새내기 파이썬

9장 문자열과 정규식

학습 목표

- 문자열들을 어떤 구분자를 기준으로 분리할 수 있나요?
- 문자열 사이에 어떤 접착문자를 넣어서 합칠 수 있나요?
- 문자열에서 다른 문자열을 찾아서 바꿀 수 있나요?
- 문자열에서 필요 없는 문자들을 삭제할 수 있나요?
- 정규식을 사용하여서 문자열 중에서 어떤 특정한 패턴을 찾을 수 있나요?

이번 장에서 만들 프로그램

```
abc123@gmail.com는 유효한 이메일 주소입니다.
abc.cde@com는 유효하지 않은 이메일 주소입니다.
```

```
패스워드를 입력하시오: abcdef
패스워드는 최소한 8글자이어야 합니다.
패스워드를 입력하시오: abcdefrg
패스워드는 적어도 하나의 숫자를 가져야 합니다.
패스워드를 입력하시오: abbabbb1
패스워드는 적어도 대문자를 가져야 합니다.
패스워드를 입력하시오: abbbabbaA1
규정에 맞는 패스워드입니다.
```

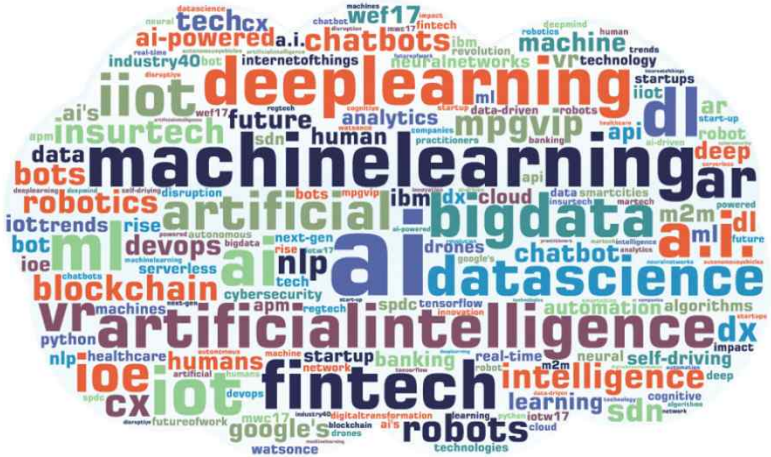
```
CPU는 무엇의 약자인가?
답안을 작성하시오(또는 quit):  Central Processing Unit
정답입니다.

제일 쉬운 프로그래밍 언어는?
답안을 작성하시오(또는 quit):  파이썬
정답입니다.
...
```

문자열 처리하기

- 컴퓨터는 근본적으로 숫자를 처리하는 기계이지만 인간을 상대하여야 하기 때문에 텍스트를 처리하는 작업도 무척 중요하다.

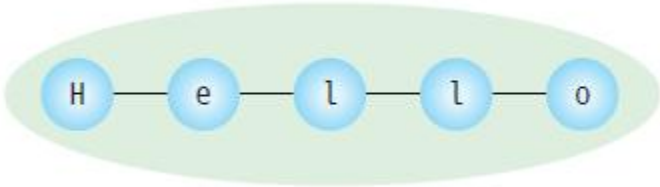
파이썬의 문자열 처리 기능

- 우리는 파이썬의 문자열 함수들을 이용하여 텍스트데이터를 수정, 보완할 수 있다. 예를 들어서 텍스트 데이터에는 대소문자가 섞여 있을 수 있고, 문자열의 앞뒤에 필요 없는 공백 문자가 붙어 있을 수도 있다.
- 파이썬은 BeautifulSoup, csv와 같은 우수한 라이브러리들도 제공하기 때문에 우리는 쉽게 텍스트를 처리하고 분석할 수 있다.



문자열

- 문자열은 문자들의 시퀀스로 정의된다. 글자들이 실(string)로 묶여 있는 것이 문자열이라고 생각하면 된다.




```
s1 = 'Hello’

s2 = "Hello“

s3 = "This is Kim's dog."          # 따옴표 안에 따옴표가 있는 경우

s = 'This is Kim\'s dog.’          # 이스케이프 문자로 작은 따옴표 표현
```

원시 문자열

- 문자열의 시작 따옴표 앞에 r을 두면 원시 문자열(raw string)이 된다.

```
>>> print(r'This is Kim\'s dog')
This is Kim\'s dog
```

- 앞에 r이 붙어 있는 원시 문자열이기 때문에 파이썬은 백슬래시를 문자열의 일부로 간주한다. 즉 백슬래시를 이스케이프 문자로 취급하지 않는다.
- 원시 문자열은 r'C:\User\Kim\Document'와 같은 윈도우 파일 경로를 나타내는 문자열에 필요하다.

인덱싱

- 문자열도 크게 보면 시퀀스(sequence)라는 자료 구조에 속한다. 시퀀스는 리스트와 같이 항목들이 순서를 가지고 모인 자료구조이다.

