

## ■ 사용자 입력 (input)

### ● 사용자가 입력한 값을 문자열로 저장

```
text = input()
print(text)
print(type(text))
```

```
Python 1234
Python 1234
<class 'str'>
```

### ● 안내 문구 출력 후 사용자 입력

```
text = input('내용을 입력해주세요.')
print(text)
print(type(text))
```

```
내용을 입력해주세요. > Python 1234
Python 1234
<class 'str'>
```

## ■ 출력 (print)

### ● 특정 값을 화면에 출력

```
num = 123
print(num)

s = 'python'
print(s)

list = [1, 2, 'a', 'b']
print(list)

tuple = (1, 2, 'a', 'b')
print(tuple)
```

```
123
python
[1, 2, 'a', 'b']
(1, 2, 'a', 'b')
```

### ● space, comma, plus 연산자 사용

```
print('Life' 'is' 'too' 'short')

print('Life' + 'is' + 'too' + 'short')

print('Life ' 'is ' 'too ' 'short')

print('Life', 'is', 'too', 'short')
```

```
Lifeistooshort
Lifeistooshort
Life is too short
Life is too short
```

## ■ 출력 (print)

### ● space (따옴표 사용)

```
a = '점수'
b = 100

print(a b) # space
```

File "<ipython-input-20-912105bc5c57>", line 4  
print(a b) # space  
 ^

SyntaxError: invalid syntax



```
print('내용' '내용') # space
```

내용내용

### ● plus (문자열 사용)

```
a = '점수'
b = 100

print(a + b) # plus
```

TypeError Traceback (most recent call last)  
<ipython-input-21-6f0d161d3c93> in <module>()  
 2 b = 100  
 3  
----> 4 print(a + b) # plus

TypeError: must be str, not int



```
a = '점수'
b = 100

print(a + str(b)) # plus
```

점수100

## ■ 출력

### ● print

- 결과값을 한줄로 출력 (줄바꿈 X)

```
print('Life', end=' ')\n\nprint('is', end=' ')\n\nprint('too', end=' ')\n\nprint('short', end=' ')
```

Life is too short

- 반복문 결과 한줄 출력

```
for i in range(1, 11):\n    print(str(i) + '번째', end = ' ')
```

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

## ■ 파일

### ● 생성 / 열기 : open()

– 파일객체 = open('파일명', '모드')

모드	설명
r	읽기 – 파일의 내용을 읽기만 할때 사용 (기본값)
w / x	쓰기 – 파일의 내용을 쓰기만 할때 사용
a	추가 – 파일의 마지막 부분에 내용을 추가할때 사용
+	읽기/쓰기 + 텍스트/바이너리
t	텍스트 모드 (기본값)
b	바이너리(이진) 모드

