Lecture #07 | 문자열 처리, 파일입출력

SE213 프로그래밍 (2018)

지난 시간에 다룬 내용

- 복합자료구조
- 반복문
  - while v.s. for
  - while문 추가 설명
- 문자열 처리
  - 문자열 서식화

오늘 다룰 내용

- 문자열 처리
  - 문자열 함수
- 파일 입출력 (텍스트 파일)
- 프로그래밍 스타일

문자열 함수 (일부)

- 문자열 함수의 용법: `str.function(arguments)`
  - 나누기/붙이기: `split()`, `join()`, `splitlines()`
  - 문자열 검색/치환: `find()`, `rfind()`, `count()`, `replace()`
  - 서식화: `rjust()`, `ljust()`, `center()`, `format()`
  - 공백문자 제거: `strip()`, `rstrip()`, `lstrip()`
- str에 위치할 수 있는 것들
  - 문자열
  - 문자열을 가르키는 변수
  - 문자열을 반환하는 함수
- 반환값: 대다수의 함수들은 문자열을 반환
  - 예외: `split()`, `splitlines()`는 리스트를 반환함

split(): 구분자를 기준으로 분리된 문자열의 리스트를 반환

`str.split(sep=None, maxsplit=-1)`

- Return a list of the words in the string, using `sep` as the delimiter string.
- If `maxsplit` is given, at most `maxsplit` splits are done (thus, the list will have at most maxsplit+1 elements).
- If `maxsplit` is not specified or `-1`, then there is no limit on the number of splits (all possible splits are made).
- If `sep` is given, consecutive delimiters are not grouped together and are deemed to delimit empty strings (for example, `'1,,2'.split(',')` returns `['1', '', '2']`).
- The `sep` argument may consist of multiple characters (for example, `'1<>2<>3'.split('<>')` returns `['1', '2','3']`).
- Splitting an empty string with a specified separator returns `['']`.

`splitlines()`: 개행문자로 분리된 문자열의 리스트를 반환

`str.splitlines([keepends])`

- Return a list of the lines in the string, breaking at line boundaries.
- Line breaks are not included in the resulting list unless **keepends** is given and true.
- This method splits on the following line boundaries.
- In particular, the boundaries are a superset of universal newlines.
- 참고: `split('\n')`과 유사하나, 마지막 빈 줄을 남기지 않음

예시: split(), splitlines() 함수

```
quotes = '''First, solve the problem. Then, write the code. - John Johnson
Without requirements or design, programming is the art of adding bugs to an empty text file. - Louis Srygley
Computers are good at following instructions, but not at reading your mind. - Donald Knuth
Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. - John Woods'''
list_quote = quotes.splitlines()
print(list_quote)
for quote in list_quote:
    sentence, author = quote.split(' - ')
    print('"' + sentence + '" by ' + author)
    # or print(f'"{sentence}" by {author}')
```

예시: split(), splitlines() 함수 (실행결과)

```
['First, solve the problem. Then, write the code. - John Johnson', 'Without requirements or design, programming is the art of adding bugs to an empty text file. - Louis Srygley', 'Computers are good at following instructions, but not at reading your mind. - Donald Knuth', 'Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. - John Woods']
"First, solve the problem. Then, write the code." by John Johnson
"Without requirements or design, programming is the art of adding bugs to an empty text file." by Louis Srygley
"Computers are good at following instructions, but not at reading your mind." by Donald Knuth
"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live." by John Woods
```

join(): 문자열들을 합친 문자열을 반환

`str.join(`*`iterable`*`)`

- Return a string which is the concatenation of the strings in *`iterable`*. A `TypeError` will be raised if there are any non-string values in *`iterable`*, including `bytes` objects. The separator between elements is the string providing this method.

```
t = ['apple', 'pear', 'banana']
s1 = '.'.join(t)
s2 = ', '.join(t)
print(s1)
print(s2)
```

```
apple.pear.banana
apple, pear, banana
```

strip(): 앞뒤로 지정된 문자열 제거

`str.strip(`*`[chars]`*`)`*

- Return a copy of the string with the leading and trailing characters removed. The *chars* argument is a string specifying the set of characters to be removed. If omitted or `None`, the *chars* argument defaults to removing whitespace. The *chars* argument is not a prefix or suffix; rather, all combinations of its values are stripped.
- The outermost leading and trailing chars argument values are stripped from the string. Characters are removed from the leading end until reaching a string character that is not contained in the set of characters in *chars*. A similar action takes place on the trailing end.
- 참고: 파일/네트웍에서 한 줄을 읽은 후, 공백을 제거하기 위해 주로 사용함

* 인자에서 [ ]는 리스트가 아닌, 생략 가능을 의미함

rstrip(), lstrip(): 앞 혹은 뒤의 지정된 문자열 제거

`str.rstrip(`*`[chars]`*`)`*

- Return a copy of the string with trailing characters removed. The *chars* argument is a string specifying the set of characters to be removed. If omitted or `None`, the *chars* argument defaults to removing whitespace. The *chars* argument is not a suffix; rather, all combinations of its values are stripped.

`str.lstrip(`*`[chars]`*`)`*

- Return a copy of the string with leading characters removed. The *chars* argument is a string specifying the set of characters to be removed. If omitted or `None`, the *chars* argument defaults to removing whitespace. The *chars* argument is not a prefix; rather, all combinations of its values are stripped.

