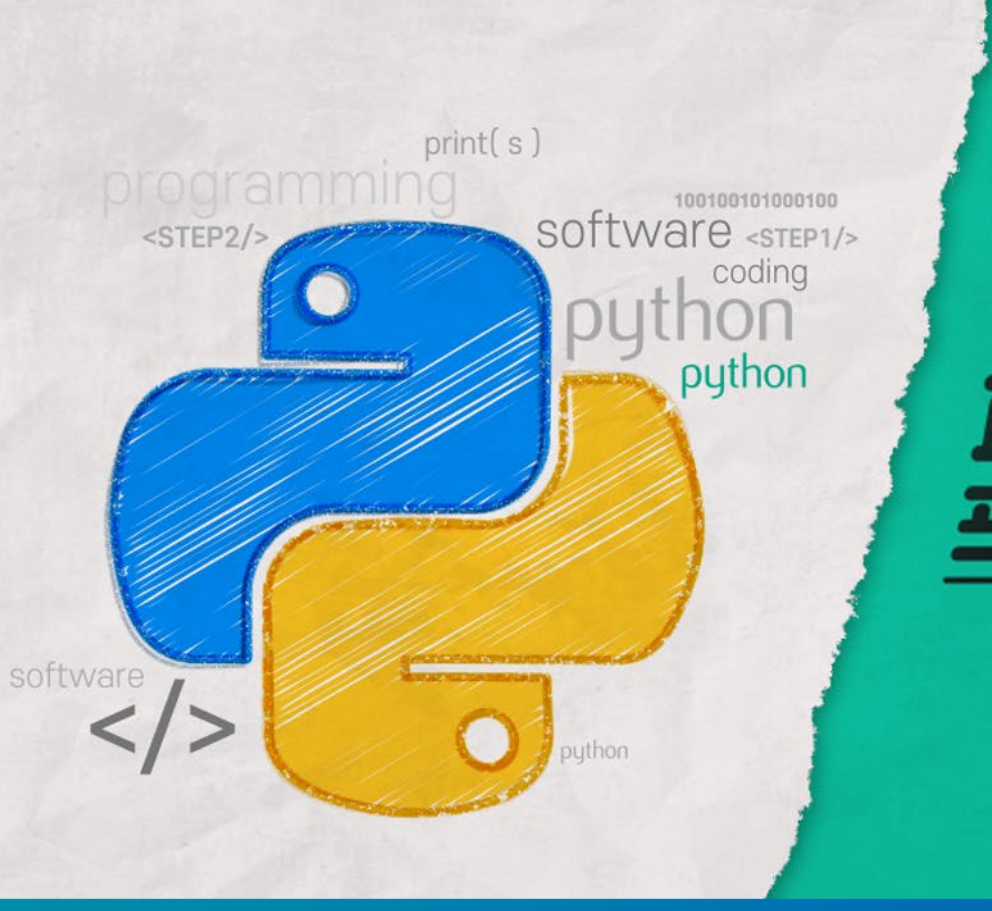


파이썬 </>
프로그래밍

파이썬의 파일 입출력

OLEI
한국기술교육대학교
온라인평생교육원

학습목표

- **파일 입출력 방법**에 대해 알고, 프로그램 실행 결과를 파일로 입출력할 수 있다.
- **파일 입출력을 활용할 수 있는 예**를 살펴보고, 기본적인 코딩을 할 수 있다.
- **파이썬에서 파일 및 디렉토리**를 다룰 수 있다.

학습내용

- 파일 입출력
- 파일 입출력의 활용
- 파일 및 디렉토리 다루기

파일 입출력

1 파일 입출력이란?

파일을 열어서 텍스트를 읽고, 쓰고, 수정하는 방법

- 파일을 열 수 있음 (파일 객체 반환)
- 열어둔 파일을 닫을 수 있음

- 파일 처리 후 close()를 통해 닫아주어야만 자원 점유를 해제하고 불필요한 오류 발생을 막을 수 있음

파일 입출력

1 파일 입출력이란?