## ■ 파일

### ● 읽기 모드 (r)

- 대상 파일이 없는 경우 오류

```
file = open('d:/test.txt', 'r')
```

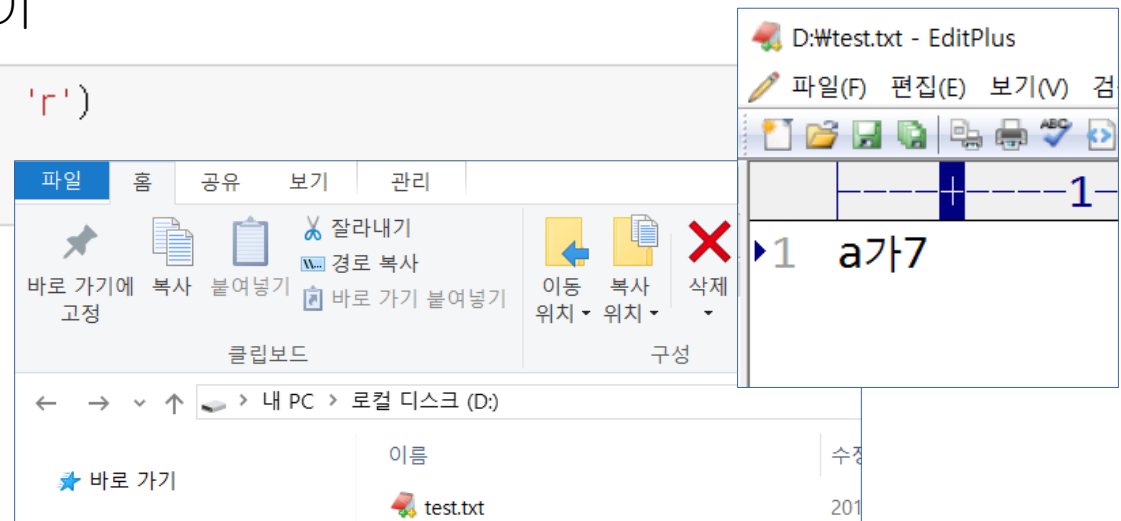
```
FileNotFoundError Traceback (most recent call last)
<ipython-input-52-882eaff4960f> in <module>()
----> 1 file = open('d:/test.txt', 'r')
```

**FileNotFoundError:** [Errno 2] No such file or directory: 'd:/test.txt'

- 작업 완료 후 파일 닫기

```
file = open('d:/test.txt', 'r')
print(file.read())
file.close()
```

a가7

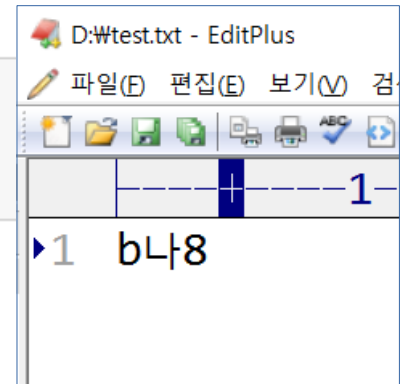


## ■ 파일

### ● 쓰기 모드 (w)

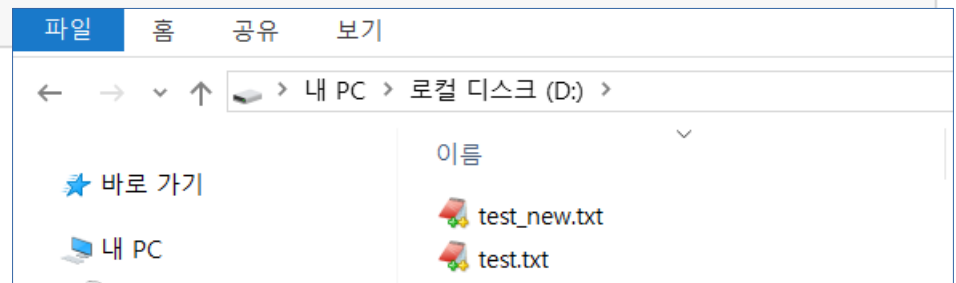
- 대상 파일 내용 삭제하고 새로 작성

```
file = open('d:/test.txt', 'w')  
file.write('b나8')  
file.close()
```



- 대상 파일이 없는 경우 새 파일을 생성 후 내용 입력

```
file = open('d:/test_new.txt', 'w')  
file.write('b나8')  
file.close()
```