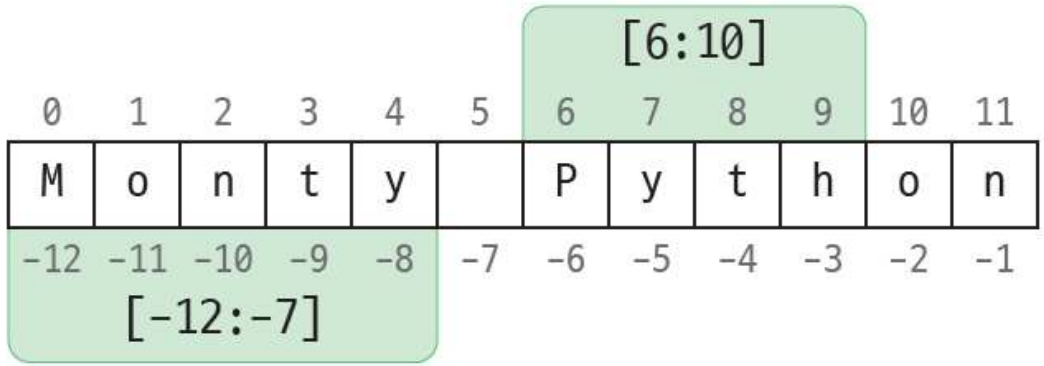


그림 9.1 문자열에서의 인덱스

인덱싱

```
>>> s = 'Monty Python'
>>> s[0]
'M'
>>> s[-1]
'n'
```

슬라이싱

- 슬라이싱이란 문자열의 일부를 잘라서 서브 문자열을 만드는 연산

그림 9.2 슬라이싱 연산

슬라이싱

```
>>> s = 'Monty Python'
>>> s[6:10]                          # 여러 문자 선택
'Pyth'


>>> t = s[:-1]                       # 마지막 문자 삭제
>>> t
'Monty Pytho'

>>> t = s[-2:]                       # 종료 인덱스 생략
>>> t
'on'

>>> s = 'Monty Python'
>>> s[:2]
'Mo'
>>> s[4:]
'y Python'
```

슬라이싱

```
>>> s = 'Monty Python'
>>> s[:2] + s[2:]
'Monty Python'
>>> s[:4] + s[4:]
'Monty Python'

>>> message='see you at noon'
>>> low = message[:5]                    # 문자열을 5번째 문자를 기준으로 둘로 나눈다.
>>> high = message[5:]
>>> low
'see y'
>>> high
'ou at noon'
```

in과 not in 연산자

```
>>> 'Hello' in 'Hello World'
True
>>> 'WORLD' in 'Hello World'
False
>>> 'WORLD' not in 'Hello World'
True
```

문자열 안에 문자열 넣기

- 다른 문자열 안에 문자열을 넣는 것은 프로그래밍에서 상당히 많이 필요한 작업이다. 파이썬에서는 + 연산자를 사용하여 이 작업을 할 수 있다.

```
>>> name = 'Kim'
>>> age = 21
>>> '제 이름은 ' + name + '입니다. 저는 ' + str(age) + '살입니다.'
'제 이름은 Kim입니다. 저는 21살입니다.'
```

```
>>> name = 'Kim'
>>> age = 21
>>> '제 이름은 %s입니다. 저는 %s살입니다.' % (name, age)
'제 이름은 Kim입니다. 저는 21살입니다.'
```

f-문자열

- f-문자열를 사용하면, 변수를 손쉽게 문자열 안에 넣어서 출력할 수 있다.

```
>>> name = 'Kim'
>>> age = 21
>>> f'제 이름은 {name}입니다. 저는 {age}살입니다.'
'제 이름은 Kim입니다. 저는 21살입니다.'
```

문자열 비교하기

- 어떤 문자열이 사전에서 앞에 있으면 < 연산자를 적용했을 때, 참이 된다.

```
>>> 'apple' < 'banana'
True
```

```
a = input('문자열을 입력하시오: ')
b = input('문자열을 입력하시오: ')
if( a < b ):
        print(a, '가 앞에 있음')
else:
        print(b, '가 앞에 있음')
```

```
문자열을 입력하시오: apple
문자열을 입력하시오: orange
apple 가 앞에 있음
```

Example: 회문 찾기

- 회문(palindrome)은 앞으로 읽으나 뒤로 읽으나 동일한 문장이다. 예를 들어서 ‘mom’,‘civic’, ‘dad’ 등이 회문의 예이다. 사용자로부터 문자열을 입력받고 회문인지를 검사하는 프로그램을 작성하여 보자.

```
문자열을 입력하시오: dad
회문입니다.
```

Example: 회문 찾기

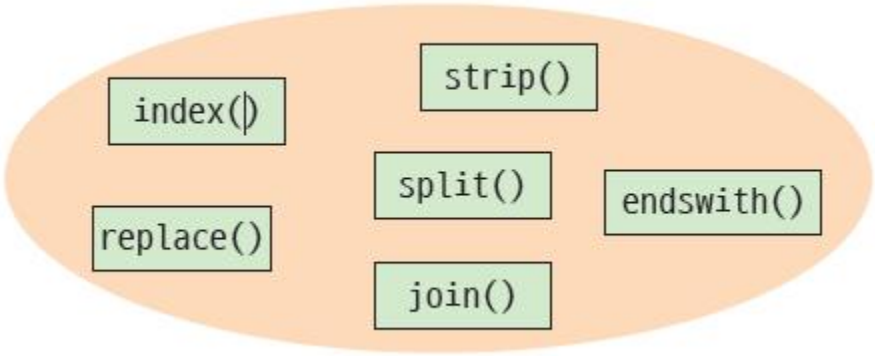
```
s = input('문자열을 입력하시오: ')
s1 = s[::-1] # 문자열을 거꾸로 만든다.

if( s == s1 ):
        print('회문입니다.')
else:
        print('회문이 아닙니다.')
```

중간점검

1. 문자열의 맨 끝에 있는 글자를 추출하는 명령문을 작성해보자.
2. 문자열 A와 문자열 B의 순서를 비교하려면 어떤 명령문을 사용하는가?

