파이썬 프로그래밍

문자열 메소드와 포멧팅

한국기술교육대학교
온라인평생교육원

▣ 문자열 메소드

```
s = 'i like programming.'
print s.upper()
print s.upper().lower()
print 'I Like Programming'.swapcase()  # 대문자는 소문자로, 소문자는 대문자로 변환
print s.capitalize() # 첫 문자를 대문자로 변환
print s.title()      # 각 단어의 첫 문자를 대문자로 변환
```

```
I LIKE PROGRAMMING.
i like programming.
i lIKE pROGRAMMING
I like programming.
I Like Programming.
```

▪s.upper() → 안 내용 전부 대문자로 변경

▪s.upper()의 결과를 lower() → 전체 결과를 전부 소문자로 변경

▪swapcase() → case를 바꾸는 것

▪case → 대문자, 소문자를 나타내는 것

▪s.capitalize() → 첫 문자만 대문자로 변환

▪s.title() → 각 단어의 첫 문자가 대문자로 변환

▪문자열은 변경을 할 수가 없음

▪메소드들이 새로운 문자열을 반환함

▣ 문자열 메소드

```
s = 'i like programming, i like swimming.'
print s.count('like') # 'like' 문자열이 출현한 횟수를 반환
print
print s.find('like') # 'like'의 첫글자 위치 (offset)를 반환
print s.find('programming') # 'programming'의 첫글자 위치를 반환
print s.find('programmin') # 'programmin'의 첫글자 위치를 반환
print s.find('programmii') # 'programmii' 단어는 없기 때문에 -1 반환
print
print s.find('like', 3) # offset=3 부터 'like'을 검색하여 'like'의 첫글자 위치 반환
print s.find('my') # 'my' 단어는 없기 때문에 -1 반환
```

```
2

2
7
7
-1

22
-1
```

▪s.count('like') → s 안에 like라는 문자열의 등장 횟수
▪s.find('like') → s안 like 문자의 인덱스 반환 → 2
▪s.find('programii') → s안 programii 문자가 없으므로 -1이 반환
▪s.find('like' * 3) → 3 인덱스부터 찾아 like 인덱스 반환

▣ 문자열 메소드

```
s = 'i like programming, i like swimming.'
print s.startswith('i like') # 'i like'로 시작하는 문자열인지 판단
print s.startswith('I like') # 대소문자 구별
print

print s.endswith('swimming.') # 'swimming.'로 끝나는 문자열인지 판단
print s.startswith('progr', 7) # 7번째 문자열이 'progr'로 시작하는지 판단
print
```

```
True
False

True
True
```

▪s.startswith() → true, false를 반환

▪대, 소문자를 정확히 반영함

▪s.startswith('progr', 7) → 7번째 인덱스부터 시작하여 progr 찾음

▣ 문자열 메소드

```
u = ' spam and ham            '
print u.strip() # 좌우 공백을 제거하여 새로운 스트링 생성
print u # 스트링은 변경불가능
y = u.strip() # strip()는 새로운 스트링을 생성함
print y
print


print u.rstrip() # 오른쪽 공백 제거
print u.lstrip() # 왼쪽 공백 제거
print '      abc '.strip()
print '><><abc<><><>'.strip('<>') # 인자로 주어진 스트링 안에 지정된 모든 문자를
                                    좌우에서 제거
```

```
spam and ham
 spam and ham
spam and ham

 spam and ham
spam and ham
abc
abc
```

- u.strip() → u 문자열 앞 뒤의 공백 제거
- u.rstrip() → 오른쪽 공백 제거
- u.lstrip() → 왼쪽 공백 제거
- u 대신 문자열을 직접 넣어 strip 메소드 사용 가능
- '><>abc<>'.strip('<>') → <, >를 문자열 앞뒤에서 모두 제거

▣ 문자열 메소드

```
p = '  \t abc  \t '
print p
print p.strip() #  \t도 공백문자이므로 제거됨
```

```
                    abc
abc
```

