세상의 속도를
따라잡고 싶다면

점프 투 파이썬

박응용 지음(위키독스 운영자)

이지스퍼블리싱

04

파이썬의 입출력

04-2 사용자 입출력

▪ 사용자 입력 활용하기

▪ input 사용하기

▪ 사용자가 키보드로 입력한 모든 것을 문자열로 저장

```
>>> a = input()
Life is too short, you need python  <-- 사용자가 문장을 입력
>>> a
'Life is too short, you need python'
```

04-2 사용자 입출력

▪ 사용자 입력 활용하기

- 프롬프트를 띄워 사용자 입력받기
  - 사용자에게 입력받을 때 안내 문구 또는 질문을 보여 주고 싶을 때

```
input("안내_문구")
```

```
>>> number = input("숫자를 입력하세요: ")
숫자를 입력하세요:
```

```
>>> number = input("숫자를 입력하세요: ")
숫자를 입력하세요: 3 <-- 3 입력
>>> print(number)
3
```

04-2 사용자 입출력

print 자세히 알기

- 데이터를 출력하는 데 사용

```
>>> a = 123
>>> print(a) <- 숫자 출력하기
123
>>> a = "Python"
>>> print(a) <- 문자열 출력하기
Python
>>> a = [1, 2, 3]
>>> print(a) <- 리스트 출력하기
[1, 2, 3]
```

04-2 사용자 입출력

print 자세히 알기

- 큰따옴표로 둘러싸인 문자열은 + 연산과 동일하다

```
>>> print("life" "is" "too short") <-①
lifeistoo short
>>> print("life"+"is"+"too short") <-②
lifeistoo short
```

- 문자열 띄어쓰기는 쉼표로 한다

```
>>> print("life", "is", "too short")
life is too short
```

- 한 줄에 결괏값 출력하기

```
>>> for i in range(10):
...     print(i, end = ' ')
...
0 1 2 3 4 5 6 7 8 9 >>>
```

04-3 파일 읽고 쓰기

▪ 파일 생성하기

- 사용자가 직접 '입력'하고 모니터 화면에 결괏값을 '출력'하는 방법만 있는 것은 아님
- 파일을 통한 입출력도 가능

```
f = open("새파일.txt", 'w')
f.close()
```

- 소스코드를 실행하면 프로그램을 실행한 디렉터리에 새로운 파일이 하나 생성됨
- 파일을 생성하기 위해 파이썬 내장 함수 open을 사용한 것

```
파일_객체 = open(파일_이름, 파일_열기_모드)
```

- f.close( )는 열려 있는 파일 객체를 닫아 주는 역할(생략 가능하지만 사용하는 것을 추천)

04-3 파일 읽고 쓰기

▪ 파일 생성하기

▪ 파일 열기 모드

파일 열기 모드	설명
r	읽기 모드: 파일을 읽기만 할 때 사용한다.
w	쓰기 모드: 파일에 내용을 쓸 때 사용한다.
a	추가 모드: 파일의 마지막에 새로운 내용을 추가할 때 사용한다.

