

PYTHON 프로그래밍 실습

파이썬 기초

(튜플, 집합, 딕셔너리, 파일)

2019-12-27



טיפ



튜플

■ 튜플(tuple)

- 리스트와 유사하지만 값을 수정할 수 없으며, 읽기만 가능
- 튜플은 괄호()로 생성

```
>>> t1 = ()  
>>> t2 = (1,)          # 항목이 하나인 경우 주의  
>>> t3 = (1, 2, 3)  
>>> t4 = 1, 2, 3       # 괄호 생략 가능
```

- 프로그램이 동작하는 중 변경되지 않는 데이터의 저장

튜플 특성

■ 튜플 인덱싱, 슬라이싱

```
>>> t = (1, 2, 3, 4, 5, 6)
>>> t[0]
1
>>> t[:3]
(1, 2, 3)
>>> t[4:6]
(5, 6)

>>> t[0] = 7 ???
```

■ 튜플 더하기, 곱하기

```
>>> t1 = (1, 2)
>>> t2 = ('a', 'b')

>>> t3 = t1 + t2
>>> t3
(1, 2, 'a', 'b')

>>> t1 * 3
(1, 2, 1, 2, 1, 2)
```

튜플 패킹, 언패킹

■ 튜플 패킹(Tuple Packing)

: 여러 데이터를 튜플로 묶는 것

```
>>> a = 1, 2, 3
>>> a
(1, 2, 3)
```

■ 튜플 언패킹(Tuple Unpacking)

: 튜플의 각 항목을 여러 개의 변수에 할당하는 것

```
>>> one, two, three = a
>>> print(one, two, three)
1 2 3
```

■ 튜플 관련 함수

■ index(), count() 제공

값 1의 위치

```
>>> a.index(1)
```

값 1과 일치하는 요소의 개수

```
>>> a.count(1)
```



집합



집합

■ 집합(set)

- 순서 없이 항목을 저장, 항목이 중복 될 수 없음

※ Python 3.x

```
>>> s1 = set([1,2,3]) # 리스트기반
>>> s2 = {3,4,5}
```

```
>>> word = "Hello" * 2
>>> word
'HelloHello'
>>> word_list = list(word)
>>> word_list
['H', 'e', 'l', 'l', 'o', 'H', 'e', 'l', 'l', 'o']
>>> s3 = set(word_list)
```

```
>>> s3
{'H', 'e', 'l', 'o'}
```

```
>>> s4 = set("Hello") #문자열기반
>>> s4
{'H', 'e', 'l', 'o'}
```

※ 자료형의 중복을 제거하기 위한 필터로 종종 사용

```
>>> t = tuple(s4)
>>> t
('H', 'e', 'l', 'o')
```

집합 관련 함수

※ Python 3.x

```
>>> s = {1, 2, 3, 4, 5, 3, 4}
>>> s
{1, 2, 3, 4, 5}
>>> s.add(10)
>>> s
{1, 2, 3, 4, 5, 10}
>>> s.remove(2)
>>> s
{1, 3, 4, 5, 10}
```

```
>>> s.update([1,3,5,7,9])
>>> s
{1, 3, 4, 5, 7, 9, 10}
>>> s.discard(7)
>>> s
{1, 3, 4, 5, 9, 10}
>>> s.clear()
>>> s
set()
```


집합

※ Python 3.x

■ 합집합, 교집합, 차집합

```
>>> s1 = {1, 2, 3}
>>> s2 = {3, 4, 5}
```

합집합

```
>>> s1.union(s2)
{1, 2, 3, 4, 5}
```

교집합

```
>>> s1.intersection(s2)
{3}
```

차집합(-) s1.difference(s2)

```
>>> s1 - s2
{1, 2}
```

합집합(|)

```
>>> s1 | s2
{1, 2, 3, 4, 5}
```

교집합(&)

```
>>> s1 & s2
{3}
```



딕셔너리



딕셔너리

- 딕셔너리(dictionary)
 - 중괄호{ }로 묶여 있으며, key와 value의 쌍으로 이루어짐
d = {key1:value1, key2:value2}
 - key로 value를 관리
key는 중복 될 수 없음. 예)학번, 주민번호
 - 항목들 사이에 순서가 없음

딕셔너리

```
>>> d = {}
>>> d['kim'] = 1      # 새 항목 추가
>>> d['park'] = 2     # 새 항목 추가
>>> d                 # 전체 출력
{'park': 2, 'kim': 1}
>>> d['park']        # key를 사용하여 값에 접근
2
# 이미 있는 키의 경우 기존의 값 변경
>>> d['kim'] = 3
>>> d
{'park': 2, 'kim': 3}
>>> d['youn'] = 1     # 새 항목 추가
>>> d
{'youn': 1, 'park': 2, 'kim': 3}
```

```
>>> d.keys()          # 키만 추출
dict_keys(['kim', 'park', 'youn'])
>>> d.values()        # 값만 추출
dict_values([3, 2, 1])
# 키와 값의 항목을 튜플 형태로
>>> d.items()
dict_items([('kim', 3), ('park', 2), ('youn', 1)])

# 딕셔너리에 키가 있는 경우
>>> 'kim' in d
True
# 딕셔너리에 키가 없는 경우
>>> 'lee' in d
False
```