▪ \ t 도 공백문자로 인식되어 제거됨

```
u = 'spam and ham'
print u.replace('spam', 'spam, egg') # replace()는 새로운 스트링을 생성함
print u
```

```
spam, egg and ham
spam and ham
```

▪u.replace(a,b) → u 안의 a 문자를 b문자로 대치

▣ 문자열 메소드

```
u = ' spam and ham      '
print u.split() # 공백으로 분리 (모든 공백 제거 및 문자열 내의 단어 리스트를 얻을 수
                                    있음)
print u.split('and') # 'and'로 분리
print

u2 = 'spam and ham \ tegg \ ncheese'
print u2.split()
```

```
['spam', 'and', 'ham']
[' spam ', ' ham      ']

['spam', 'and', 'ham', 'egg', 'cheese']
```

▪split은 공백을 기준으로 분류함

▪u.split() → 각 단어가 공백을 기준으로 분리되어 리스트로 반환

▪u.split('and') → and를 기준으로 분리

▪ \ t, \ n 도 공백으로 분류

▣ 문자열 메소드

```
u = 'spam ham \ tegg \ ncheese'
t = u.split() # 문자열 내의 단어 리스트
print t
print
t2 = ':'.join(t) # 리스트 t 내부의 각 원소들을 ':'로 연결한 문자열 반환
print type(t2)
print t2
print
t3 = ','.join(t) # 리스트 t 내부의 각 원소들을 ','으로 연결한 문자열 반환
print t3
print
t4 = ' \ n'.join(t) # 리스트 t 내부의 각 원소들을 ' \ n'으로 연결한 문자열 반환
print t4
print
```

```
['spam', 'ham', 'egg', 'cheese']

<type 'str'>
spam:ham:egg:cheese

spam,ham,egg,cheese

spam
ham
egg
cheese
```

▪join은 split의 정반대에 해다하는 메소드
▪(문자열).join(t) → 문자열이 리스트 t 내부의 각 원소 가운데 연결함
▪ \ n 이 개행이므로 개행으로 연결됨

▣ 문자열 메소드

```
u2 = u"스팸 햄 계란 치즈"
t2 = u2.split()
print t2
print t2[0], t2[1], t2[2], t2[3]
```

```
[u' \ uc2a4 \ ud338', u' \ ud584', u' \ uacc4 \ ub780', u' \ uce58 \ uc988']
스팸 햄 계란 치즈
```

▪유니코드인 경우 각각의 문자열을 지정하여 사용

```
lines = '''first line
second line
third line'''
print type(lines)
lines2 = lines.splitlines() # 문자열을 라인 단위로 분리한 각 원소들을 지닌 리스트 반환
print type(lines2)
print lines2
```

```
<type 'str'>
<type 'list'>
['first line', 'second line', 'third line']
```

▪lines라는 문자열이 여러 개의 문자열로 이루어짐

▪splitlines() →line기준으로 분리하여 리스트 반환

▣ 문자열 메소드

```
u = 'spam and egg'
c = u.center(60) # 60자리를 확보하되 기존 문자열을 가운데 정렬한 새로운 문자열 반환
print type(c)
print c
print u.ljust(60) # 60자리를 확보하되 기존 문자열을 왼쪽 정렬한 새로운 문자열 반환
print u.rjust(60) # 60자리를 확보하되 기존 문자열을 오른쪽 정렬한 새로운 문자열 반환
```

```
<type 'str'>
                    spam and egg
spam and egg
                                              spam and egg
```

▪u.center(60) → 60자리를 확보하되, 기존 문자열을 가운데 정렬
▪u.ljust(60) → 60자리를 확보하되, 기존 문자열을 왼쪽 정렬
▪u.rjust(60) → 60자리를 확보하되, 기존 문자열을 오른쪽 정렬

```
u = 'spam and egg'
print u.center(60, '-') # 공백에 채워질 문자를 선택할 수 있음
print u.ljust(60, '-')
print u.rjust(60, '-')
```

```
------------------------spam and egg------------------------
spam and egg----------------------------------------------
----------------------------------------------spam and egg
```

