CS101 – 시퀀스: 리스트, 문자열, 튜플

Lecture 15

School of Computing
KAIST

학습 목표:

- 리스트를 활용하는 다양한 방법을 이해할 수 있다.
- 시퀀스를 표현하는 리스트, 문자열, 튜플의 차이점을 이해한다.

많은 데이터

다음 표는 2014 소치 동계올림픽의 메달 집계 결과입니다.

Australia	0	2	1
Austria	4	8	5
Belarus	5	0	1
Canada	10	10	5
China	3	4	2
Croatia	0	1	0
Czech Republic	2	4	2
Finland	1	3	1
France	4	4	7
Germany	8	6	5
Great Britain	1	1	2
Italy	0	2	6
Japan	1	4	3
Kazakhstan	0	0	1
Latvia	0	2	2
Netherlands	8	7	9
Norway	11	5	10
Poland	4	1	1
Russia	13	11	9
Slovakia	1	0	0
Slovenia	2	2	4
South Korea	3	3	2
Sweden	2	7	6
Switzerland	6	3	2
Ukraine	1	0	1
United States	9	7	12

이 데이터를 Python에서 어떻게 저장할 수 있을까요?
하나하나 변수로 만들려면 총 4×26개의 변수가 필요하네요..

리스트(List)를 사용하면 여러 값들을 모아서 보관할 수 있습니다.

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')]
```

리스트 자르기

리스트의 특정 부분을 잘라내서 **새로운 리스트**로 만들 수 있습니다 (slicing)

```
sublist = mylist[i:j]
```

`sublist`는 `mylist`의 인덱스 `i`, `i+1`, …, `j-1`에 해당하는 원소를 포함한 리스트입니다.

`i` 를 생략하면, `sublist`는 `mylist`의 첫 번째 원소부터 가지게 됩니다.

`j` 를 생략하면, `sublist`는 `mylist`의 마지막 원소까지 가지게 됩니다.

다음처럼 쓰면 리스트를 복사할 수 있습니다.

```
list2 = list1[:]
```

리스트 뒤집기

다음처럼 리스트를 뒤집어, 메달이 많은 나라가 앞쪽에 오게 할 수 있습니다.

```
>>> totals.reverse()
>>> totals
[(33, 'Russia'), (28, 'United States'), ..., (8,
    'South Korea'), ..., (1, 'Kazakhstan'), (1,
    'Croatia')]
```

메달이 가장 많은 10개의 나라만 따로 볼 수도 있습니다.

```
>>> top_ten = totals[:10]
>>> for p in top_ten:
...     medals, country = p
...     print(medals, country)
```

리스트의 원소는 다음처럼 풀어낼 수 있습니다.

```
>>> for medals, country in top_ten:
...     print(medals, country)
```

메달 순위

메달 수 상위 10개의 나라를 메달수에 관해 정렬해봅시다.

```
table = []
for i in range(len(countries)):
  table.append( (gold[i], silver[i], bronze[i],
      countries[i]) )
table.sort()
top_ten = table[-10:]
top_ten.reverse()
for g, s, b, country in top_ten:
  print(country, g, s, b)
```

```
Russia 13 11 9
Norway 11 5 10
Canada 10 10 5
United States 9 7 12
Netherlands 8 7 9
Germany 8 6 5
Switzerland 6 3 2
Belarus 5 0 1
Austria 4 8 5
France 4 4 7
```

한 종류의 메달만 획득한 나라

한 종류의 메달만 획득한 나라들을 모두 찾아봅시다.
(메달을 하나도 획득하지 못한 나라는 없다고 가정합니다)

```
def no_medals(countries, al, bl):
  result = []
  for i in range(len(countries)):
    if al[i] == 0 and bl[i] == 0:
      result.append(countries[i])
  return result

only_gold = no_medals(countries, silver, bronze)
only_silver = no_medals(countries, gold, bronze)
only_bronze = no_medals(countries, gold, silver)

only_one = only_gold + only_silver + only_bronze
```

많은 데이터

다음 표는 2014 소치 동계올림픽의 메달 집계 결과입니다.

Australia	0	2	1
Austria	4	8	5
Belarus	5	0	1
Canada	10	10	5
China	3	4	2
Croatia	0	1	0
Czech Republic	2	4	2
Finland	1	3	1
France	4	4	7
Germany	8	6	5
Great Britain	1	1	2
Italy	0	2	6
Japan	1	4	3
Kazakhstan	0	0	1
Latvia	0	2	2
Netherlands	8	7	9
Norway	11	5	10
Poland	4	1	1
Russia	13	11	9
Slovakia	1	0	0
Slovenia	2	2	4
South Korea	3	3	2
Sweden	2	7	6
Switzerland	6	3	2
Ukraine	1	0	1
United States	9	7	12

리스트의 멤버 함수

리스트 객체 L은 다음과 같은 멤버 함수들을 가지고 있습니다.

- `L.append(v)` v 객체를 리스트 끝에 추가
- `L.insert(i, v)` 객체를 리스트의 i번째 위치에 추가
- `L.pop()` 리스트의 마지막 원소를 삭제하고, 그 값을 반환
- `L.pop(i)` i번째 원소를 삭제하고, 그 값을 반환
- `L.remove(v)` v와 일치하는 첫 번째 원소를 삭제
- `L.index(v)` v와 일치하는 첫 번째 원소의 위치를 반환
- `L.count(v)` v와 일치하는 원소들의 개수를 반환
- `L.extend(K)` K의 모든 원소를 리스트 L 끝에 추가
- `L.reverse()` 리스트의 모든 원소를 역순으로 재배열
- `L.sort()` 리스트 정렬

다음 두 코드는 어떤 차이가 있을까요?