※ Python 3.x

딕셔너리

```
>>> d  
{'youn': 1, 'park': 2, 'kim': 3}
```

항목 1개 삭제

```
>>> del d['kim']  
>>> d  
{'youn': 1, 'park': 2}
```

전체 삭제

```
>>> d.clear()  
>>> d  
{}
```

딕셔너리 – 실습 1

- 어떤 문장을 입력 받으면 해당 문장에서 각 알파벳이 몇 개씩 나오는지 저장하는 딕셔너리를 만든 후, 아래와 같이 출력하시오.
- 실행예시(밑줄은 사용자 입력)

```
Enter a sentence: Python is fun!  
{ '!': 1, ' ': 2, 'f': 1, 'i': 1, 'h': 1, 'o': 1, 'n': 2, 'P': 1,  
's': 1, 'u': 1, 't': 1, 'y': 1 }
```



파일



파일에서 데이터 읽기

파일을 연다.



파일에서 데이터를
읽거나 쓴다.



파일을 닫는다.

파일객체

파일이름

파일 열기 모드

```
f = open("input.txt", "r")  
...  
f.close()
```


파일 열기 모드	모드 이름	설명
"r"	읽기 모드 (read)	파일을 읽기만 할 때 사용. 파일의 처음부터 읽는다.
"w"	쓰기 모드 (write)	파일에 쓸 때 사용. 파일의 처음부터 쓴다. 파일이 없으면 생성된다. 파일이 존재하면 기존의 내용은 지워진다.
"a"	추가 모드 (append)	파일의 마지막에 새로운 내용을 추가 시킬 때 사용 파일이 없으면 생성된다.
"r+"	읽기/쓰기 겸용 모드	

파일 쓰기

■ `f = open("new.txt", "w")`

write.py

```
f = open("new.txt", "w")  
  
for i in range(1, 6):  
    data = "%d line\n" % i  
    f.write(data)  
  
f.close()
```

new. txt

```
1 line  
2 line  
3 line  
4 line  
5 line
```

파일 읽기(1/2)

(1) *readline()*

파일에서 한 줄 읽어옴

```
f = open("new.txt", "r")
line = f.readline()
print (line, end="")
line = f.readline()
print (line, end="")
f.close()
```

```
1 line
2 line
>>>
```

(2) *readlines()*

모든 라인을 한꺼번에 읽어서 각각의 줄을 요소로 갖는 리스트를 반환

```
f = open("new.txt", "r")
lines = f.readlines()
print (lines)
f.close()
```

```
['1 line\n', '2 line\n', '3 line\n', '4 line\n', '5 line\n']
>>>
```

파일 읽기(1/2)

(3) *f.read()*:

파일을 전부 읽은 문자열을 반환

```
f = open("new.txt", "r")
```

```
data = f.read()
```

```
print (data)
```

```
f.close()
```

```
1 line  
2 line  
3 line  
4 line  
5 line
```

```
>>>
```

(4) *for* 문 사용

```
f = open("new.txt", "r")
```

```
for line in f:  
    print (line, end='')
```

```
f.close()
```

```
1 line  
2 line  
3 line  
4 line  
5 line  
>>> |
```

파일 - 실습 1

- 파일에 있는 각각의 단어의 수 구하기

test.txt

```
first line  
second line  
third line
```

실행예시

```
line 3  
second 1  
third 1  
first 1  
>>>
```

(힌트) 딕셔너리 사용

파일 - 실습 2

- 파일명을 입력 받아, 해당 파일을 한 줄씩 읽어 파일의 내용을 모두 대문자로 출력하는 프로그램을 작성하시오.

(힌트)

- 문자열을 대문자로 변환

```
>>> "hello".upper()
```

```
'HELLO'
```

- 문자열을 소문자로 변환

```
>>> "World".lower()
```

```
'world'
```

- `os.path.exists(파일명)`: 파일이나 디렉토리가 존재하는지 검사

```
>>> import os
```

```
>>> os.path.exists("test.txt")
```

```
True
```

```
Enter a file name: test.txt
FIRST LINE
SECOND LINE
THIRD LINE
>>>
```

파일에 내용 추가하기

```
f = open("new.txt", "a")

for i in range(6, 11):
    data = "%d line\n" % i
    f.write(data)

f.close()
```

new. txt

1 line
2 line
3 line
4 line
5 line
6 line
7 line
8 line
9 line
10 line

추가된 내용

파일을 열고 자동으로 닫기(with~as)

- open() ~ close() 사용

```
f = open( "output.txt", "w" )  
f.write("Python is fun!")  
f.close()
```

- with open() as 사용

```
with open( "output.txt", "w" ) as f :  
    f.write("Python is fun!")
```

with 블록을 벗어나는 순간 파일을 자동으로 닫아준다.