▪u.center(60, '-') → 확보한 60자리 중 공백에 '-' 문자 채움

▣ 문자열 메소드

```
print '1234'.isdigit() # 문자열 내의 Character들이 모두 숫자인가?
print 'abcd'.isalpha() # 문자열 내의 Character들이 모두 영문자인가?
print '1abc234'.isalnum() # 문자열 내의 Character들이 모두 영문자 또는 숫자인가?
print 'abc'.islower() # 문자열 내의 Character들이 모두 소문자인가?
print 'ABC'.isupper() # 문자열 내의 Character들이 모두 대문자인가?
print ' \ t \ r \ n'.isspace() # 문자열 내의 Character들이 모두 공백 문자인가?
print 'This Is A Title'.istitle() # 문자열이 Title 형식 (각 단어의 첫글자가 대문자)인가?
```

```
True
True
True
True
True
True
True
True
```

▪is가 들어가는 메소드 → 반드시 true, false로 반환
▪isdigit() → 문자열 내 character가 모두 숫자인가?
▪isalpha() → 문자열 내 character가 모두 영문자인가?
▪isalnum() → 문자열 내 character가 모두 영문자 또는 숫자인가?
▪islower() → 소문자? isupper() → 대문자?
▪isspace() → 문자열 내 character이 모두 공백문자인가?
▪istitle() → 문자열이 title 형식(각 단어의 첫글자가 대문자) 인가?

```
s = '123'
print s.zfill(5) # 5글자 자리 확보뒤 문자열을 쓰되 남는 공백에는 zero (0)를 채움
print 'goofy'.zfill(6) # 6글자 자리 확보뒤 ...
```

```
00123
0goofy
```

▪s.zfill(5) → 5글자 자리 확보 후 문자열 쓰고, 남은 공백에 0 채움

파이썬 프로그래밍

문자열 메소드와 포멧팅

한국기술교육대학교
온라인평생교육원

▣ 문자열 포매팅

1. 튜플을 이용한 포매팅

1) 문자열 변환

- 포맷팅 문자 : 문자열 내에 존재하는 %
- 포캣팅을 활용한 문자열 변환 : 포맷팅 문자를 포함하는 문자열 % 튜플

```
print 'name = %s, age = %s' % ('gslee', '24')
```

```
name = gslee, age = 24
```

▪문자열 구성 시 문자 내 % 사용 → 포멧팅 문자를 지칭하는 방법
▪문자열 + % + 튜플
▪문자열에 %가 두 개 → 튜플이 2개
▪튜플 첫번째 요소가 앞 %s, 두번째 요소가 뒤 %s에 들어감
▪자리를 잡아주면서 %s에 들어오는 것이 문자열임을 알려줌

▣ 문자열 포매팅

1. 튜플을 이용한 포매팅

```
letter = '''
안녕하세요 %s님,

오늘 밤 파티에 참석해 주실 수 있나요?

그럼..

이강성 드림'''
name = '홍길동'
print letter % name
print
names = ['한학신', '정인숙', '박미경']
for name in names:
    print letter % name
    print '-' * 40
    print
```

▣ 문자열 포매팅

1. 튜플을 이용한 포매팅

안녕하세요 홍길동님,

오늘 밤 파티에 참석해 주실 수 있나요?

그럼..

이강성 드림

안녕하세요 한학신님,

오늘 밤 파티에 참석해 주실 수 있나요?

그럼..

이강성 드림

안녕하세요 정인숙님,

오늘 밤 파티에 참석해 주실 수 있나요?

그럼..

이강성 드림

안녕하세요 박미경님,

오늘 밤 파티에 참석해 주실 수 있나요?

그럼..