## ■ 파일

### ● 쓰기 모드 (x)

- 대상 파일이 있는 경우 오류

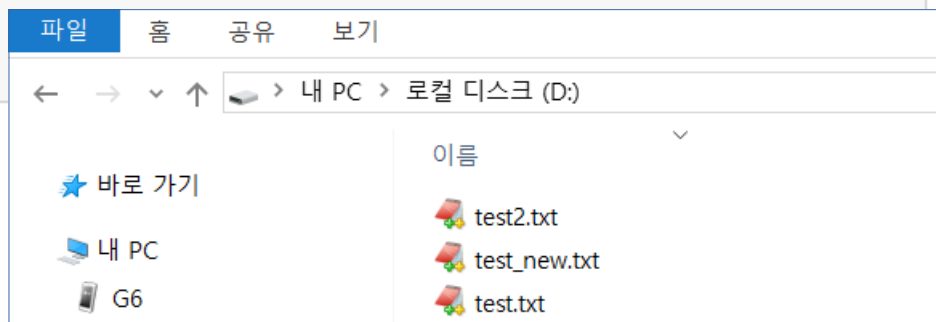
```
file = open('d:/test.txt', 'x')
file.write('Hello')
file.close()
```

```
FileExistsError                                Traceback (most recent call last)
<ipython-input-58-d3b63aa414b5> in <module>()
----> 1 file = open('d:/test.txt', 'x')
      2 file.write('Hello')
      3 file.close()
```

**FileExistsError:** [Errno 17] File exists: 'd:/test.txt'

- 대상 파일이 없는 경우 새 파일을 생성 후 내용 입력

```
file = open('d:/test2.txt', 'x')
file.write('Hello')
file.close()
```



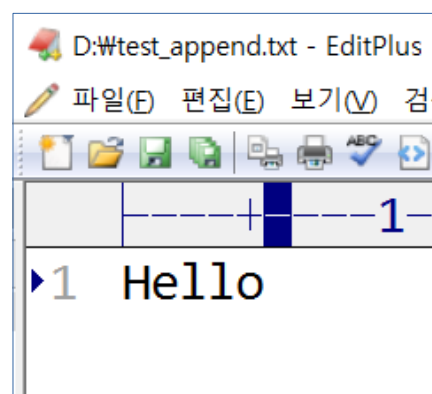
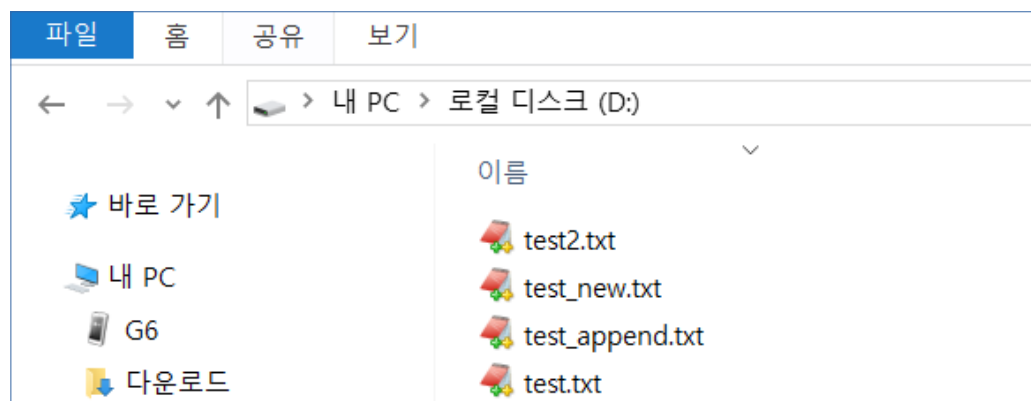


## ■ 파일

### ● 추가 모드 (a)

- 대상 파일 내용의 끝 부분에 내용 추가
- 대상 파일이 없는 경우 새 파일을 생성 후 내용 입력

```
file = open('d:/test_append.txt', 'a')  
file.write('Hello')  
file.close()
```



## ■ 파일

### ● 바이너리 모드 (wb/rb) – List

```
file = open('d:/list.bin', 'wb')  
list = [10, 20, 30, 40, 50] # 정수만 가능  
file.write(bytes(list))  
file.close()
```

```
file = open('d:/list.bin', 'rb')  
data = file.read()  
print(type(data))  
for d in data:  
    print(d)  
file.close()
```

```
<class 'bytes'>  
10  
20  
30  
40  
50
```

