1] > a[i]):
                a[i-1],a[i]=a[i],a[i-1]
                sorted = False
```

예. bubbleSort는 [5,1,4,2,8]을 [1,2,4,5,8]로 정렬합니다.

i	a [i-1]	a [i]	a[i-1] > a[i]
1	5	1	True
2	5	4	True
3	5	2	True
4	5	8	False

i	a [i-1]	a [i]	a[i-1] > a[i]
1	1	4	False
2	4	2	True
3	4	5	False
4	5	8	False

i	a [i-1]	a [i]	a[i-1] > a[i]
1	1	2	False
2	2	4	False
3	4	5	False
4	5	8	False

첫번째 반복

(**5 1** 4 2 8) -> (**1 5** 4 2 8)

(1 **5 4** 2 8) -> (1 **4 5** 2 8)

(1 4 **5 2** 8) -> (1 4 **2 5** 8)

(1 4 2 **5 8**) = (1 4 2 **5 8**)

sorted = False

두번째 반복

(**1 4** 2 5 8) = (**1 4** 2 5 8)

(1 **4 2** 5 8) -> (1 **2 4** 5 8)

(1 2 **4 5** 8) = (1 2 **4 5** 8)

(1 2 4 **5 8**) = (1 2 4 **5 8**)

sorted = False

세번째 반복

(**1 2** 4 5 8) = (**1 2** 4 5 8)

(1 **2 4** 5 8) = (1 **2 4** 5 8)

(1 2 **4 5** 8) = (1 2 **4 5** 8)

(1 2 4 **5 8**) = (1 2 4 **5 8**)

sorted = True :프로그램 종료

소수 구하기

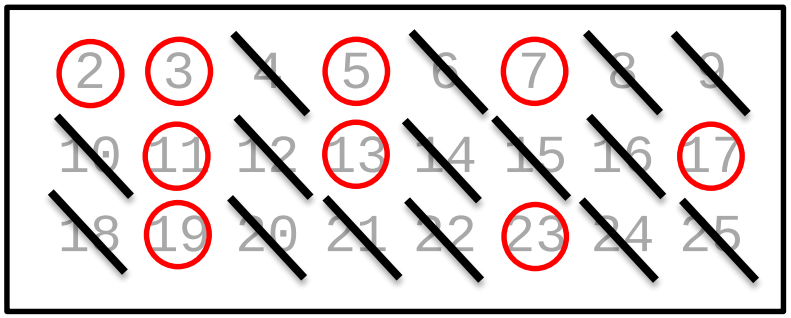
에라토스테네스의 체를 사용해봅시다.

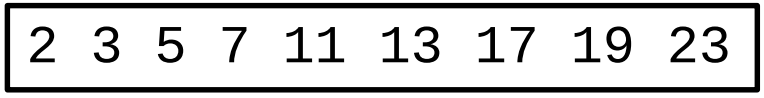
```
# n 보다 작은 소수 리스트 구하기
def sieve(n):
  candidates = list(range(2,n))
  i = 0
  # i: 확인된 소수에 대한 인덱스
  while i < len(candidates):
    prime = candidates[i]

    # j: 소수임을 확인해야 하는
    # 숫자에 대한 인덱스
    j = i + 1
    while j < len(candidates):
      if candidates[j]%prime==0:
        candidates.pop(j)
      else :
        j= j+1
    i = i + 1
  return candidates
```

sieve(26):

n이 26일 때 반환되는 리스트

candidates=





정리 및 예습

본 강의 학습 목표:

- 2중 반복문을 통해 리스트를 정렬할 수 있다.
- 소수 (prime number)를 구하는 알고리즘을 구현할 수 있다.

다음 강의 학습 목표:

- 문자열을 다양한 방법으로 활용할 수 있다.
- 집합 (set) 자료구조를 활용할 수 있다.