파일 처리 모드의 종류

모드	의미	비고
r	읽기 모드	• 파일 객체를 읽기 모드로 생성 • 파일의 처음 위치로 포인터를 이동
w	쓰기 모드	• 파일을 쓰기 모드로 엶 • 파일에 데이터를 쓰면 기존 파일의 내용은 모두 사라짐 • 주어진 파일이 존재하지 않으면 새로운 파일을 만듦
x	쓰기 전용	• 새 파일 쓰기 모드로 엶 • 주어진 이름의 파일이 존재하면 에러 발생
a	추가 모드	• 파일을 추가 모드로 엶 • 기존 파일의 내용의 끝에 새 내용을 추가하여 기록
+	갱신 모드	• 파일을 읽기와 쓰기가 모두 가능한 모드로 엶

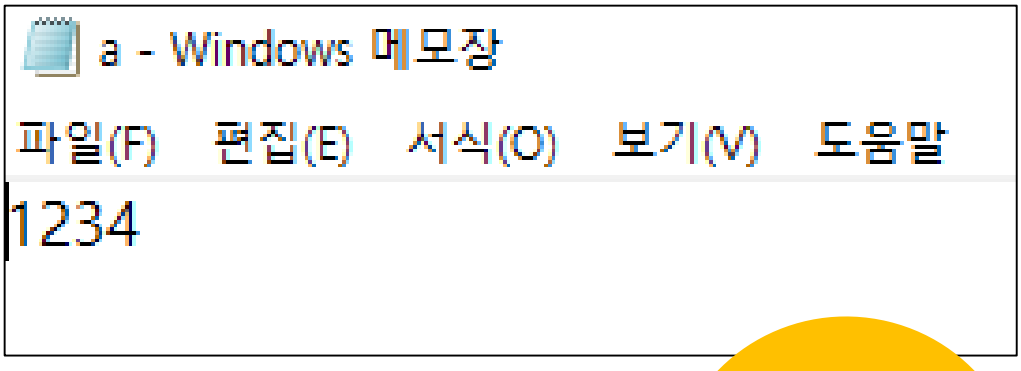
파일 입출력

2 파일 쓰기

파일명으로 파일을 생성 후 **write 함수**를 활용해 내용 작성 가능

- 디렉토리 경로 없이 파일명만 적은 경우 현재 해당 파이썬이 실행되는 경로에 파일 생성

```
f = open("a.txt",'w')
f.write("1234")
f.close()
```




```
import os
print(os.getcwd())
```

```
C:₩Users₩zoost₩Downloads
```

os라는 모듈을 활용해 현재 디렉토리 경로 확인 가능

파일 입출력

2 파일 쓰기

특정 경로에 파일을 생성하고 싶을 때는
전체 경로 및 파일명 입력

- **₩는 이스케이프 문자로 경로 설정 시 주의!**

```
f = open("C:₩₩Users₩₩zoost₩₩Downloads₩₩a.txt",'w')
```

```
f = open("C:₩Users₩zoost₩Downloads₩a.txt",'w')
```

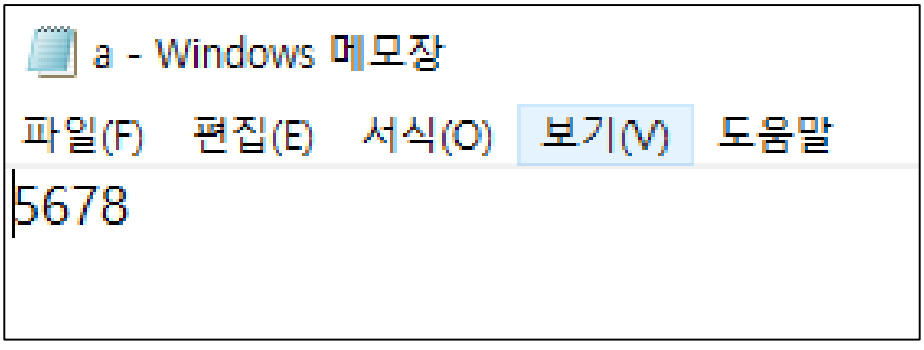
```
  File "<ipython-input-2-41848dcf1e8c>", line 1
    f = open("C:₩Users₩zoost₩Downloads₩a.txt",'w')
                                                  ^
SyntaxError: (unicode error) 'unicodeescape' codec can't de
```