파일 포인터

- 파일 포인터
 - 파일의 현재 위치를 가리키는 것
- *f.tell()*: 현재 파일 포인터의 위치 반환
- *f.seek()*: 지정하는 곳으로 포인터의 위치를 변경

test.txt

first line
second line
third line

```
>>> f = open("test.txt", "r")
>>> f.tell()
0
>>> f.readline()
'first line\n'
>>> f.tell()
11
>>> 
```



파일(참고자료)



파이썬 프로그램에 입력인수 전달

- sys 모듈 입력
- 도스(Dos)나 리눅스셸에서 파이썬 명령어를 실행하면서 입력인수를 넣어줄 수 있음.

Ex) 도스명령어 [인수1 인수2]

```
#sys1.py
import sys

args = sys.argv[1:]
for i in args:
    print (i)
```

```
$ python sys1.py aaa bbb ccc
```

```
c:\>python sys1.py aaa bbb ccc
```

argv[0]

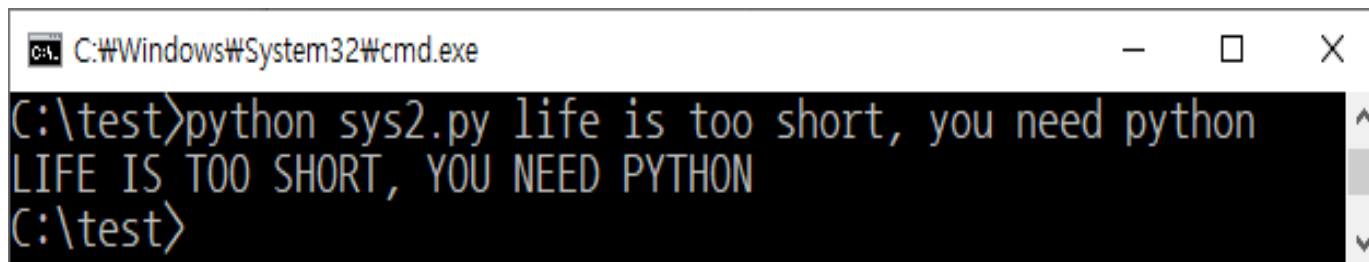
argv[3]

```
C:\test>python sys1.py this is test
this
is
test
C:\test>
```

입력인수 전달 실습

- 명령행에 입력된 소문자를 대문자로 바꾸어 주는 프로그램을 작성하자.

```
c:\>python sys2.py life is too short, you need python
```



A screenshot of a Windows command prompt window. The title bar shows 'C:\Windows\System32\cmd.exe'. The command prompt shows the following sequence of text: 'C:\test>python sys2.py life is too short, you need python', followed by the output 'LIFE IS TOO SHORT, YOU NEED PYTHON', and then the prompt 'C:\test>'.

```
C:\test>python sys2.py life is too short, you need python
LIFE IS TOO SHORT, YOU NEED PYTHON
C:\test>
```

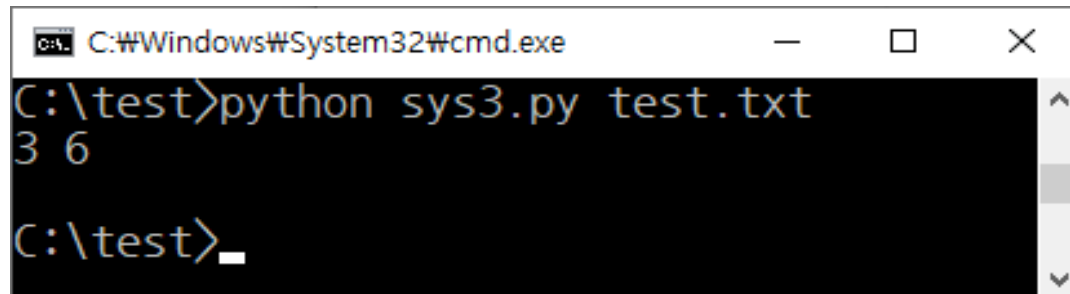
실습문제

- 파일의 라인 수, 단어 수를 구하여 출력하는 프로그램을 작성하시오.(단, 단어는 공백으로 구분된 문자열을 의미한다고 가정)

```
c:\>python sys3.py test.txt
```

test.txt

```
first line  
second line  
third line
```



```
C:\Windows\System32\cmd.exe  
C:\test>python sys3.py test.txt  
3 6  
C:\test>_
```