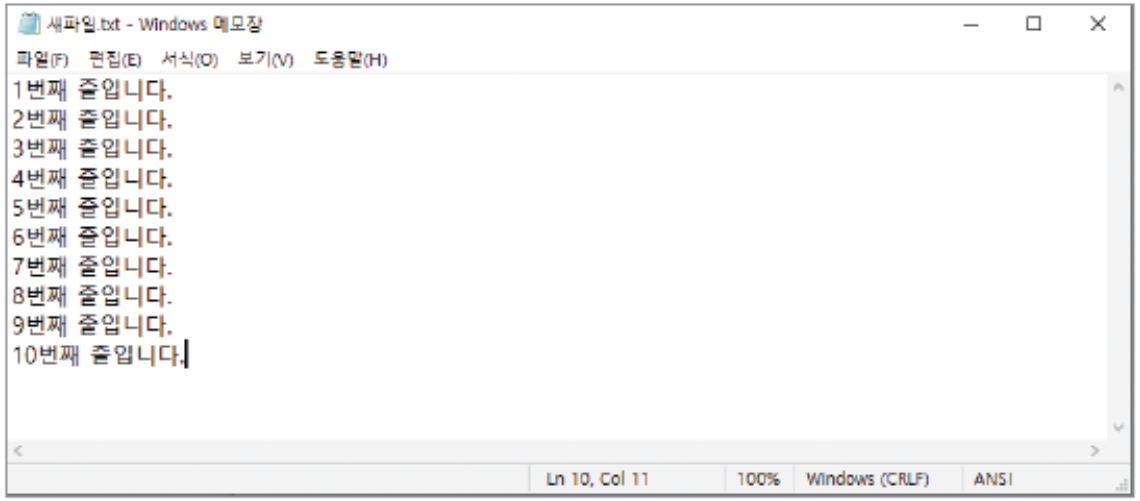
- 파일을 쓰기 모드(w)로 열면 해당 파일이 이미 존재할 경우 원래 있던 내용이 모두 사라지고, 해당 파일이 존재하지 않으면 새로운 파일이 생성됨

04-3 파일 읽고 쓰기

▪ 파일을 쓰기 모드로 열어 내용 쓰기

▪ 문자열 데이터를 파일에 직접 써서 출력

```
f = open("C:/doit/새파일.txt", 'w')
for i in range(1, 11):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)    # data를 파일 객체 f에 써라.
f.close()
```



04-3 파일 읽고 쓰기

▪ 파일을 읽는 여러 가지 방법

- readline 함수 사용하기
  - f.open("새파일.txt", 'r')로 파일을 읽기 모드로 연 후 readline( )을 사용해서 파일의 첫 번째
  - 줄을 읽어 출력하는 코드

```
f = open("C:/doit/새파일.txt", 'r')
line = f.readline()
print(line)
f.close()
```

```
1번째 줄입니다.
```

- 모든 줄을 읽어 화면에 출력하는 코드

```
f = open("C:/doit/새파일.txt", 'r')
while True:
    line = f.readline()
    if not line: break
    print(line)
f.close()
```

  - 무한루프 안에서 f.readline( )을 사용해 파일을 계속 한 줄씩 읽어 들임
  - 더 이상 읽을 줄이 없으면 break 수행
  - readline( )은 더 이상 읽을 줄이 없을 경우 빈 문자열('')을 리턴

04-3 파일 읽고 쓰기

▪ 파일을 읽는 여러 가지 방법

- readlines 함수 사용하기
  - 파일의 모든 줄을 읽어서 각각의 줄을 요소로 가지는 리스트를 리턴
  - ["1번째 줄입니다.₩n", "2번째 줄입니다.₩n", ..., "10번째 줄입니다.₩n"]를 리턴

```
f = open("C:/doit/새파일.txt", 'r')
lines = f.readlines()
for line in lines:
    print(line)
f.close()
```

04-3 파일 읽고 쓰기

▪ 파일을 읽는 여러 가지 방법

- read 함수 사용하기
  - f.read( )는 파일의 내용 전체를 문자열로 리턴
  - data는 파일의 전체 내용

```
f = open("C:/doit/새파일.txt", 'r')
data = f.read()
print(data)
f.close()
```