### ● 바이너리 모드 (wb/rb) – Tuple

```
file = open('d:/tuple.bin', 'wb')  
t = (10, 20, 30, 40, 50) # 정수만 가능  
file.write(bytes(t))  
file.close()
```

```
file = open('d:/tuple.bin', 'rb')  
data = file.read()  
print(type(data))  
for d in data:  
    print(d)  
file.close()
```

```
<class 'bytes'>  
10  
20  
30  
40  
50
```

## ■ 파일

### ● 바이너리 모드 (wb/rb) – Dictionary

```
file = open('d:/dictionary.bin', 'wb')
d = {1:10, 2:20, 3:30, 3:40}
file.write(bytes(d))
file.close()
```

```
file = open('d:/dictionary.bin', 'rb')
data = file.read()
print(type(data))
for d in data:
    print(d)
file.close()
```

<class 'bytes'>

1  
2  
3

### ● 바이너리 모드 (wb/rb) – Set

```
file = open('d:/set.bin', 'wb')
s = {1, 2, 3, 4, 5}
file.write(bytes(s))
file.close()
```

```
file = open('d:/set.bin', 'rb')
data = file.read()
l = list(data)
print(l)
file.close()
```

[1, 2, 3, 4, 5]

## ■ 파일

### ● 바이너리 모드 (wb/rb) – Object 형태

```
class Data:
    name = 'ggoreb'
    age = 30

    def __str__(self):
        return 'Data ' + self.name + ', ' + str(self.age)

d = Data()

import pickle

file = open('d:/object.bin', 'wb')

pickle.dump(d, file)

file.close()

file = open('d:/object.bin', 'rb')
d = pickle.load(file)
print(d)

file.close()
```

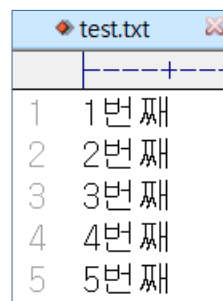
Data ggoreb, 30

## ■ 파일

### ● 쓰기 : write()

– 쓰기모드로 파일을 연 후 내용을 출력

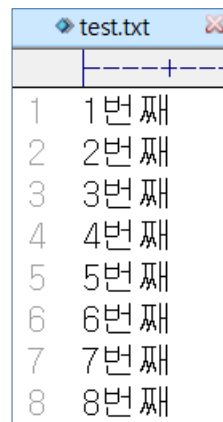
```
file = open('D:/test.txt', 'w')  
  
for i in range(1, 6):  
    data = '%d번째\n' % i  
    file.write(data)  
  
file.close()
```



1	1번째
2	2번째
3	3번째
4	4번째
5	5번째

– 추가모드로 파일을 연 후 내용을 출력

```
file = open('D:/test.txt', 'a')  
  
for i in range(6, 9):  
    data = '%d번째\n' % i  
    file.write(data)  
  
file.close()
```

