

파이썬 프로그래밍

리스트의 기초



한국기술교육대학교
온라인평생교육원

■ 리스트 생성 및 기본 연산

- 리스트: 시퀀스 자료형이면서 변경 가능(Mutable)
- 리스트에서 지원하는 일반적인 연산
 - 인덱싱, 슬라이싱, 연결, 반복, 멤버십 테스트

```
l = []  
l = [1, 2, "Great"]  
print l[0], l[-1]  
print l[1:3], l[:]  
print
```

```
L = range(10)  
print L[:2]  
print
```

```
print l * 2  
print l + [3, 4, 5]  
print len(l)  
print  
print 4 in L
```

```
1 Great  
[2, 'Great'] [1, 2, 'Great']  
  
[0, 2, 4, 6, 8]  
  
[1, 2, 'Great', 1, 2, 'Great']  
[1, 2, 'Great', 3, 4, 5]  
3  
  
True
```

▣ 리스트 생성 및 기본 연산

- 시퀀스 자료형 : 각 원소에 존재하는 인덱스를 기반으로 한 여러 연산
- 리스트 자료형 → 변경 가능
- 각 원소마다 인덱스가 붙음
- 1 → 인덱스 0, 2 → 인덱스 1, Great → 인덱스 2
- l[0] → 1, l[-1] → Great 출력
- l[1:3] → 슬라이싱으로 인덱스 1에서 2까지
- l[:] → 전체 내용을 다 가지고 옴
- range(10) → 1에서 9까지의 숫자가 차례대로 리스트화
- l[::2] → 확장 슬라이드 → step부분에 2로 하나씩 건너뛰워 가져옴
- l * 2 → 리스트를 두 번 반복
- l + [3, 4, 5] → 리스트와 리스트 그대로 연결
- len(l) → 리스트 l 안 원소의 개수

▣ 리스트 생성 및 기본 연산

- Mutable 특성을 지닌 리스트의 값 변경

```
a = ['spam', 'eggs', 100, 1234]
a[2] = a[2] + 23
print a
```

```
['spam', 'eggs', 123, 1234]
```

- 리스트 → Mutable한 특징이 있어 바뀔 수 있음
- *, + 형태로 바뀌는 것이 아니라 치환(할당)연산에 의해 변경됨
- $a[2] = a[2] + 23 \rightarrow a$ 의 인덱스 2 자리에 새로운 값을 할당해라

▣ 리스트 생성 및 기본 연산

- 리스트 원소에 대한 슬라이스 치환

```
a = ['spam', 'eggs', 123, 1234]
a[0:2] = [1,12] # 동일한 크기에 대한 슬라이스 치환
print a
```

```
a[0:2] = [1] # 서로 다른 크기에 대한 슬라이스 치환
print a
```

```
a[0:1] = [1, 2, 3] # 서로 다른 크기에 대한 슬라이스 치환
print a
```

```
[1, 12, 123, 1234]
[1, 123, 1234]
[1, 2, 3, 123, 1234]
```

- a[0:2]는 =의 왼쪽에 들어가서 슬라이싱X, 위치를 잡음
- a[0:2] = [1,12] → 'spam'과 'eggs' 자리에 리스트가 들어감

▣ 리스트 생성 및 기본 연산

- 리스트 원소에 대한 슬라이스 삭제

```
a = [1, 12, 123, 1234]
```

```
a[0:2] = []
```

```
print a
```

```
[123, 1234]
```

- $a[0:2] \rightarrow 1, 12$ 의 자리

▣ 리스트 생성 및 기본 연산

- 리스트 원소에 대한 슬라이스 삽입

```
a = [123, 1234]
a[1:1] = ['spam', 'ham'] # 1번째 인덱스에 삽입
print a
```

```
b = [123, 1234]
b[1:2] = ['spam', 'ham'] # 1번째 원소에 대한 치환
print b
```

```
a[:0] = a # 리스트 맨 앞에 자기 자신을 삽입
print a
```

```
[123, 'spam', 'ham', 1234]
[123, 'spam', 'ham']
[123, 'spam', 'ham', 1234, 123, 'spam', 'ham', 1234]
```

- a[1:1] → 시작과 끝이 1 이므로, 1 자리의 바로 앞을 의미
- b[1:2] → 1234 자리
- a[:0] = a[0:0] → 0 자리의 바로 앞을 의미

■ 리스트 생성 및 기본 연산

- del을 이용한 리스트 요소 삭제

```
a = [1, 2, 3, 4]
del a[0]
print a

del a[1:]
print a
```

```
[2, 3, 4]
[2]
```