문자열 메소드 사용하기

- 파이썬은 아주 강력한 문자열 메소드들을 제공한다.



대소문자 변환하기

- upper() 및 lower() 메서드를 적용하면 대문자 또는 소문자로 변환된 새 문자열을 반환한다.

```
>>> s = 'Breakfast At Tiffany'
>>> s.upper()
'BREAKFAST AT TIFFANY'
```

- 이러한 메소드가 원본 문자열를 변경하지 않고 새로운 문자열을 반환한다는 것을 기억하여야 한다.

```
>>> s = 'Breakfast At Tiffany'
>>> s = s.lower()                    # 원본 문자열 s가 변경된다.
'breakfast at tiffany'
```

lower() 사용 예제

- upper() 및 lower() 메소드는 대소문자를 구분하지 않고 비교할 때 아주 유용하다. 다음 코드에서는 문자열이 먼저 소문자로 변환되기 때문에, 사용자가 yes, Yes, YES 중에서 어떤 것을 입력하더라도 올바르게 동작한다

```
print('게임을 계속하시겠어요?')
response = input()
if response.lower() == 'yes':
        print('게임을 계속합니다.')
else:
        print('다음에 또 뵈요.')
```

```
게임을 계속하시겠어요?
Yes
게임을 계속합니다.
```

문자열 검사 메소드

```
>>> 'HELLO'.isupper()          #문자열이 대문자로만 구성되는 경우 True를 반환한다.
True

>>> 'hello'.islower()          #문자열이 소문자로만 구성되는 경우 True를 반환한다.
True

>>> 'abc'.isalpha()            #문자열이 영문자로만 구성되는 경우 True를 반환한다.
True

>>> 'abc123'.isalpha()         #문자열이 영문자로만 구성되는 경우 True
False

>>> 'abc123'.isalnum()         #문자열이 영문자와 숫자로만 구성되는 경우
True

>>> '123'.isdecimal()          #문자열이 숫자로만 구성되는 경우 True
True

>>> ' \n'.isspace()            #문자열이 공백, 탭 및 줄바꿈 문자인 경우 True
True
```

입력의 유효성 검사하기

- 예를 들어 다음 프로그램은 사용자가 유효한 입력을 제공할 때까지 사용자에게 암호를 반복적으로 묻는다.

```
while True:
        print('새로운 패스워드를 선택하시오 (문자와 숫자만 가능)')
        password = input()
        if password.isalnum():
                break
        print('문자와 숫자만을 이용하여 패스워드를 선택하시오.')
```

```
새로운 패스워드를 선택하시오 (문자와 숫자만 가능)
password7$$$
문자와 숫자만을 이용하여 패스워드를 선택하시오.
새로운 패스워드를 선택하시오 (문자와 숫자만 가능)
password7
```

startswith()와 endswith() 메소드

- startswith(s)는 문자열 s로 시작되는 문자열이면 True가 반환된다.
  endswith(s)는 문자열 s로 종료되는 문자열이면 True가 반환된다.

```
>>> 'Breakfast At Tiffany'.startswith('Breakfast')
True

>>> 'Breakfast At Tiffany'.endswith('Tiffany')
True
```

파이썬 소스 파일 확인하기

- 사용자가 입력한 문자열이 파이썬 소스 파일 이름인지를 검사하려면 다음과 같이 하면 될 것이다.

```
s = input('파이썬 소스 파일 이름을 입력하시오: ')
if s.endswith('.py'):
        print('올바른 파일 이름입니다')
else :
        print('올바른 파일 이름이 아닙니다.')
```

```
파이썬 소스 파일 이름을 입력하시오: aaa.py
올바른 파일 이름입니다
```

split()로 문자열 분해하기

- split() 함수에서는 주어진 분리자를 이용하여 문자열을 토큰들의 리스트로 반환한다.