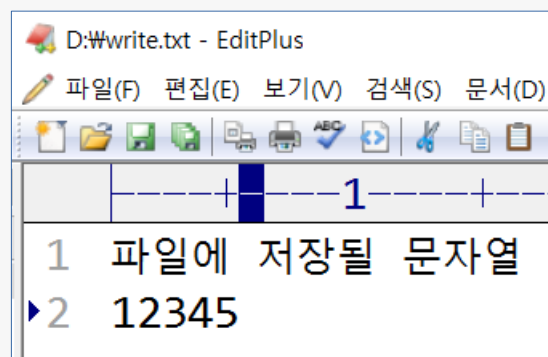


1	1번째
2	2번째
3	3번째
4	4번째
5	5번째
6	6번째
7	7번째
8	8번째

## ■ 파일

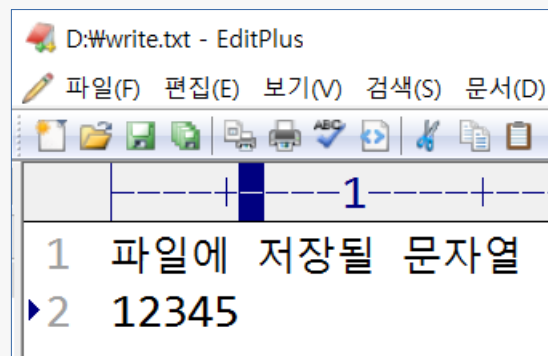
### ● 쓰기 : write()

```
file = open('d:/write.txt', 'w')  
file.write('파일에 저장될 문자열\n')  
  
numbers = ['1', '2', '3', '4', '5']  
for num in numbers:  
    file.write(num)  
  
file.close()
```



### ● 쓰기 : writelines()

```
file = open('d:/write.txt', 'w')  
file.write('파일에 저장될 문자열\n')  
  
numbers = ['1', '2', '3', '4', '5']  
file.writelines(numbers)  
  
file.close()
```



## ■ 파일

### ● 읽기 : read()

– 파일의 모든 내용을 읽음

```
file = open('D:/test.txt', 'r')  
  
data = file.read()  
print(data)  
  
file.close()
```

```
1호 파일  
2호 파일  
3호 파일  
4호 파일  
5호 파일  
6호 파일  
7호 파일  
8호 파일
```

## ■ 파일

### ● 읽기 : readline()

- 줄바꿈 문자까지(**한줄만**) 내용 읽음

```
file = open('D:/test.txt', 'r')  
  
data = file.readline()  
print(data)  
  
file.close()
```

1번째

- 내용이 읽히지 않을때까지(**마지막줄까지**) 내용 읽음

```
file = open('D:/test.txt', 'r')  
  
while True:  
    data = file.readline()  
    if not data:  
        break  
  
    print(data)  
  
file.close()
```

1번째  
2번째  
3번째  
4번째  
5번째  
6번째  
7번째  
8번째



## ■ 파일

### ● 읽기 : readlines()

- 줄바꿈 문자를 기준으로 모든 내용을 리스트 요소로 생성

```
file = open('D:/test.txt', 'r')  
  
data = file.readlines()  
print(data)  
  
file.close()
```

```
['1번째\n', '2번째\n', '3번째\n',  
 '4번째\n', '5번째\n', '6번째\n',  
 '7번째\n', '8번째\n']
```

- 모든 내용을 리스트로 읽은 후 요소별 출력

```
file = open('D:/test.txt', 'r')  
  
data = file.readlines()  
  
for line in data:  
    print(line)  
  
file.close()
```

```
1번째  
2번째  
3번째  
4번째  
5번째  
6번째  
7번째  
8번째
```

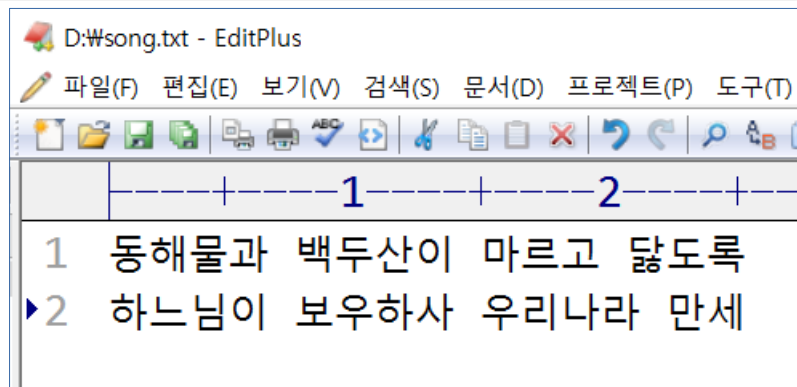
## ■ 파일

### ● 파일 내용 중 현재 위치 확인

- 영어 / 숫자 : 1Byte
- 한글, Wn 등 특수/예외 문자 : 2Byte

```
file = open('d:/song.txt', 'r+t')  
  
song = file.read()  
print(song)  
  
nowidx = file.tell()  
# 총 문자수 4*7 = 28*2 = 56 / 띄어쓰기 6 / 공백문자 2  
print(nowidx)  
  
file.close()
```