- del → delete의 약자
- del → 뒤에 인덱스 연산, 슬라이스 연산 모두 가능
- a[1:] → 1부터 끝까지 가리킴

```
a = range(4)
print a
print a[::2]

del a[::2]
print a
```

```
[0, 1, 2, 3]
[0, 2]
[1, 3]
```

- range(4) = [0, 1, 2, 3]

▣ 리스트 생성 및 기본 연산

- 리스트 자체에 대한 삭제
 - 정확히는 리스트를 가리키는 레퍼런스를 지닌 변수 a 삭제

```
a = range(5)
del a
print a
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-3-0820131cb511> in <module>()
      1 a = range(5)
      2 del a
----> 3 print a

NameError: name 'a' is not defined
```

- del a → 리스트가 삭제되는 것이 아니라 레퍼런스 a가 삭제

- 중첩 리스트: 리스트 안에 요소로서 리스트를 지닌 리스트

```
s = [1, 2, 3]
t = ['begin', s, 'end']
print t

print t[1][1]
```

```
['begin', [1, 2, 3], 'end']
2
```

- 인덱스 연산을 여러 개 붙일 수 있음
- t[1] → s → t[1][1] = s[1]

■ 리스트 생성 및 기본 연산

```
s[1] = 100  
print t
```

```
['begin', [1, 100, 3], 'end']
```

- s 리스트 값을 바꿨지만 t 리스트에도 반영됨

```
L = [1, ['a', ['x', 'y'], 'b'], 3]  
print L[0]  
print L[1]  
print L[1][1]  
print L[1][1][1]
```

```
1  
['a', ['x', 'y'], 'b']  
['x', 'y']  
y
```

- L[1] = ['a', ['x', 'y'], 'b']
- L[1][1] = ['x', 'y']
- L[1][1][1] = 'y'

▣ 리스트 생성 및 기본 연산

- range: 순차적인 정수 리스트 만들기

```
print range(10) # 0(Included)부터 10(Excluded)까지
```

```
print range(5, 15) # 5(Included)부터 15(Excluded)까지
```

```
print range(5, 15, 2) # 0(Included)부터 10(Excluded)까지, Step: 2
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]  
[5, 7, 9, 11, 13]
```

- range가 반환하는 값이 리스트

■ 리스트 생성 및 기본 연산

```
for el in range(10):  
    print el, 'inch=', el * 2.54, 'centi'
```

```
0 inch= 0.0 centi  
1 inch= 2.54 centi  
2 inch= 5.08 centi  
3 inch= 7.62 centi  
4 inch= 10.16 centi  
5 inch= 12.7 centi  
6 inch= 15.24 centi  
7 inch= 17.78 centi  
8 inch= 20.32 centi  
9 inch= 22.86 centi
```

- `range(10) = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`

```
sun, mon, tue, wed, thu, fri, sat = range(7)  
print sun, mon, tue, wed, thu, fri, sat
```

```
0 1 2 3 4 5 6
```

- `range(7) = [0, 1, 2, 3, 4, 5, 6, 7]`

▣ 리스트 생성 및 기본 연산

•리스트 안의 각 자료가 튜플일 때 for 문을 사용하여 값 추출 방법

```
lt = [('one', 1), ('two', 2), ('three', 3)]  
for t in lt:  
    print 'name =', t[0] ,', num =', t[1]
```

```
name = one , num = 1  
name = two , num = 2  
name = three , num = 3
```

- 리스트 안 자료가 튜플일 때 이를 추출하는 방법
- t에 들어가는 것은 각각의 튜플
- t[0] = 'one' , t[1] = '2'

▣ 리스트 생성 및 기본 연산

- 아래 코드가 더 효율적

```
lt = [('one', 1), ('two', 2), ('three', 3)]
for t in lt:
    print 'name = %s, num = %s' % t
```

```
name = one, num = 1
name = two, num = 2
name = three, num = 3
```

- %s → 문자열로 포매팅

- for 문의 헤더에서 각 튜플의 값 추출

```
lt = [('one', 1), ('two', 2), ('three', 3)]
for name, num in lt:
    print name, num
```

```
one 1
two 2
three 3
```

- for 뒤에 element 변수가 그대로 쓰임

▣ 리스트 생성 및 기본 연산

- 리스트 안의 각 자료가 리스트여도 for 문의 헤더에서 동일하게 값 추출 가능

```
LL = [['one', 1], ['two', 2], ['three', 3]]  
for name, num in LL:  
    print name, num
```

```
one 1  
two 2  
three 3
```

- for 문은 in 뒤에 튜플이어도 되고 리스트여도 됨