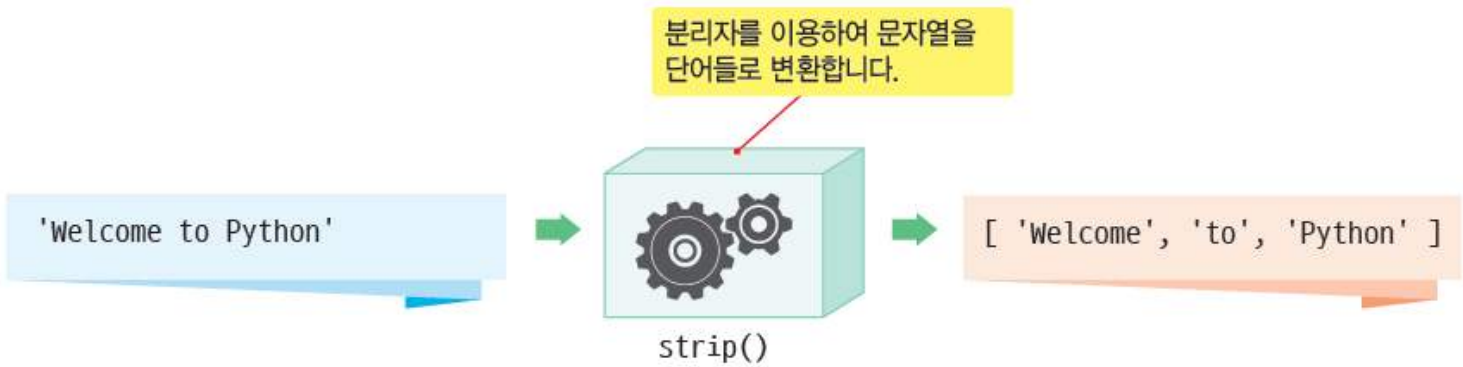


그림 9.3 split() 함수

split()로 문자열 분해하기

- split() 함수에서는 주어진 분리자를 이용하여 문자열을 토큰들의 리스트로 반환한다.

```
>>> s = 'Welcome to Python'
>>> s.split()
['Welcome', 'to', 'Python']
```

```
>>> s = 'Hello, World!'
>>> s.split(',')
['Hello', ' World!']
```

- 공백까지 제거하려면

```
>>> s = 'Hello, World!'
>>> s.split(', ')
['Hello', 'World!']
```

split()로 문자열 분해하기

- 여러 줄로 이루어진 문자열을 한 줄씩 분리하는데도 split()가 사용된다.

```
>>> lyric = '''Silent night, holy night
All is calm, all is bright
'Round yon virgin Mother and Child
Holy infant so tender and mild
Sleep in heavenly peace'''

>>> lyric.split('\n')
['Silent night, holy night', 'All is calm, all is bright', "'Round yon virgin Mother and Child',
'Holy infant so tender and mild', 'Sleep in heavenly peace']
```

문자열을 문자로 분해하려면

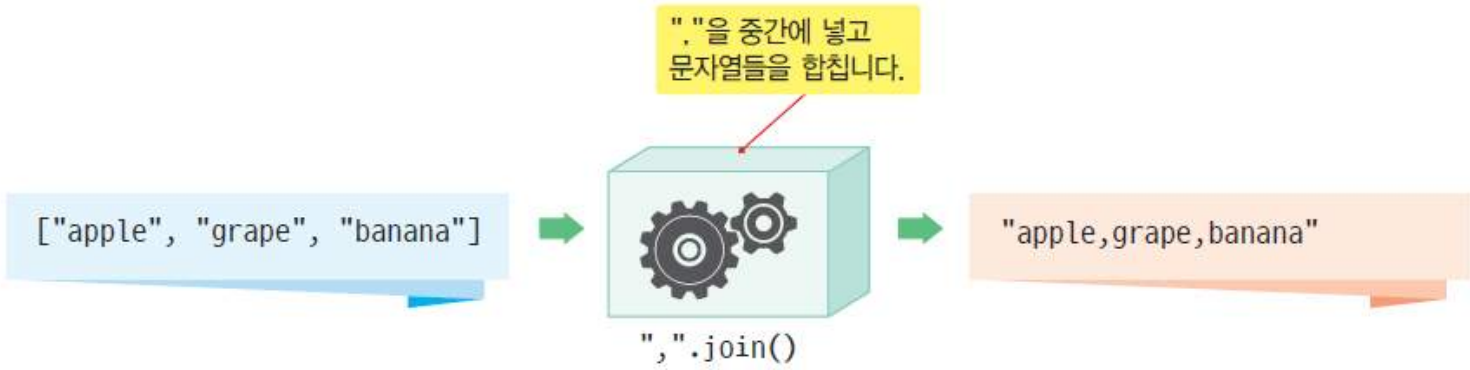
- list()를 호출해주면 된다. list()는 주어진 객체를 리스트화하는 함수이다.

```
>>> list('Hello, World!')
['H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!']
```

join()으로 문자열 합치기

- join()은 문자열 리스트를 전달받아 접착제 문자를 중간에 넣어서 문자열을 결합한다.

```
>>> '-'.join(['apple', 'grape', 'banana'])
'apple-grape-banana'

>>> ' '.join(['My', 'name', 'is', 'Kim'])
'My name is Kim'
```



join()으로 문자열 합치기

- 전화번호에서 '.' 문자 대신에 '-' 문자를 사용하고 싶으면 다음과 같은 코드를 사용한다.

```
>>> '-'.join('010.1234.5678'.split('.'))
'010-1234-5678'
```