파일 입출력

2 파일 쓰기

w 모드일 경우 파일을 새로 생성하며, 기존에 파일이 있다면 덮어쓰기 때문에 기존 내용이 사라짐

```
f = open("a.txt",'w')
f.write("1234")
f.close()
f = open("a.txt",'w')
f.write("5678")
f.close()
```



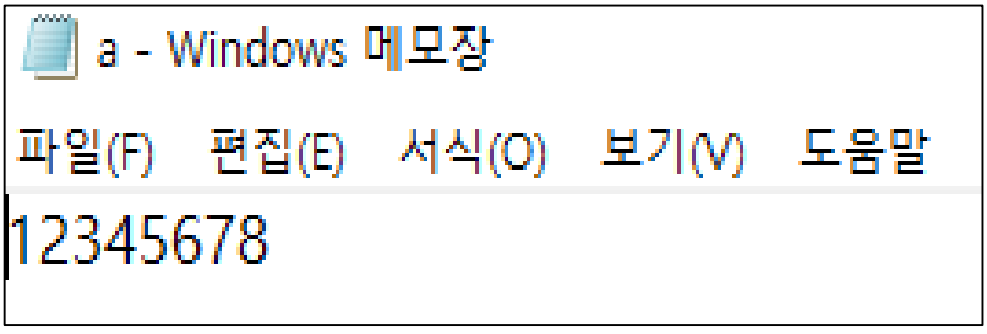
파일 입출력

2 파일 쓰기

기존 파일에 새로운 내용을 추가하기 위해서는 a 모드를 사용

- **해당 파일명으로 파일이 없더라도 a 모드로 생성 가능**

```
f = open("a.txt",'w')
f.write("1234")
f.close()
f = open("a.txt",'a')
f.write("5678")
f.close()
```



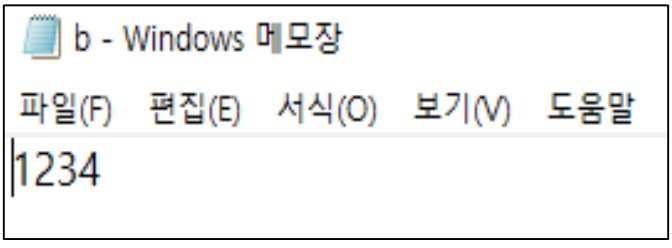
파일 입출력

2 파일 쓰기

새로운 파일을 쓰기 위해 x 모드도 사용 가능

- w 모드와 기능은 똑같지만, x 모드는 기존 파일이 있다면 오류 발생(덮어쓰기 방지)

```
f = open("b.txt",'x')
f.write("1234")
f.close()
```




```
f = open("a.txt",'x')
f.write("1234")
f.close()

---------------------------------------------
FileExistsError                         Traceba
<ipython-input-12-ebef3e9d473c> in <module>
----> 1 f = open("a.txt",'x')
      2 f.write("5678")
      3 f.close()

FileExistsError: [Errno 17] File exists: 'a.txt'
```

