

Section01 문자열의 이해

- 문자열과 메모리 공간
 - 컴퓨터 공학자들은 이러한 문자를 처리하기 위해 이진수로 변환되는 표준 규칙을 만들었다. ASCII, CP949, MS949, UTF-8 등 이러한 규칙을 인코딩(encoding)이라고 한다.
 - ① 컴퓨터는 문자를 직접 인식하지 못한다.
 - ② 컴퓨터는 문자를 숫자로 변환하여 인식한다.
 - ③ 사람들은 문자를 숫자로 변환하기 위한 규칙을 만들었다.
 - ④ 일반적으로 이 규칙은 1개의 영문자를 1바이트, 즉 2의 8승(28) 정도의 공간에 저장될 수 있 도록 정하였다.

Section01 문자열의 이해

- 문자열과 메모리 공간

 - 다음과 같은 코드로 문자열이 저장된 공간의 크기를 확인할 수 있다.

```
>>> import sys # sys 모듈을 호출
>>> print(sys.getsizeof("a"), sys.getsizeof("abc"))
50 51 52 # "a", "ab", "abc" 각각의 메모리 크기 출력
```

 컴퓨터에 a라고 알려 줘도 컴퓨터는 정확히 a라는 텍스트를 인식하는 것이 아니다. 대신 컴 퓨터는 이 정보를 이진수로 변환하여 저장한다.

2/2

Section01 문자열의 이해

■ 문자열과 메모리 공간

```
032 sp 048 0 064 9
033 ! 049 1 065 A
034 " 050 2 066 B
035 # 051 3 067 C
                 016 ▶ (dle)
      (nul)
001 @ (soh)
                 017 ◀ (dc1)
                                                                 081 Q
002 • (stx)
003 • (etx)
                 018 : (dc2)
                                                                082 R
083 S
                                                                          098 b
                                                                                    114 r
                 019 # (dc3)
                                                                          099 c
                                                                                    115 s
004 + (eot)
                 020 ¶ (dc4)
                                  036 $
                                             052 4
                                                      068 D
                                                                 084 T
                                                                                    116 t
005 & (eng)
                                  037 %
                                             053 5
                                                       069 E
                                 038 &
                                            054 6 070 F
055 7 071 G
006 • (ack)
007 • (bel)
                 022 - (syn)
                                                                 086 V
                                                                          102 f
103 g
                                                                                    118 v
                023 : (etb)
                                                                 087 W
                                                                                    119 w
                                            056 8 072 H
057 9 073 I
058 : 074 J
008 a (bs)
                 024 † (can)
                                  040 (
009 (tab)
010 (lf)
                                 041 )
                 025 ¿ (em)
                                                                 089 Y
                                                                                    121 y
                                                                090 2
                                                                          106 i
                 026 (eof)
                                                                                    122 z
011 & (vt)
                 027 + (esc)
                                 043 +
                                             059 ; 075 K 091 [
                                                                          107 k
                                                                                    123
                                 044 ,
045 -
                 028 L (fs)
                                             060 <
                                 045 - 061 = 077 M 093 ] 109 m 125 ]
046 . 062 > 078 N 094 ^ 110 n 126 ~
047 / 063 ? 079 0 095 _ 111 o 127 o
013 (cr)
                029 + (gs)
014 # (30)
014 $ (so) 030 ▲ (rs)
015 $ (si) 031 ▼ (us)
```

[UTF-8의 유니코드(출처: Nicolas Bouliane)]