```
L.append(13)
L + [13]
```

시퀀스

리스트는 **시퀀스(Sequence)**의 한 종류입니다.
문자열과 튜플 또한 시퀀스의 한 종류입니다.

문자열

```
>>> a = "CS101"
>>> a[-1]
'1'
>>> a[2:]
'101'
>>> for i in a:
...  print (i)
C
S
1
0
1
```

튜플

```
>>> t = ("CS101", "A+", 13)
>>> t[0]
'CS101'
>>> t[-1]
13
>>> t[1:]
('A+', 13)
>>> for i in t:
...  print (i)
CS101
A+
13
```

리스트, 튜플, 문자열

KAIST

리스트과 튜플은 비슷하지만 큰 차이점이 있습니다.
리스트는 **가변 객체**지만, 튜플(과 문자열)은 **불변 객체**입니다.

```
>>> t[0] = "CS206"
TypeError: 'tuple' object does not support item
    assignment
```

시퀀스는 `list`, `tuple` 함수를 이용해 리스트이나 튜플로 만들 수 있습니다.

```
>>> list(t)
['CS101', 'A+', 13]
>>> tuple(gold)
(0, 4, 5, 10, 3, 0, 2, 1, 4, ..., 2, 6, 1, 9)
>>> list("CS101")
['C', 'S', '1', '0', '1']
```

메달 리스트

Python에서는 올림픽 메달 집계 정보를 리스트 4개로 만드는 것보다, 튜플들의 리스트로 만드는 방법을 더 많이 사용합니다.

```
medals = [ ( 'Australia', 0, 2, 1 ),
           ( 'Austria', 4, 8, 5 ),
           ...
           ( 'United States', 9, 7, 12 ) ]
```

나라별 총 메달 수는 다음처럼 출력할 수 있습니다.

```
def print_totals1():
  for country, g, s, b in medals:
    print(country + ":", g + s + b)

def print_totals2():
  for item in medals:
    print(item[0] + ":", sum(item[1:]))
```

히스토그램

메달 집계 결과로 히스토그램을 만들어 봅시다.

```
def histogram():
    t = [0] * 13
    for item in medals:
        total = sum(item[1:])
        t[total // 3] += 1
    for i in range(13):
        print (str(3*i) + "~" +
            str(3*i+2)+":\t"+("*"* t[i]))
```

실행 결과

```
0~2:    ****
3~5:    ****
6~8:    *******
9~11:   **
12~14:
15~17:  ***
18~20:  *
21~23:
24~26:  ***
27~29:  *
30~32:
33~35:  *
36~38:
```

히스토그램

메달 집계 결과로 히스토그램을 만들어 봅시다.

```
def histogram():
    t = [0] * 13
    for item in medals:
        total = sum(item[1:])
        t[total // 3] += 1
    for i in range(13):
        print (str(3*i) + "~" +
            str(3*i+2)+":\t"+("*"* t[i]))
```

실행 결과

```
t[0]   0~2:      ****
t[1]   3~5:      ****
t[2]   6~8:      ********
t[3]   9~11:     **
t[4]   12~14:
t[5]   15~17:    ***
t[6]   18~20:    *
t[7]   21~23:
t[8]   24~26:    ***
t[9]   27~29:    *
t[10]  30~32:
t[11]  33~35:    *
t[12]  36~38:
```

item	total	total // 3
('Australia', 0, 2, 1)	3	1
('Austria', 4, 8, 5)	17	5
('Belarus', 5, 0, 1)	6	2
('Canada', 10, 10, 5)	25	8
('China', 3, 4, 2)	9	3
('Croatia', 0, 1, 0)	1	0
('Czech Republic', 2, 4,2)	8	2

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	...
0	1	0	0	0	0	0	0	0	...
0	1	0	0	0	1	0	0	0	...
0	1	1	0	0	1	0	0	0	...
0	1	1	0	0	1	0	0	1	...
0	1	1	1	0	1	0	0	1	...
1	1	1	1	0	1	0	0	1	...
1	1	2	1	0	1	0	0	1	...

정리 및 예습

본 강의 학습 목표:

- 리스트를 활용하는 다양한 방법을 이해할 수 있다.
- 시퀀스를 표현하는 리스트, 문자열, 튜플의 차이점을 이해한다.

다음 강의 학습 목표:

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