- 문자들을 모아서 다시 문자열로 만들 때도 join()을 사용한다.

```
>>> s = 'hello world'
>>> clist = list(s)
>>> clist
['h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd']
>>> ''.join(clist)
'hello world'
```

join()으로 문자열 합치기

- split()와 join()을 함께 사용하면 문자열 중에서 필요 없는 공백을 제거할 수 있다.

```
>>> ' '.join('Actions \n\t speak louder   than words'.split())
'Actions speak louder than words'
```

strip()으로 공백 문자 제거하기

- strip()은 시작이나 끝에 공백 문자가 없는 새 문자열을 반환한다.

```
>>> s = ' Hello, World! '
>>> s.strip()
'Hello, World!'
```

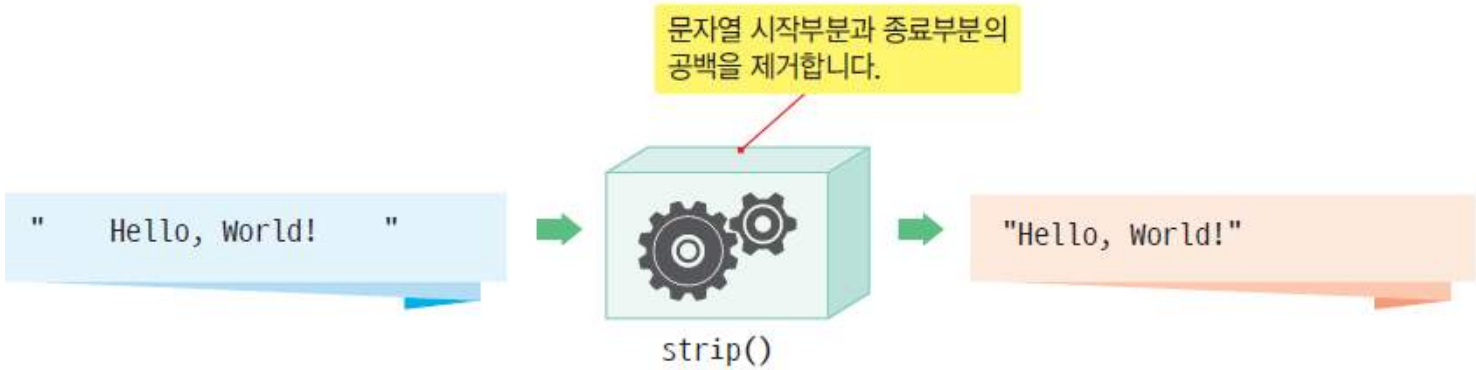


그림 9.4 strip() 함수

strip()으로 공백 문자 제거하기

- strip()은 시작이나 끝에 공백 문자가 없는 새 문자열을 반환한다.

```
>>> s = '###$$$this is example$$$###'
>>> s.strip('#$')
'this is example'
```

```
>>> s = '###$$$this is example$$$###'
>>> s.lstrip('#')
'$$$this is example$$$###'
>>> s.rstrip('#')
'###$$$this is example$$$'
```

ord()와 chr() 함수

- ord() 함수를 사용하여 문자의 코드값을 가져올 수 있다.

```
>>> ord('a')
97
>>> ord('가')
44032
```

- chr()는 ord()의 반대 기능을 수행한다. 숫자 값 n을 전달하면, n에 해당하는 문자를 반환한다.

```
>>> chr(97)
'a'
>>> chr(44032)
'가'
```

len() 함수

- len(s)는 문자열의 길이를 반환한다.

```
>>> s = 'Python is powerful!'
>>> print(len(s))
19
```

- str(obj)은 객체의 문자열 표현을 반환한다.

```
>>> str( 1+2j )
'(1+2j)'
```

찾기 및 바꾸기

- find() 함수는 문자열 안에서 특정 단어를 찾아서 인덱스를 반환한다. 찾지 못했을 경우에는 –1을 반환한다.

```
>>> s = 'www.naver.co.kr'
>>> s.find('.kr')                    # '.kr'의 인덱스를 반환한다.
12
```

- s.rfind(<sub>[, <start>[, <end>]])는 역순으로 문자열 안에서 단어를 검색한다.

```
>>> s = 'Let it be, let it be, let it be'
>>> s.rfind('let')                   # 문자열의 끝에서부터 탐색한다.
22
```

찾기 및 바꾸기

- count() 함수는 문자열 중에서 단어가 등장하는 횟수를 반환한다. .

```
>>> s = 'www.naver.co.kr'
>>> s.count('.')                    # 단어가 등장하는 횟수를 반환한다.
3
```

- replace() 함수는 문자열에서 하나의 단어를 다른 단어로 교체할 때 사용한다. 간단한 예는 다음과 같다.

```
>>> s = 'www.naver.com'
>>> s.replace('com', 'co.kr')       # 하나의 단어를 다른 단어로 교체한다.
'www.naver.co.kr'
```

머리 글자어 만들기

- 머리 글자어(acronym)은 NATO(North Atlantic Treaty Organization) 처럼 각 단어의 첫글자를 모아서 만든 문자열이다.

```
문자열을 입력하시오: North Atlantic Treaty Organization
NATO
```

```
phrase  = input('문자열을 입력하시오: ')
acronym = ''

# 대문자로 만든 후에 단어들로 분리한다.
for word in phrase.upper().split():
        acronym += word[0]          # 단어를 첫 글자만을 acronym에 추가한다.

print( acronym )
```