* 인자에서 [ ]는 리스트가 아닌, 생략 가능을 의미함

예제: strip(), rstrip(), lstrip()

```
print('   spacious   '.strip())
print('www.example.com'.strip('cmowz.'))
comment_string = '#....... Section 3.2.1 Issue #32 .......'
print(comment_string.strip('.#! '))
print('   spacious   '.rstrip())
print('mississippi'.rstrip('ipz'))
print('   spacious   '.lstrip())
print('www.example.com'.lstrip('cmowz.'))
```

```
spacious
example
Section 3.2.1 Issue #32
   spacious
mississ
spacious
example.com
```

파일

- 파일
  - 문자, 숫자 등으로 이루어진 정보의 집합체
  - HDD, SSD, USB drive, 클라우드 등의 저장장치에 저장됨
  - 파일이 저장되어 있는 공간(파일 경로 혹은 path)과 이름(파일명)으로 구분됨
- 텍스트 파일: 여러 줄의 사람들이 인지할 수 있는 문자들로 이루어진 파일
  - 문자인코딩(예: ASCII, UTF-8, CP-949)에 따라 표현할 수 있는 문자가 달라짐
  - 참고: 파이썬 소스 파일(.py), html 파일 (.html) 등도 텍스트 파일의 일종
- 바이너리 파일: 텍스트 파일이 아닌 파일
  - 예: 아래아한글 문서파일 (.hwp), 워드파일 (.doc), 이미지파일 (.jpg, .png) 등

텍스트 파일입출력

- 파일 입출력 전후로 파일을 열고, 닫는 단계가 필요함 (컴퓨터로 문서를 편집하기 위해서 파일을 여는 것과 유사함)
  - 파일 열기 (open)
  - 파일 읽기 혹은 쓰기 (read or write)
  - 파일 닫기 (close): `with`를 사용한 경우 생략
- python에서 제일 간단한 텍스트 파일 입출력 방법
  - 파일입력

  1. `read()` 함수 이용 → 파일 전체를 하나의 문자열로 읽음
  2. `split()`, `splitlines()` 함수 이용 → 한 줄씩 처리

  - 파일출력: `print()` 함수 이용 → 화면에 인쇄하는 것처럼 파일 출력 가능

텍스트 파일 읽기

- 코드 설명
  - example.txt라는 이름의 파일의 내용을 변수 file_contents에 저장
- 일반적인 방법

```
fileobj = open('example.txt', 'rt')
file_content = fileobj.read()
fileobj.close()
```

- with를 사용 (close()를 호출할 필요가 없음)

```
with open('example.txt', 'rt') as fileobj:
    file_content = fileobj.read()
```

텍스트 파일 쓰기

- 코드 설명
  - example.txt라는 이름의 파일에 문자열 something을 쓰기
- 일반적인 방법

```
fileobj = open('example.txt', 'wt')
print('something', file=fileobj)
fileobj.close()
```

- with를 사용 (close()를 호출할 필요가 없음)

```
with open('example.txt', 'wt') as fileobj:
    print('something', file=fileobj)
```

예시: 파일입출력 함수

```
quotes = '''First, solve the problem. Then, write the code. - John Johnson

Without requirements or design, programming is the art of adding bugs to an empty text file. - Louis Srygley

Computers are good at following instructions, but not at reading your mind. - Donald Knuth

Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. - John Woods'''

with open('quotes.txt', 'wt') as f:

    print(quotes, file=f)

with open('quotes.txt', 'rt') as f:

    file_contents = f.read()

print(file_contents)
```

예시: 파일입출력 함수 (실행 결과)

```
First, solve the problem. Then, write the code. - John Johnson
Without requirements or design, programming is the art of adding bugs to an empty text file. - Louis Srygley
Computers are good at following instructions, but not at reading your mind. - Donald Knuth
Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. - John Woods
```

First, solve the problem.
Then, write the code.

John Johnson

코딩스타일

- 프로그램을 작성할 때의 규칙 혹은 가이드라인
  - 함수/변수 이름, 주석, 띄어쓰기, 함수 작성 기준 등 여러 가지 사항들이 정의되어 있음
- 프로젝트/언어별로 정의되어 있는 경우가 많음
  - 가장 중요한 것을 한 프로젝트 내에서의 일관성(consistency)
- python의 공식 style guide
  - PEP 8: https://www.python.org/dev/peps/pep-0008/ (번역1, 번역2, 번역3)
  - PEP 257 (docstring): https://www.python.org/dev/peps/pep-0257/
- Google python style guide: https://google.github.io/styleguide/pyguide.html

주석

- 프로그램의 동작에는 영향을 끼치지 않으나, 프로그램을 사용 혹은 변경하는 사람들을 위하여 코드의 기능 혹은 코드를 왜 작성했는지에 대한 설명
- 주석달 때 주의할 점들
  - 함수/클래스 등의 인자, 반환값, 기능 등은 함수/클래스를 작성하면서 반드시 작성할 것
  - 코드가 어떻게 작동하는 것보다 왜 작성을 했는지를 설명하는 것이 필요
  - 코드를 변경하면 항상 주석을 업데이트 해야 됨
    → 코드와 주석이 일치하지 않으면, 주석이 없는 경우보다 훨씬 더 안 좋음
  - 읽기 좋은 코드 자체가 가장 훌륭한 주석임

Always code as if the guy who ends up maintaining your code will be ***a violent psychopath*** who knows where you live.

Code for readability.

John Woods

ANY QUESTIONS?