CS101 -리스트 활용법

Lecture 14

School of Computing
KAIST

학습 목표:

- 리스트를 통해 복잡한 자료를 프로그램으로 분석하는 방법을 이해할 수 있다.

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.

리스트

리스트는 여러 값들을 대괄호 안에 나열해서 적는 방법으로 만들 수 있습니다.

```
>>> countries = [ "Australia", ... , "United States" ]
>>> gold = [0, 4, 5, 10, 3, 0, 2, 1, 4, 8, 1, 0, 1, 0, 0,
     8, 11, 4, 13, 1, 2, 3, 2, 6, 1, 9]
```

리스트는 list 타입의 객체입니다.

리스트의 각 원소는 위치 값을 사용해서 접근할 수 있습니다.

첫 번째 원소는 0번째 위치에, 두 번째 원소는 1번째 위치에 있습니다.

```
>>> countries[0]
'Australia'
>>> countries[21]
'South Korea'
>>> gold[21]
3
```

음수 위치를 사용하면 리스트의 끝에서부터 접근할 수 있습니다.

```
>>> countries[-1]
'United States'
>>> countries[-5]
'South Korea'
```

리스트

리스트의 길이는 len을 사용해서 구할 수 있습니다.

```
>>> len(countries)
26
```

빈 리스트는 [] 로 표기할 수 있습니다. 빈 리스트의 길이는 0입니다.

하나의 리스트는 여러 다른 종류의 객체를 담을 수도 있습니다.

```
>>> korea = [ 'Korea', 'KR', 3, 3, 2 ]
>>> korea[1]
'KR'
>>> korea[2]
3
```

튜플을 담을 수도 있습니다.

```
>>> korea = [ "Korea", 'KR', (3, 3, 2) ]
```

리스트과 관련된 내장 함수들

len은 리스트의 길이를 반환합니다.
sum은 리스트의 각 원소의 합을 반환합니다.
max는 리스트에서 가장 큰 원소를, min은 가장 작은 원소를 반환합니다.

```
>>> len(gold), sum(gold), max(gold), min(gold)
(26, 99, 13, 0)
>>> len(silver), sum(silver), max(silver)
(26, 97, 11)
>>> len(bronze), sum(bronze), max(bronze)
(26, 99, 12)
```

리스트는 가변 객체이다

다음 리스트는 비활성기체의 이름을 담은 리스트입니다.

```
>>> nobles = [ 'helium', 'none', 'argon', 'krypton',
    'xenon' ]
```

저런, 오타가 있네요. 오타를 고쳐볼게요.

```
>>> nobles[1] = "neon"
>>> nobles
['helium', 'neon', 'argon', 'krypton', 'xenon']
```

이번에는 라돈을 빠뜨렸네요.

```
>>> nobles.append('radon')
>>> nobles
['helium', 'neon', 'argon', 'krypton', 'xenon',
 'radon']
```

Aliasing

다시 보기: 하나의 객체는 여러 이름을 가질 수 있습니다.
이를 Aliasing이라 부릅니다.
가변 객체를 사용할 때는 객체가 잘못 바뀌지 않도록 조심해서 사용해야 합니다.

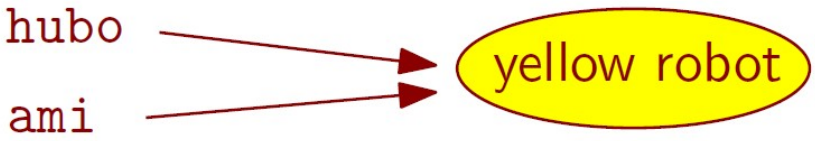
```
>>> list1 = ["A","B","C"]
>>> list2 = list1
>>> len(list1)
3
>>> list2.append("D")
>>> len(list1)
4
>>> list1[1] = "X"
>>> list2
['A', 'X', 'C', 'D']
>>> list1 is list2
True
```

```
>>> list1 = ["A","B","C"]
>>> list2 = ["A","B","C"]
>>> len(list1)
3
>>> list2.append("D")
>>> len(list1)
3
>>> list1[1] = "X"
>>> list2
['A', 'B', 'C', 'D']
>>> list1 is list2
False
```

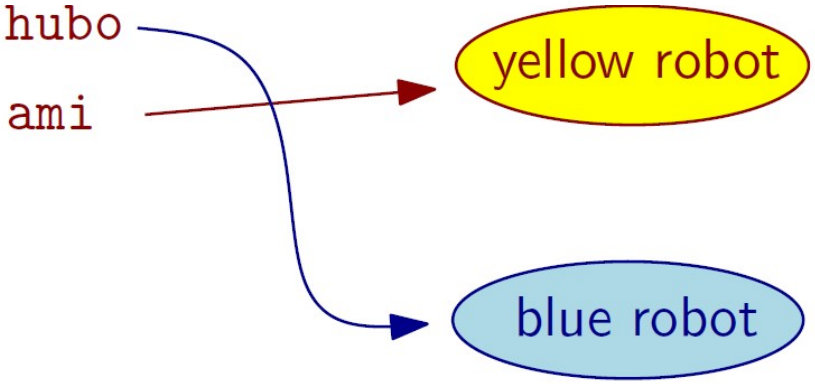
다시 보기: 여러 이름을 가진 객체

하나의 객체는 여러 이름을 가질 수 있습니다.

```
hubo = Robot("yellow")
hubo.move()
ami = hubo
ami.turn_left()
hubo.move()
```




```
hubo = Robot("blue")
hubo.move()
ami.turn_left()
ami.move()
```



리스트 탐색하기

반복문을 이용해 리스트의 각 원소를 탐색할 수 있습니다.

```
for country in countries:
  print(country)
```

함수 **range**는 range 타입의 객체를 만들어주는 함수입니다.

```
>>> range(10)
range(0, 10)
>>> type(range(10))
<class 'range'>
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(10, 15))
[10, 11, 12, 13, 14]
```

리스트의 원소 값을 바꾸고 싶다면, 그 원소의 위치를 알아야 합니다.

```
>>> l = list(range(1, 11))
>>> for i in range(len(l)):
...   l[i] = l[i] ** 2
>>> l
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

리스트 탐색하기

나라별 총 메달 수를 출력해봅시다.

```
>>> for i in range(len(countries)):
...  print(countries[i], gold[i]+silver[i]+bronze[i])
```

위에서 구한 값을 저장하는 새로운 리스트를 만들 수 있습니다.

```
>>> totals = []
>>> for i in range(len(countries)):
...   medals = gold[i]+silver[i]+bronze[i]
...   totals.append( (medals, countries[i]) )
```

새로 만든 `totals` 리스트는 튜플 **(medals, country)** 의 리스트입니다.

```
>>> totals
[(3, 'Australia'), (17, 'Austria'), (6, 'Belarus'),
    ..., (4, 'Latvia'), (24, 'Netherlands'), ...,
    (8, 'South Korea'), ..., (2, 'Ukraine'), (28,
    'United States')]
```

정리 및 예습

본 강의 학습 목표:

- 리스트를 통해 복잡한 자료를 프로그램으로 분석하는 방법을 이해할 수 있다.

다음 강의 학습 목표:

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