파일 입출력

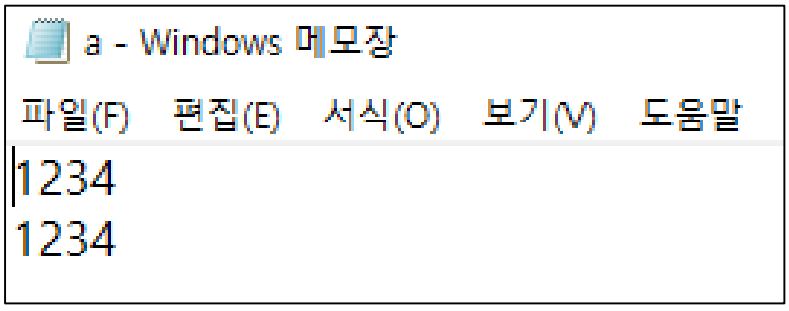
2 파일 쓰기

여러 줄의 내용을 입력하는 방법

- **여러 줄 문자열** 사용(따옴표 세 개)
- **개행 문자** 사용

```
f = open("a.txt",'w')
f.write("""1234
1234""")
f.close()
```

```
f = open("a.txt",'w')
f.write("1234₩n1234")
f.close()
```



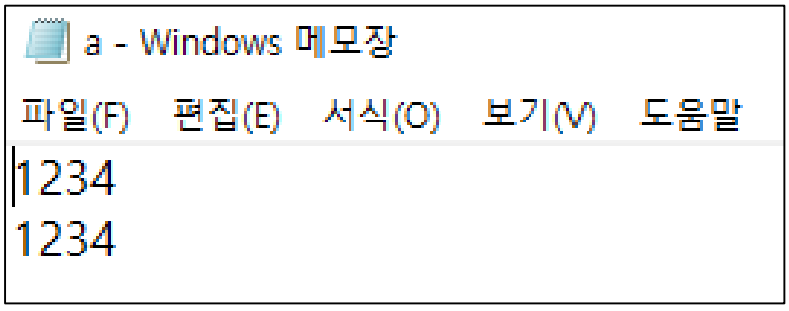
파일 입출력

2 파일 쓰기

리스트, 튜플 등의 내용을 입력하는 방법

- **writelines() 함수 사용**

```
t = ("1","2","3","4","₩n")
l = ["1","2","3","4"]
f = open("a.txt",'w')
f.writelines(t)
f.writelines(l)
f.close()
```



파일 입출력

3 파일 읽기

r 모드로 파일 열기, read() 함수로 파일의 전체 내용 불러오기

- **읽으려는 파일이 없으면 오류 발생**

```
f = open("a.txt",'w')
f.write("1234")
f.close()
f = open("a.txt",'r')
print(f.read())
f.close()
```

```
1234
```

```
f = open("b.txt",'r')
print(f.read())
f.close()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recen
<ipython-input-18-c5705a295a4f> in <module>
----> 1 f = open("b.txt",'r')
      2 print(f.read())
      3 f.close()

FileNotFoundError: [Errno 2] No such file or directory: 'b.txt'
```

파일 입출력

3 파일 읽기

readline() 함수로 파일의 내용을 한 줄씩 가져올 수 있음

- 더 이상 읽을 줄이 없다면 None()을 반환
- while + if 제어문을 사용하여 모든 줄 출력 가능

```
f = open("a.txt",'r')
print(f.readline())
print(f.readline())
f.close()
```