4/25



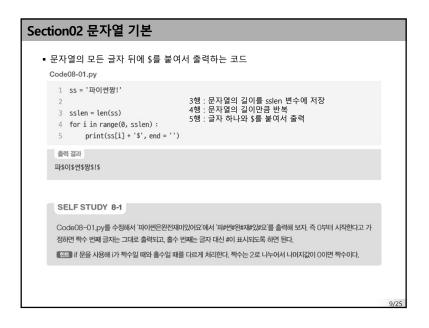
```
Section02 문자열 기본
■ 문자열의 개념
   ■ 리스트 코드와 비교 : 리스트는 대괄호 □로 묶고 문자열은 작은따옴표로 묶어 출력
     aa = [10, 20, 30, 40, 50]
     aa[1:3]
      aa[3:]
     출력 결과
     [20, 30]
     [40, 50]
     ss = "파이썬최고"
      ss[0]
      ss[1:3]
      ss[3:]
      출력 결과
      '파'
      '이썬'
      '최고'
```

```
Section02 문자열 기본

• len() 함수 : 리스트나 문자열의 개수를 셀 때 사용

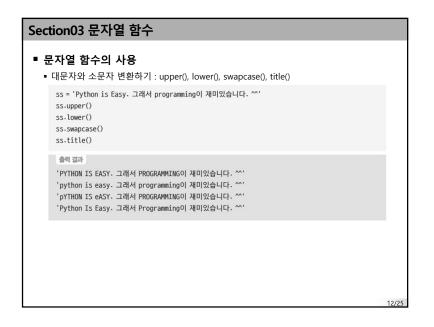
ss = '파이썬abcd'
len(ss)

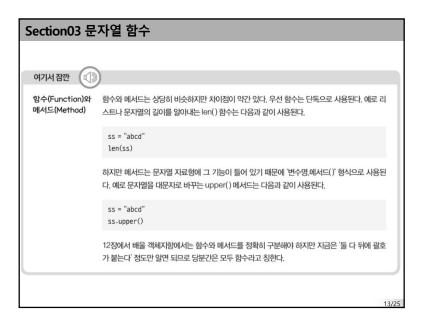
출력결과
7
```



Section03 문자열 함수 ■ 문자열의 연산 함수명 기능 좌우 공백 삭제 strip() 오른쪽 공백 삭제 rstrip() 왼쪽 공백 삭제 Istrip() 문자열을 공백이나 다른 문자로 나누어 리스트로 반환 split() isdigit() 문자열이 숫자인지 여부 반환 문자열이 소문자인지 여부 반환 islower() 문자열이 대문자인지 여부 반환 isupper()

Section03 문자열 함수 ■ 문자열의 연산 함수명 len() 문자열의 문자 개수를 반환 upper() 대문자로 변환 lower() 소문자로 변환 각 단어의 앞글지만 대문자로 변환 title() 첫 문자를 대문자로 변환 capitalize() count('찾을 문자열') '찾을 문자열'이 몇 개 들어 있는지 개수 반환 '찾을 문자열'이 왼쪽 끝부터 시작하여 몇 번째에 있는지 반환 find('찾을 문자열') find() 함수와 반대로 '찾을 문자열'이 오른쪽 끝부터 시작하여 몇 번째에 있는지 반환 rfind('찾을 문자열') '찾을 문자열'로 시작하는지 여부 반환 startswith('찾을 문자열') '찾을 문자열'로 끝나는지 여부 반환 endswith('찾을 문자열')





```
Section03 문자열 함수
   ■ 문자열이 괄호로 감싸 있지 않으면 괄호로 감싸 주는 프로그램
     Code08-03.py
      1 ss = input("입력 문자열 ==> ")
                                   1행 : 문자열 입력
       2 print("출력 문자열 ==> ", end = '')
                                    4행 : 문자열의 시작이 (가 아니면 (를 우선 출력
                                    7행 : 입력한 문자열을 그대로 출력
      4 if ss.startswith('(') == False :
                                   9행 : 문자열의 끝이 )가 아니면 )를 우선 출력
      5 print("(", end = '')
      7 print(ss, end = '')
      9 if ss.endswith(')') == False :
      10 print(")", end = '')
      출력 결과
      입력 문자열 ==> 파이썬 열공 중~~
      출력 문자열 ==> (파이썬 열공 중~~)
```