아이디와 도메인 구분하기

- 이메일 주소에서 아이디와 도메인을 구분하는 프로그램을 작성하여 보자.

```
이메일 주소를 입력하시오: aaa@google.com
aaa@google.com
아이디:aaa
도메인:google.com
```

```
address=input('이메일 주소를 입력하시오: ')
(id, domain) = address.split('@')

print(address)
print('아이디:'+id)
print('도메인:'+domain)
```

문자열의 공통 문자 찾기

- 사용자로부터 2개의 문자열을 받아서 두 문자열의 공통 문자를 출력하는 프로그램을 작성해보자.

```
첫 번째 문자열: Hello World!
두 번째 문자열: How are you?

공통적인 글자: o H r e
```

```
s1=input('첫 번째 문자열:')
s2=input('두 번째 문자열:')

list1 = list( set(s1) & set(s2) )              # 세트로 만들고 교집합 연산을 한다.

print('\n공통적인 글자:', end=' ')
for i in list1:
    print(i, end=' ')
```

일회용 암호 만들기

- 파이썬에서 random() 함수는 난수 생성 함수로서 임의의 OTP를 생성하는 데 사용할 수 있다. sample()을 사용하면 문자열에서 지정된 개수의 글자를 랜덤하게 고를 수 있다.

```
import random

s = '0123456789'                    # 대상 문자열
passlen = 4                                 # 패스워드 길이

p =  ''.join(random.sample(s, passlen ))
print(p)
```

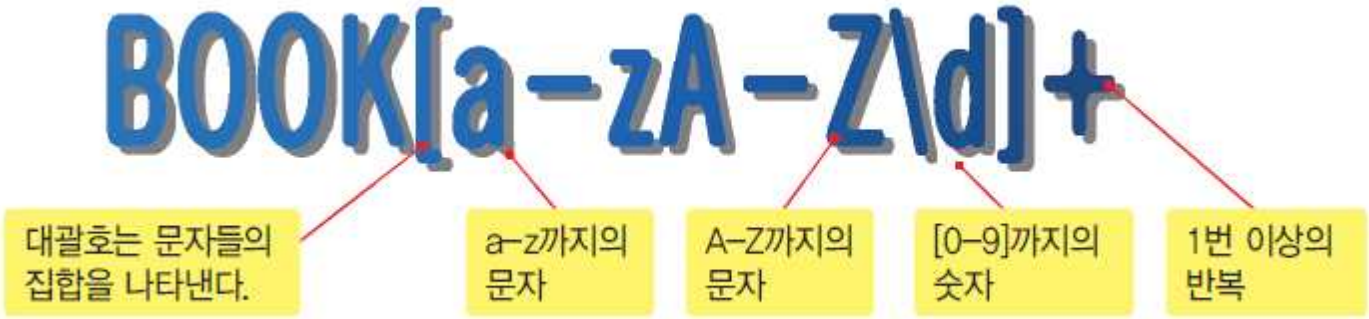
```
3482
```

중간점검

1. 문자열에 포함된 공백을 제거하기 위하여 사용되는 함수는?
2. 문자열과 문자열을 결합하는데 사용되는 함수는?
3. 문자열 안의 단어들을 분리하는 데 사용되는 함수는?

정규식

- 정규식(regular expression)은 어떤 패턴을 이용하여 문자열을 찾을 수 있는 기능이다.



정규식의 예

- 예를 들어서 우리나라의 스마트폰 번호를 문서에서 찾는다고 가정하자.
- 항상 010으로 시작하고 이어서 '-', 숫자 4개, '-', 숫자 4개가 온다. 예를 들어서 010-1234-5678은 스마트폰 번호이지만 01,012,345,678는 아마 스마트폰 번호가 아닐 것이다.

정규식을 사용하지 않고 스마트폰 번호 찾기

```
def checkNumber(phoneNumber):
        if len(phoneNumber) != 13:
                return False
        if phoneNumber[0:3] != '010':
                return False
        if phoneNumber[3] != '-':
                return False
        for i in range(4, 8):
                if not phoneNumber[i].isdecimal():
                        return False
        if phoneNumber[8] != '-':
                        return False
        for i in range(9, 13):
                if not phoneNumber[i].isdecimal():
                        return False
        return True
print('010-8888-6666->', checkNumber('010-8888-6666'))
print('000-1111-abcd->', checkNumber('000-1111-abcd'))
```

```
010-8888-6666-> True
000-1111-abcd-> False
```

정규식을 사용하여 스마트폰 번호 찾기

- 정규식에서 \d는 숫자를 의미한다. 정규식 "010-\d\d\d\d-\d\d\d\d" 은 이전 checkNumber() 함수가 수행한 것과 동일한 작업을 할 수 있다.

```
import re

pattern = r'010-\d\d\d\d-\d\d\d\d'
found = re.search(pattern, '제 휴대폰 번호는 010-1234-5678입니다.')
print('발견된 휴대폰 번호: ' + found.group())
```

```
010-8888-6666-> True
000-1111-abcd-> False
```

정규식

- 정규식(regular expression)이란 특정한 규칙을 가지고 있는 문자열들을 표현하는 수식이다.