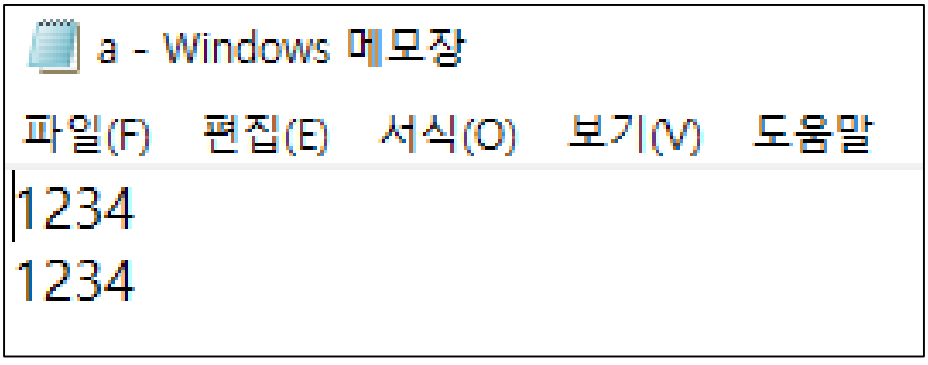
```
1234

1234
```

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

```
1234

1234
```



파일 입출력

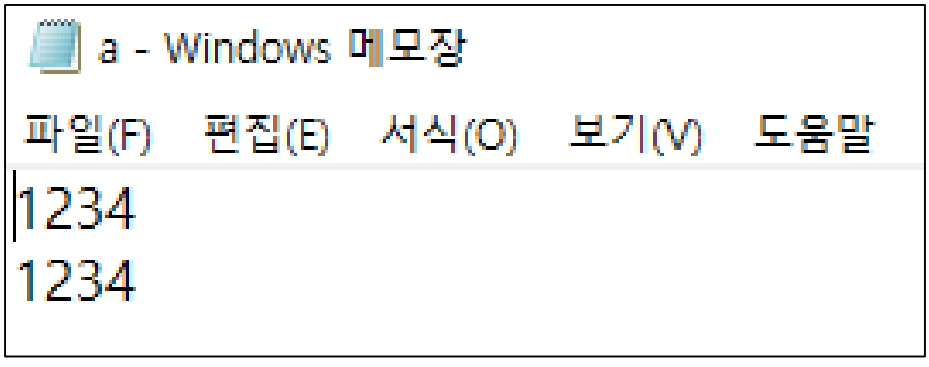
3 파일 읽기

readlines() 함수로 파일의 내용을 리스트로 가져올 수 있음

- 주로 반복문과 함께 사용해 한 줄씩 리스트의 요소로 가져와 활용

```
f = open("a.txt",'r')
print(f.readlines())
f.close()
```

```
['1234₩n', '1234']
```



파일 입출력의 활용

1 표준 출력 전환

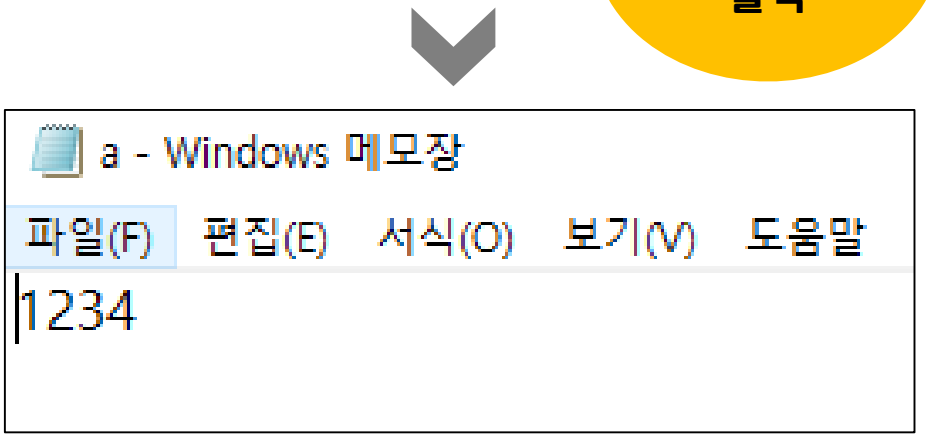
파이썬의 표준 출력은 print() 함수를 활용해 파이썬 쉘 환경(콘솔)에 출력

- 파이썬의 sys 모듈을 활용해 표준 출력을 파일로 전환 가능

sys.stdout	sys.stdin
표준 출력	표준 입력

```
import sys
f = open('a.txt', 'w')
sys.stdout = f
print("1234")
f.close()
```

print() 내용이 생성한 파일로 출력



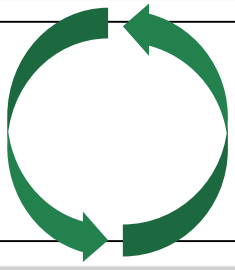
파일 입출력의 활용

1 표준 출력 전환

로그, 에러 등을 기록할 때 활용 가능

- 표준 출력을 잠시 다른 변수에 저장해두고 필요할 때 콘솔로 되돌려서 사용

```
import sys
stdout = sys.stdout
sys.stdout = 파일
```



```
sys.stdout = stdout
```