이강성 드림

▪'-' * 40 → -라는 문자가 40번 반복하여 출력됨

▣ 문자열 포매팅

1. 튜플을 이용한 포매팅

2) 숫자 변환

포맷팅 문자	설명
%s	문자열을 포함한 임의의 객체를 문자열로 변환하여 출력 (str() 내장 함수 사용)
%r	문자열을 포함한 임의의 객체를 문자열로 변환하여 출력 (repr() 내장 함수 사용)
%c	1글자 문자 (ex. '%c' % 'k)
%d	10진 정수 (%5d: 5자리를 확보한 후 정수 포맷팅)
%i	%d와 동일
%u	부호 없는 정수. 음수는 양수처럼 해석함 (ex. '%u' % -12 --> '4294967284')
%o	8진수 정수 (ex. '%o' % 13 --> 15)
%x	16진수 정수 (소문자 표현) (ex. '%x' % 13 --> 'd')
%X	16진수 정수 (대문자 표현) (ex. '%X' % 13 --> 'D')
%e	부동 소수점 실수를 지수 형태로 표현 (%.2e: 2자리는 소수점 이하 자리수)
%E	%e 와 동일 (대문자 E 표현)
%f	부동 소수점 실수 (%5.2f: 소수점 포함 총 5자리 확보한 후 2자리는 소수점 이하 자리수)
%g	부동 소수점을 편의에 따라 일반 실수 형식이나 지수 형식으로 변환
%G	%g와 동일 (대문자 E 표현)

- %s는 %뒤에 문자열일 경우 사용
- str() → 임의의 객체를 문자열로 반환
- str() , repr()→ 모두 문자열로 반환해주는 함수
- %d → 앞에 자리를 확보하기 위해 %5d로 사용 가능 → 5자리 확보
- % 뒤 실수가 들어가는 경우 → %f
- %5.2f → 소수점 포함 총 5자리 확보한 후 소수점 이하는 2자리 사용
- %5.2f 에서 5가 전체자리수, 2가 소수점 이하 자리수
- %s, %r, %c, %d, %f

▣ 문자열 포매팅

1. 튜플을 이용한 포매팅

```
print "%s -- %s -- %d -- %f -- %e" % ((1, 2), [3,4,5], 5, 5.3, 101.3)
```

```
(1, 2) -- [3, 4, 5] -- 5 -- 5.300000 -- 1.013000e+02
```

▪'%s→ (1,2)를 str함수로 불러서 문자화 함
▪두 번째 %s → [3,4,5]
▪%d → 5
▪%f → 5.3
▪%e → 101.2 → 지수표현방식으로 표현

```
print "%3d -- %5.2f -- %.2e" % (5, 5.356, 101.3)
```

```
5 --  5.36 -- 1.01e+02
```

▪%3d → 5 → 3개의 자리로 확보한 다음 5 숫자 입력
▪%5.2f → 5개 자리 확보 후 소수점 이하 2자리 → 5.36
▪%5.2f의 f는 자리수를 맞춰주기 위해 반올림이 됨
▪%e → 지수형태의 표현 방식 (많이 활용되지 않음)
▪%.2e → .2가 있으므로 소수점 뒤에 01이 2에 맵핑
▪101.3이 지수형태로 찍힌 것

```
a = 456
print '%d -- %o -- %x -- %X' % (a, a, a, a)
```

```
456 -- 710 -- 1c8 -- 1C8
```

▪%o → 8진법에 해당하는 수
▪%x → 16진수
▪%X → 같이 사용하는 알파벳 문자를 대문자로 변경

▣ 문자열 포매팅

2. 사전(Dictionary)을 이용한 포매팅

```
print '%(이름)s -- %(전화번호)s' %{'이름':'홍길동', '전화번호':5284}
print '%(이름)s -- %(전화번호)s' %{'전화번호':5284, '이름':'홍길동'}
print '%(이름)s -- %(전화번호)s' %{'전화번호':5284, '이름':'홍길동', '주소':'Seoul'}
```

```
홍길동 -- 5284
홍길동 -- 5284
홍길동 -- 5284
```

▪사전형태이기 때문에 순서가 바뀌어도 괜찮음

▪주소 항목은 % 앞에 없기 때문에 무시