파이썬 프로그래밍

문자열 정의 및 기초 연산

한국기술교육대학교
온라인평생교육원

▣ 시퀀스 자료형의 지원 연산

1. 시퀀스 자료형이란?

- 저장된 각 요소를 정수 Index를 이용하여 참조가 가능한 자료형
- 시퀀스(Sequence) 자료형: 문자열, 리스트, 튜플

▪문자열: 시퀀스 자료형의 대표적인 자료형
▪시퀀스 자료형이 가지고 있는 공통적인 연산 존재

```
s = 'abcdef'
L = [100,200,300]
t = ('tuple', 'object', 1, 2)
```

- 시퀀스 자료형이 가지는 공통적인 연산⊠인덱싱 (Indexing)
  - 슬라이싱 (Slicing)
  - 확장 슬라이싱 (Extended Slicing)
  - 연결 (Concatenation)
  - 반복 (Repitition)
  - 멤버쉽 테스트 (Membership Test)
  - 길이 정보 (Length)
  - for ~ in 문

▣ 시퀀스 자료형의 지원 연산

2. 인덱싱

```
s = 'abcdef'
l = [100, 200, 300]
print s[0]
print s[1]
print s[-1]
print
print l[1]
l[1] = 900
print l[1]
```

```
a
b
f

200
900
```

- s= 문자열, l = 리스트가 할당
- 숫자는 반드시 하나의 정수
- [하나의 정수] = 인덱싱 연산
- s[0] = 0번째 해당하는 문자 = a
- s[1] = 1에 해당하는 인덱스 = b
- s[-1] = 맨 마지막 인덱스 = f
- Print (아무것도 없음): 한 줄 띄우기
- l[1] = 두 번째 해당하는 인덱스 = 200
- list(리스트): 변경이 가능한 자료형

▣ 시퀀스 자료형의 지원 연산

2. 인덱싱

```
print l[100]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-2-dd85adc9a089> in <module>()
----> 1 print l[100]

IndexError: list index out of range
```

- l [100] → list l에 존재하지 않음 → error 발생
- error의 종류: Index Error

▣ 시퀀스 자료형의 지원 연산

3. 슬라이싱

• L[start:end]: start는 inclusive, end는 exclusive

```
s = 'abcdef'
L = [100, 200, 300]

print s[1:3]
print s[1:]
print s[:]
print s[-100:100]
print
print L[:-1]      # L[:2] 와 동일
print L[:2]
```

```
bc
bcdef
abcdef
abcdef

[100, 200]
[100, 200]
```

▪슬라이싱: '[ ]'가 쓰이고 안쪽에 반드시 ':(콜론)'이 쓰임
▪시퀀스 자료형의 일부분을 가져옴
▪s[1:3] = 1은 start에 해당→ b
▪s[1:3] = 3은 마지막 인덱스로 포함하지 않음
▪c → '2'에 해당하는 인덱스
▪s[1:3] = 1과 2에 해당하는 문자열만 반환
▪start 값 O stop값 X → 해당 문자열의 마지막까지 슬라이싱 됨
▪start 값 X stop값 X → 전체를 다 슬라이싱
▪-100 → 처음 값, 100 → 마지막 값
▪L[:-1] → 끝 인덱스 → 300 → '2'에 해당
▪L[:-1] = L[:2]

▣ 시퀀스 자료형의 지원 연산

4. 확장 슬라이싱

• L[start:end:step]: 인덱싱되어지는 각 원소들 사이의 거리가 인덱스 기준으로 step 만큼 떨어짐

```
s = 'abcd'
print s[::2]    #step:2 - 각 원소들 사이의 거리가 인덱스 기준으로 2가 됨
print s[::-1]   #step:-1 - 왼쪽 방향으로 1칸씩
```

```
ac
dcba
```

▪확장 슬라이싱: 콜론(:)이 2개가 쓰임
▪s[::2] → '2'는 step에 해당
▪2[::] → 전체 내용을 다 들고 오는 것
▪s[::2] → 들고 나오는 문자에 해당하는 인덱스의 차이가 2가 됨
▪a의 인덱스: 0, c의 인덱스: 2 → 2-0=2
▪스탭에 해당하는 숫자만큼 인덱스에 차이를 두어 반환
▪-1 → 역순으로 값을 가지고 옴
▪d의 인덱스: 3, c의 인덱스: 2 → 2-3=-1
▪c의 인덱스: 2, b의 인덱스: 1 → 1-2=-1

▣ 시퀀스 자료형의 지원 연산

5. 연결하기

```
s = 'abc' + 'def'
print s

L = [1,2,3] + [4,5,6]
print L
```

```
abcdef
[1, 2, 3, 4, 5, 6]
```

▪시퀀스 + 시퀀스 → 2개의 시퀀스를 연결해서 하나로

▪리스트 + 리스트 → 하나의 리스트로 반환

▣ 시퀀스 자료형의 지원 연산

6. 반복하기

```
s = 'abc'
print s * 4

L = [1,2,3]
print L * 2
```

```
abcabcabcabc
[1, 2, 3, 1, 2, 3]
```

▪반복하기: '곱하기' 연산을 보는 것

▪s * 4= s의 내용을 4번 반복하여 반환

▣ 시퀀스 자료형의 지원 연산

7. 멤버십 테스트

```
s = 'abcde'
print 'c' in s

t = (1,2,3,4,5)
print 2 in t
print 10 in t
print 10 not in t
```

```
True
True
False
True
```

▪'c' in s : c라고 하는 문자열이 s 안에 존재하는지 확인
▪True 아니면 False로 반환
▪10 not in t : t 안에 10이 존재하지 않는지 확인

```
print 'ab' in 'abcd'
print 'ad' in 'abcd'
print ' ' in 'abcd'
print ' ' in 'abcd '
```

```
True
False
False
True
```

▪ad(연속된 문자)는 'abcd'에 존재하지 않음
▪'(공백)' 은 'abcd'에 존재하지 않음
▪'abcd "는 'abcd(공백)'으로 false가 아닌 true

▣ 시퀀스 자료형의 지원 연산

8. 길이 정보

```
s = 'abcde'
l = [1,2,3]
t = (1, 2, 3, 4)
print len(s)
print len(l)
print len(t)
```

```
5
3
4
```

▪문자열, 리스트, 튜플 모두 len 함수의 인자가 될 수 있음

▣ 시퀀스 자료형의 지원 연산

9. for~in 문

```
for c in 'abcd':
    print c,
```

```
a b c d
```

▪컨테이너 종류: 시퀀스 자료형, 시퀀스 자료형이 아닌 것
▪시퀀스 자료형 = 문자열, 리스트, 튜플
▪시퀀스 자료형이 아닌 것 = 사전