파일 입출력의 활용

2 다른 자료형의 파일 입출력

기존 파일 입출력은 단순 텍스트만 파일로 입출력 가능

- **다른 자료형의 객체의 형태를 그대로 유지하면서 파일에 저장하기 위해 pickle 모듈 활용**

```
f = open("a.txt",'w')
f.write("[1,2,3]")
f.close()
f = open("a.txt",'r')
a = f.read()
f.close()
print(a)
print(type(a))
```

```
[1,2,3]
<class 'str'>
```

```
f = open("a.txt",'w')
f.write("{'a': 1, 'b':2}")
f.close()
f = open("a.txt",'r')
a = f.read()
f.close()
print(a)
print(type(a))
```

```
{'a': 1, 'b':2}
<class 'str'>
```

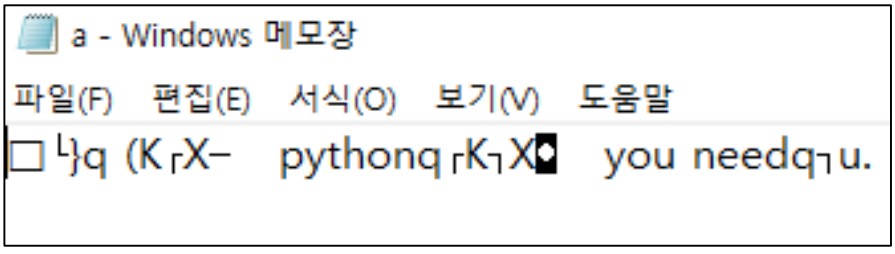
파일 입출력의 활용

2 다른 자료형의 파일 입출력

pickle 모듈로 파일을 저장할 때는 **바이너리 형식**으로 입출력해야 함(wb, rb 모드)

- 파이썬의 모든 객체들을 그대로 저장 가능

```
import pickle
f = open("a.txt", 'wb')
data = {1: 'python', 2: 'you need'}
pickle.dump(data, f)
f.close()
```




```
f = open("a.txt", 'rb')
data_read = pickle.load(f)
f.close()
print(data_read)
print(type(data_read))
```

```
{1: 'python', 2: 'you need'}
<class 'dict'>
```

파일 및 디렉토리 다루기

1 파일 다루기

시스템의 환경 변수, 디렉토리, 파일 등을 제어할 수 있는 파이썬의 OS 모듈을 활용

1 listdir 함수로 해당 디렉토리의 파일 목록 반환

```
print (os.listdir('.'))
```

```
['.ipynb_checkpoints', 'a.txt', 'desktop.ini',
```

. 은
상대경로로
현재 디렉토리
의미

../ 은
부모 디렉토리
의미

```
print (os.listdir('../'))
```

```
['.ipython', '.jupyter', '3D Obje
wnloads', 'Favorites', 'Links',
at.LOG1', 'ntuser.dat.LOG2', 'NTU
```

파일 및 디렉토리 다루기

1 파일 다루기

2 rename 함수로 파일의 이름 변경

```
print (os.listdir('.'))
os.rename('a.txt', 'b.txt')
print (os.listdir('.'))
```

```
['.ipynb_checkpoints', 'a.txt', 'desktop.ini',
['.ipynb_checkpoints', 'b.txt', 'desktop.ini',
```

3 path.exists 함수로 파일의 존재 유무 확인

```
print(os.path.exists('a.txt'))
print(os.path.exists('b.txt'))
```

```
False
True
```

파일 및 디렉토리 다루기

1 파일 다루기

4 path.abspath 함수로 파일의 존재 유무와 관계 없이 해당 파일의 절대 경로를 반환

- 파일이 없어도 생성 가능하므로, 파일을 입력할 때 자주 활용

예 모든 사용자가 특정 위치에 해당 파일을 생성하도록 하고 싶은 경우

```
print(os.path.exists('a.txt'))
print (os.path.abspath('a.txt'))
```

```
False
C:₩Users₩zoost₩Downloads₩a.txt
```

```
print(os.path.exists('b.txt'))
print (os.path.abspath('b.txt'))
```

```
True
C:₩Users₩zoost₩Downloads₩b.txt
```

```
f = open(os.path.abspath('a.txt'), 'w')
f.write("1234")
f.close()
```

