파이썬 프로그래밍

수치형 자료형,
문자열 자료형

한국기술교육대학교
온라인평생교육원

▣ 문자열

1. 문자열 형식

```
print 'Hello World!'
print "Hello World!"
```

```
Hello World!
Hello World!
```

- 문자열을 만들 때에는 단일 따옴표 또는 이중 따옴표를 사용
- 두 따옴표는 사용 시 차이 없으므로 선택적으로 사용 가능
- 두 따옴표를 모두 사용할 수 있도록 만든 이유는?
- Hello “World”를 출력하려면 단일 따옴표로 문자열 생성
- Hello ‘World’를 출력하려면 이중 따옴표로 문자열 생성

▣ 문자열

1. 문자열 형식

```
multiline = '''
To be, or not to be
that is the question
'''
print multiline

multiline2 = """
To be, or not to be
that is the question
"""
print multiline2
```

```
To be, or not to be
that is the question

To be, or not to be
that is the question
```

- 여러 줄의 문자열을 만들 때에는 ''' ''' 또는 """ """ 사용
- 첫 번째 ''' 뒤에 개행 문자가 있음
- 문자열 첫 번째 줄 바꿈이 된 이유는 개행 문자 때문
- 마지막 ''' 앞에 개행 문자가 있음
- 문자열 마지막 줄 바꿈이 된 이유도 개행 문자 때문
- 이중 따옴표 세 개로 만든 여러 줄의 문자열
- 마찬가지로 개행 문자가 처음과 끝에 있음

▣ 문자열

2. 인덱싱과 슬라이싱

인덱싱

```
s = "Hello world!"
print s[0]
print s[1]
print s[-1]
print s[-2]
```

```
H
e
!
d
```

- 문자열은 시퀀스(순차자료)형으로 인덱스가 붙음
-

문자	H	e	l	l	o	(공백)	w	o	r	l	d	!
인덱스	0	1	2	3	4	5	6	7	8	9	10	11

- 인덱싱: 꺽쇠괄호([]) 안에 인덱스 번호 입력
- print s[0] → 문자열 s에서 인덱스 0인 문자(H) 출력
- print s[1] → 문자열 s에서 인덱스 1인 문자(e) 출력
- print s[11] → 문자열 s에서 인덱스 11인 문자(!) 출력
- 인덱스 앞에 '–(마이너스)' 붙이면 뒤에서부터 셈

▣ 문자열

2. 인덱싱과 슬라이싱

슬라이싱

```
s = "Hello world!"
print s[1:3]
print s[0:5]
```

```
el
Hello
```

- 슬라이싱: 꺽쇠괄호([]) 안에 인덱스 번호와 콜론(:) 입력
- [시작할 인덱스(포함) : 끝낼 인덱스(제외) : 스텝]
- S[1:3] → 인덱스 1인 문자(포함)부터 3인 문자(제외)까지
- 슬라이싱이란 자르기
- [1:3] → 인덱스 1(e)부터 인덱스 3 앞(l)까지 자르기
- 인덱스 3(두 번째 l)은 제외, 인덱스 2(첫 번째 l)까지만 출력
- [0:5] → 인덱스 0(H)부터 인덱스 5 앞(o)까지 자르기

▣ 문자열

2. 인덱싱과 슬라이싱

슬라이싱

```
s = 'Hello'
print s[1:]
print s[:3]
print s[:]
```

```
ello
Hel
Hello
```

- 콜론(:)만 있고 값이 없을 경우 → 기본값으로 사용
- [1:] → 인덱스 1(H)부터 마지막(자료형의 크기)까지
- 자료형의 크기는 5 → [1:]는 [1:5]와 같음
- [1:5] → 인덱스 1(e)부터 인덱스 5앞(o)까지 자르기
- [:3] → 처음(인덱스 0)부터 인덱스 3앞(l)까지
- [:] → 전체 문자열(슬라이싱 하지 않음)

▣ 문자열

2. 인덱싱과 슬라이싱

슬라이싱

```
s = 'abcd'
print s[::2]
print s[::-1]
```

```
ac
dcba
```

- [::값] → 해당 값으로 스텝
- H(인덱스 1)와 e(인덱스 2)의 차이는 1 ← 이 차이가 스텝
- 기본적인 인덱스의 차이(스텝)는 1
- 스텝 2 → 인덱스 차이가 2씩 나도록 슬라이싱
- [::2] → 처음부터 마지막까지 인덱스 차이가 2씩 나도록 슬라이싱
- 인덱스 0(a)을 가져온 다음 인덱스 2(c)를 가져옴
- 인덱스 2(c)를 가져온 다음 인덱스 4를 가져옴(없으므로 끝)
- 스텝의 값만큼 문자의 인덱스 차이가 나도록 슬라이싱
- 스텝의 값이 마이너스 → 인덱스의 차이가 마이너스
- 문자열을 거꾸로 가져오게 됨
- 2(c의 인덱스) - 3(d의 인덱스) = -1
- 1(b의 인덱스) - 2(c의 인덱스) = -1
- [::-1] → 문자열을 거꾸로 뒤집어 가져옴

▣ 문자열

2. 인덱싱과 슬라이싱

슬라이싱

```
s = 'Hello World'
s[0] = 'h'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-88-28020f3d59a5> in <module>()
      1 s = 'Hello World'
----> 2 s[0] = 'h'

TypeError: 'str' object does not support item assignment
```

- 문자열 자료형은 내부 내용 변경 불가능
- s[0] = h → 문자열 s의 인덱스 0을 h로 변경한다는 의미
- s[0] = h → 수행해보면 에러 발생
- 문자열 내의 문자를 바꿀 수 없으므로 주의

▣ 문자열

2. 인덱싱과 슬라이싱

슬라이싱

```
s = 'Hello World'
s = 'h' + s[1:]
s
```

```
'hello World'
```

- 문자열을 바꾸기 위해서는 슬라이싱을 활용
- 'h' + s[1:] → 문자 h와 인덱스 1부터 슬라이싱한 결과를 합함
- 'h' + s[1:] = h + ello World = hello World
- 문자열의 기존 내용이 바뀐 것이 아님
- 새로운 문자열이 구성되어 할당된 것
- 따라서 기존 문자열은 쓰레기(garbage)

▣ 문자열

3. 문자열 연산

```
print 'Hello' + '' + 'World'
print 'Hello' * 3
print '-' * 60
```

```
HelloWorld
HelloHelloHello
------------------------------------------------------------
```

- 문자열 + 문자열 → 순서 그대로 두 문자열 연결
- print 'Hello' + ''(공백없음) + 'World' → HelloWorld
- 문자열 * 숫자 → 숫자만큼 문자열 반복
- print 'Hello' * 3 → HelloHelloHello
- print '-' * 60 → -가 60번 반복되어 나타남

▣ 문자열

4. 문자열의 길이

```
s = 'Hello World'
len(s)
```

```
11
```

- len(s) → s의 문자열 길이를 의미
- 주의! 이클립스에서는 print를 반드시 추가해야 결과값 출력됨

▣ 문자열

5. 문자열내 포함 관계 여부

```
s = 'Hello World'
print 'World' in s
print 'World' not in s
```

```
True
False
```

- 'A' in B → B 안에 A가 있다(결과는 True or False)
- print 'World' in s → 문자열 s 안에 World가 있다 → True
- 'A' not in B → B 안에 A가 없다(결과는 True or False)
- print 'World!' in s → 문자열 s 안에 World!가 있다 → False
- print ' ' in s → 문자열 s 안에 공백이 있다 → True