Section03 문자열 함수 • 문자열 찾기 : count(), find(), rfind(), index(), rindex(), startswith(), endswith() ss = '파이센 공부는 즐겁습니다. 물론 모든 공부가 다 재미있지는 않죠. ^^' ss.count('공부'), ss.rfind('공부'), ss.find('공부', 5), ss.find('업다')) print(ss.find('공부'), ss.rindex('공부'), ss.sindex('공부', 5)) print(ss.startswith('파이썬'), ss.startswith('파이썬', 10), ss.endswith('^^')) 출력 결과 2 4 21 21 -1 4 21 21 True False True

```
Section03 문자열 함수

- 문자열 공백 삭제·변경하기 : strip(), rstrip(), lstrip(), replace()

ss = ' 파 이 썬 '
ss.·strip()
ss.·lstrip()

플릭결과
'파 이 썬 '
'파 이 썬 '
'파 이 썬 '
```

```
Section03 문자열 함수

• 앞뒤의 특정 문자 삭제

ss = '----파--이---썬---'
print(ss.strip('-'))
ss = '<<파 < 이 >> 썬>>>'
print(ss.strip('◇'))

출력 결과
파---이---썬
파 < 이 >> 썬
```

```
Section03 문자열 함수

- 문자열 변경

ss = '열심히 파이썬 공부 중~'
ss.replace('파이썬', 'Python')

출력결과
'열심히 Python 공부 중~'
```

```
Section03 문자열 함수
   ■ 문자열 중간의 공백까지 삭제해 주는 코드
    Code08-04.py
    1 inStr = " 한글 Python 프로그래밍 "
     2 outStr = ""
      4 for i in range(0, len(inStr)):
      5 if inStr[i] != ' ':
           outStr += inStr[i]
     8 print("원래 문자열 ==> " + '[' + inStr + ']')
      9 print("공백 삭제 문자열 ==> " + '[' + outStr + ']')
      출력 결과
      원래 문자열 ==> [ 한글 Python 프로그래밍 ]
      공백 삭제 문자열 ==> [한글Python프로그래밍]
     SELF STUDY 8-2
      Code08-04.py를 수정해서 '(((파((이))썬)))'이 '파이썬'으로 출력되도록 해 보자.
      ● if 문이 '('이 아닐 때와 ')'이 아닐 때를 and로 연결해야 한다.
                                                                              18/25
```

```
Section03 문자열 함수

- 문자열을 입력받아 그중 o를 $로 변경하는 문자열 변경을 응용
Code08-05.py

1 ss = input("입력 문자열 =>> ") 1행 : 문자열을 입력
4행 : 입력된 문자열의 개수만큼 반복
5~8행 : 문자가 o 라면 $ 대신 출력

5 if ss[i] != 'o':
6 print(ss[i], end = '')
7 else:
8 print('$', end = '')

출력 문자열 => IT CookBook for Python
출력 문자열 => IT CSSkBSSk f$r Pyth$n

- 4~8행을 한줄로

print(ss.replace('o', '$'))
```

```
Section03 문자열 함수

- 문자열 분리·결합하기: split(), splitlines(), join()

ss = 'Python을 열심히 공부 중'
ss.split():
ss = '하나:돌:셋'
ss.split(':')
ss = '하나n돌\n셋'
ss.splitlines()
ss = '%'
ss.join('파이썬')

출력 결과
['Python을', '열심히', '공부', '중']
['하나', '돌', '셋']
'파杉이��썬'
```

```
Section03 문자열 함수

• 함수명에 대입하기: map() 함수

before = ['2019', '12', '31']
    after = list(map(int , before))
    after

출력결과
[2019, 12, 31]
```

```
Section03 문자열 함수

• 연/월/일 형식으로 문자열을 입력받아 10년 후 날짜를 출력하는 코드
Code08-06.py

1 ss = input("날짜(면/월/일) 입력 =⇒ ") 3행: 입력한 문자열을 /로 분리
따라서 ssList에 ['2019'; '12', '31'] 형식으로 분리
이행: 연도를 가리키는 문자열인 ssList[0](이 코드에서는
'2019)을 먼저 int() 함수를 사용해 정수로 변환한 후
10을 더함. 그리고 다시 str() 함수로 문자열로 변경한
print("입력한 날짜의 10년 후 =⇒ ", end = '')
print(ssList[1] + "월", end = '')
print(ssList[1] + "월", end = '')
print(ssList[2] + "일")

출력결과
날짜(면/월/일) 입력 =⇒ 2019/12/31
입력한 날짜의 10년 후 =⇒ 2029년12월31년
```

```
Section03 문자열 함수

- 문자열 정렬하기, 채우기: center(), ljust(), rjust(), zfill()

ss = '파이썬'
ss.center(10)
ss.center(10, '-')
ss.ljust(10)
ss.rjust(10)
ss.zfill(10)

- 플릭결과
- ' 파이썬 '
'---파이썬---'
'파이썬 '
' 파이썬 '
' 파이썬 '
' 아이앤'
' 아이앤'
' 아이앤'
```