식	기능	설명
^	시작	문자열의 시작을 표시
$	끝	문자열의 끝을 표시
.	문자	한 개의 문자와 일치
\d	숫자	한 개의 숫자와 일치
\w	문자와 숫자	한 개의 문자나 숫자와 일치
\s	공백문자	공백, 탭, 줄바꿈, 캐리지리턴 문자와 일치
[]	문자 종류, 문자 범위	[abc]는 a 또는 b 또는 c를 나타낸다. [a-z]는 a부터 z까지 중의 하나, [1-9]는 1부터 9까지 중의 하나를 나타낸다.

정규식의 예

- abc 정확히 “abc”와만 일치된다.
- . 한자리의 문자, 예를 들어서 “A”, “1”, “$”,
- \d\d\d 3자리의 숫자, 예를 들어서 “010”, “123”
- [a-z] a부터 z사이의 한 글자와 일치
- \w 한자리 문자나 숫자, 예를 들어서 “8”, “A”,

정규식의 수량 한정자

- 정규 표현식에는 메타문자 뒤에 수량 한정자(quantifier)를 붙일 수 있다. 수량 한정자는 문자가 몇 번 반복되느냐를 나타낸다.

수량 한정자	기능	설명
*	0회 이상 반복	"a*"는 "", "a", "aa", "aaa"를 나타낸다.
+	1회 이상	"a+"는 "a", "aa", "aaa"를 나타낸다.
?	0 또는 1회	"a?"는 "", "a"를 나타낸다.
{m}	m회	"a{3}"는 "aaa"만 나타낸다.
{m, n}	m회 이상 n회 이하	"a{1, 3}"는 "a", "aa", "aaa"를 나타낸다.
(ab)	그룹화	(ab)*은 "", "ab", "abab" 등을 나타낸다.

정규식의 예

- .+ 문자가 1회 이상 반복
- ^[1-9][0-9]*$ 처음 숫자는 0이 아닌 숫자,
- ^\d{6}-\d{7}$ 중간에 –이 있는 주민등록번호
- (Good)?Bye “GoodBye” 또는 “Bye”

중요한 메타 문자

- 메타 문자 중에서 가장 중요한 문자는 점(.)과 별표(*)이다. 점은 어떤 문자가 와도 상관없다는 의미이다. 별표는 몇 번 반복되어도 상관없다는 것을 의미한다. 예를 들어보자.

- “X-Men: First Class“, “X-Men: Days of Future Past“, “X-Men Origins: Wolverine” 과 매칭된다.

숫자로 시작하는 줄 찾기

- 미국 헌법이 저장된 텍스트 파일에서 숫자로 시작되는 줄만을 추출하여고 한다.

```
import re
f = open('uscons.txt')
for line in f:
        line = line.rstrip()
        if re.search('^[0-9]+', line) :
                print(line)
f.close()
```

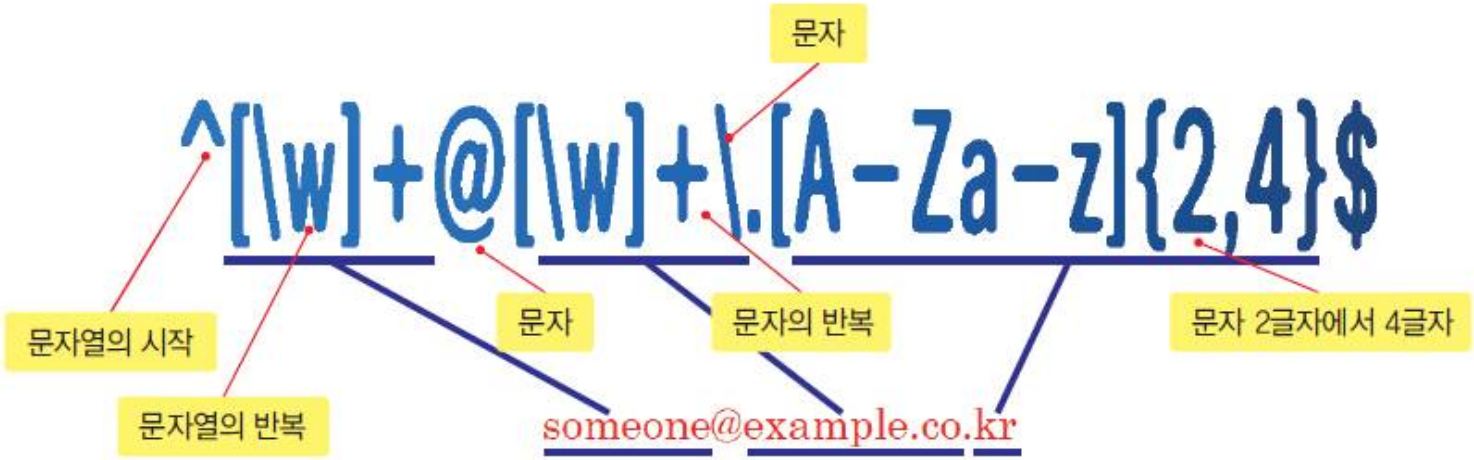
```
1. Neither slavery nor involuntary servitude, except as a punishment for crime
2. Congress shall have power to enforce this article by appropriate
...
```

중간점검

1. 'a'에서 'z'까지의 글자로만 이루어진 줄을 파일에 찾고 싶으면 어떻게 정규식을 만들어야 하는가?
2. 숫자로만 이루어진 줄을 파일에 찾고 싶으면 어떻게 정규식을 만들어야 하는가?