동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세  
64



## ■ 파일

### ● 파일 내용 중 현재 위치 변경

```
file = open('d:/song.txt', 'r+t')

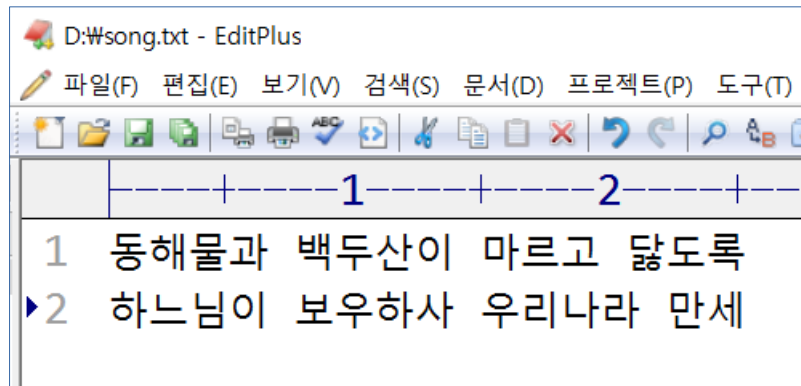
song = file.read()
print(song)

nowidx = file.tell()
# 총 문자수 4*7 = 28*2 = 56 / 띄어쓰기 6 / 공백문자 2
print('현재 위치', nowidx)

nowidx = file.seek(0)
print('현재 위치', nowidx)

file.close()
```

동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세  
현재 위치 64  
현재 위치 0



## ■ 파일

### ● 자동 close

```
file = open('d:/song.txt', 'r+t')  
  
song = file.read()  
print(song)  
  
file.close()
```

동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세



```
file = open('d:/song.txt', 'r+t')  
  
with file:  
    song = file.read()  
    print(song)
```

동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세

```
with open('d:/song.txt', 'r+t') as file:  
    song = file.read()  
    print(song)
```

동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세

## ■ 파일

### ● with

- 파이썬의 프로그램은 `__enter__`와 `__exit__` 메소드를 가질 수 있는데  
with 블록의 시작과 끝 부분에서 자동으로 실행됨
- 파이썬의 클래스 코드

```
class MyFile:
```

```
    def __init__(self):  
        print('MyFile 만들어짐')
```

클래스가 생성될 때 자동으로 실행되는 메소드

```
mf = MyFile()
```

MyFile 만들어짐

```
class MyFile:
```

```
    def __init__(self):  
        print('MyFile 만들어짐')
```

```
    def __enter__(self):  
        print('열림')
```

```
    def __exit__(self, type, value, traceback):  
        print('닫힘')
```

```
mf = MyFile()
```

MyFile 만들어짐

## ■ 파일

### ● with

– with 구문 사용 : 자동으로 enter와 exit 메소드 호출

```
class MyFile:
    def __init__(self):
        print('MyFile 만들어짐')
    def __enter__(self):
        print('열림')
    def __exit__(self, type, value, traceback):
        print('닫힘')

with MyFile() as mf:
    pass
```

MyFile 만들어짐

열림

닫힘

## ■ 파일

### ● with

- with 구문 사용 : enter / exit 메소드를 작성하지 않고 사용하는 경우 오류

```
class MyFile:
    def __init__(self):
        print('MyFile 만들어짐')

with MyFile() as mf:
    pass
```

MyFile 만들어짐

---

```
AttributeError                                Traceback (most recent call last)
<ipython-input-43-dec591ec81ab> in <module>()
      3         print('MyFile 만들어짐')
      4
--> 5 with MyFile() as mf:
      6     pass
```

AttributeError: \_\_enter\_\_