- 파일 객체를 for 문과 함께 사용하기
  - 파일 객체(f)는 for 문과 함께 사용하여 파일을 줄 단위로 읽을 수 있음

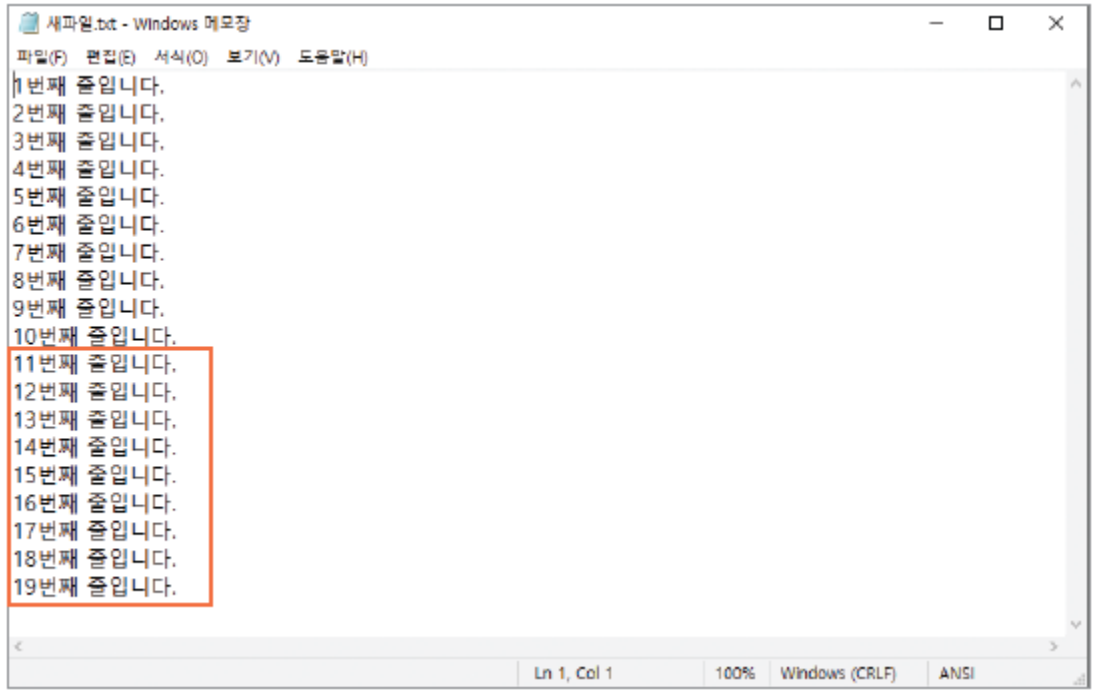
```
f = open("C:/doit/새파일.txt", 'r')
for line in f:
    print(line)
f.close()
```

04-3 파일 읽고 쓰기

▪ 파일에 새로운 내용 추가하기

- 원래 있던 값을 유지하면서 단지 새로운 값만 추가해야 할 경우
- 파일을 추가 모드('a')로 열기

```
f = open("C:/doit/새파일.txt", 'a')
for i in range(11, 20):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()
```



04-3 파일 읽고 쓰기

with 문과 함께 사용하기

- 지금까지 파일을 열고 닫은 방법

```
f = open("foo.txt", 'w') <-- 파일 열기
f.write("Life is too short, you need python")
f.close() <-- 파일 닫기
```

  - f.close( )는 열려 있는 파일 객체를 닫아 주는 역할
  - 쓰기 모드로 열었던 파일을 닫지 않고 다시 사용하면 오류가 발생하기 때문에, close( )를 사용해서 열려 있는 파일을 직접 닫아 주는 것이 좋음

04-3 파일 읽고 쓰기

with 문과 함께 사용하기

- with 문은 파일을 열고 닫는 것을 자동으로 처리해주는 문법
- 앞선 예제를 with 문을 사용하여 수정한 코드

```
with open("foo.txt", "w") as f:
    f.write("Life is too short, you need python")
```

- with 문을 사용하면 with 블록을 벗어나는 순간 열린 파일 객체 f가 자동으로 닫힘

04-4 프로그램의 입출력

sys 모듈 사용하기

- 파이썬에서는 sys 모듈을 사용하여 프로그램에 인수 전달 가능
- import 명령어 사용

```
import sys

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

- argv는 프로그램 실행 시 전달된 인수

- argv[0]은 파일 이름 sys1.py, argv[1]부터는 뒤에 따라오는 인수가 차례대로 argv의 요소

04-4 프로그램의 입출력

sys 모듈 사용하기

- 전달된 인수를 모두 대문자로 바꾸는 간단한 프로그램 만들기

```
import sys
args = sys.argv[1:]
for i in args:
    print(i.upper(), end=' ')
```

- 명령 프롬프트

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

- 실행 결과

```
LIFE IS TOO SHORT, YOU NEED PYTHON
```

감사합니다.