Lab: 이메일 주소를 찾아보자

- 여러분이 웹페이지에서 웹크롤링을 통하여 텍스트를 가져와다고 하자. 이 텍스트 중에서 이메일 주소만을 추출할려고 한다.



Sol.

```
import re

pattern = r'^[\w]+@[\w]+\.[A-Za-z]{2,4}$'

def checkEmail(emailAddress):
        if(re.fullmatch(pattern, emailAddress)):
                print(f"{emailAddress}는 유효한 이메일 주소입니다.")
        else:
                print(f"{emailAddress}는 유효하지 않은 이메일 주소입니다.")

email = "abc123@gmail.com"
checkEmail(email)

email = "abc.cde@com"
checkEmail(email)
```

```
유효한 이메일 주소입니다.
유효하지 않은 이메일 주소입니다.
```

Lab: 패스워드 검사 프로그램

- 패스워드의 조건은 다음과 같다.
  - 최소 8글자
  - 적어도 하나의 대문자
  - 적어도 하나의 숫자

```
패스워드를 입력하시오: abcdef
패스워드는 최소한 8글자이어야 합니다.
패스워드를 입력하시오: abcdefrg
패스워드는 적어도 하나의 숫자를 가져야 합니다.
패스워드를 입력하시오: abbabbb1
패스워드는 적어도 대문자를 가져야 합니다.
패스워드를 입력하시오: abbbabbaA1
규정에 맞는 패스워드입니다.
```

Sol.

```
import re

def check():
    while True:
        password = input("패스워드를 입력하시오: ")
        if len(password) < 8:
            print("패스워드는 최소한 8글자이어야 합니다.")
        elif re.search('[0-9]', password) is None:
            print("패스워드는 적어도 하나의 숫자를 가져야 합니다.")
        elif re.search('[A-Z]', password) is None:
            print("패스워드는 적어도 대문자를 가져야 합니다.")
        else:
            print("규정에 맞는 패스워드입니다.")
            break

check()
```

Lab: 단어 카운터 만들기

- 다음 프로그램은 문자열의 각 단어가 나타나는 횟수를 계산한다. 딕셔너리를 사용한다. split()는 문자열을 토큰으로 만드는 함수이다.

```
Create 의 등장횟수= 1
because 의 등장횟수= 1
become 의 등장횟수= 1
believe 의 등장횟수= 1
...
```

Sol.

```
text_data ='Create the highest, grandest vision possible for your life, because you
become what you believe'

word_dic = {}                          # 단어들과 출현 횟수를 저장하는 딕셔너리를 생성

for w in text_data.split():                    # 텍스트를 단어들로 분리하여 반복한다.
    if w in word_dic:                          # 단어가 이미 딕셔너리에 있으면
        word_dic[w]  += 1                      # 출현 횟수를 1 증가한다.
    else:                                      # 처음 나온 단어이면 1로 초기화한다.
        word_dic[w]   = 1

for w, count in sorted(word_dic.items()): # 키와 값을 정렬하여 반복 처리한다.
    print(w, '의 등장횟수=', count)
```

Mini Project: 단답형 퀴즈 프로그램 작성

- 단답형 문제를 출제하고 채점하는 퀴즈 프로그램을 작성해보자.

```
CPU는 무엇의 약자인가?
답안을 작성하시오(또는 quit):  Central Processing Unit
정답입니다.

제일 쉬운 프로그래밍 언어는?
답안을 작성하시오(또는 quit):  파이썬
정답입니다.
...
```

이번 장에서 배운 것

- 파이썬에서 문자열은 일련의 문자이다. 파이썬 문자열은 변경할 수 없다.
- 작은따옴표나 큰따옴표를 사용하여 문자열 리터럴을 만들 수 있다.
- 문자열에서 따옴표를 이스케이프 하려면 백슬래시 문자 ₩를 사용한다.
- 문자열은 + 연산으로 합칠 수 있고 * 연산으로 반복시킬 수 있다.
- r'...'은 원시 문자열로서 각종 특수 문자가 일반 문자로 취급된다.
- f-문자열을 사용하여 문자열에 변수의 값을 삽입할 수 있다.
- len() 함수를 사용하여 문자열의 크기를 계산할 수 있다.
- 문자열에 대해서도 슬라이싱과 인덱싱이 가능하다.
- 슬라이싱을 사용하여 문자열에서 하위 문자열을 추출한다.
- 문자열의 n번 위치에 있는 문자에 액세스 하려면 str[n]을 사용한다.
- 문자열의 split() 메소드는 분리자 문자를 받아서 문자열을 단어로 분리한다.
- 문자열의 join() 메소드는 접착제 문자를 받아서 단어로 접착하여서 문자열로 만든다.
- 정규식은 문자열의 특정 패턴을 정의한다.
- 정규식을 사용하면 스마트폰 번호와 같은 특정한 패턴을 쉽게 찾을 수 있다.

Q & A