파일 및 디렉토리 다루기

1 파일 다루기

5 path.basename, dirname, split 함수로 해당 파일의 파일명과 경로명을 분리·반환

- 파일명과 파일의 경로명을 따로 분리 가능
- 환경마다 위치가 다를 경우 자주 활용

예 모두 다른 경로에 있는 파일들의 목록을 가져오거나, 새로운 파일을 만들 경우

```
print (os.path.basename("C:₩₩Users₩₩zoost₩₩Downloads₩₩b.txt"))
print (os.path.dirname("C:₩₩Users₩₩zoost₩₩Downloads₩₩b.txt"))
```

```
b.txt
C:₩Users₩zoost₩Downloads
```

```
print(os.path.split("C:₩₩Users₩₩zoost₩₩Downloads₩₩b.txt"))
```

```
('C:₩₩Users₩₩zoost₩₩Downloads', 'b.txt')
```

파일 및 디렉토리 다루기

1 파일 다루기

6 path.splitdrive, splitext 함수로 해당 파일 경로의 드라이브, 확장자를 분리·반환

- 저장된 드라이브가 다르거나, doc, docx처럼 확장자가 다를 경우 자주 활용

```
print (os.path.splitdrive("C:₩₩Users₩₩zoost₩₩Downloads₩₩b.txt"))
print (os.path.splitext("C:₩₩Users₩₩zoost₩₩Downloads₩₩b.txt"))
```

```
('C:', '₩₩Users₩₩zoost₩₩Downloads₩₩b.txt')
('C:₩₩Users₩₩zoost₩₩Downloads₩₩b', '.txt')
```

파일 및 디렉토리 다루기

2 디렉토리 다루기

시스템의 환경 변수, 디렉토리, 파일 등을 제어할 수 있는 파이썬의 OS 모듈을 활용

1 getcwd() 함수로 현재 작업 중인 디렉토리를 반환

- cwd : Current Working Directory의 약자

```
print (os.getcwd())
```

```
C:₩Users₩zoost₩Downloads
```

파일 및 디렉토리 다루기

2 디렉토리 다루기

2 chdir() 함수로 현재 작업 중인 디렉토리 경로를 변경

- chdir : Change Directory의 약자

예 파일 입출력 코드가 모두 상대 경로로 코딩되어 있는 경우 작업 환경이 달라지면 오류가 발생할 수 있음
➡ 작업 디렉토리 경로를 적절하게 변경하여 활용

```
os.chdir('C:₩₩Users₩₩zoost₩₩Desktop')
print (os.getcwd())
```

```
C:₩Users₩zoost₩Desktop
```

파일 및 디렉토리 다루기

2 디렉토리 다루기

3 mkdir() 함수로 새로운 폴더 생성

- mkdir : Make Directory의 약자

```
os.mkdir('temp')
print(os.path.exists('temp'))
```

```
True
```

Run! 프로그래밍

Mission 1 표준 출력을 파일로 변경하여 구구단을 파일에 저장하는 코드 작성

```
import sys
f = open("a.txt",'w')
sys.stdout = f for i in range(2,10):
    for j in range(1,10):
        print("{} X {} = {}".format(i,j,i*j))
    print()
f.close()
```

Run! 프로그래밍

Mission 2 바탕화면에 python 폴더 생성 후, 파일 생성

```
import os
user = os.getlogin() # 사용자 이름 (학습자님 사용자 명을 적어주세요)
os.mkdir('C:₩₩Users₩₩'+user+'₩₩Desktop₩₩python')
os.chdir('C:₩₩Users₩₩'+user+'₩₩Desktop₩₩python')os.getcwd()
f = open("a.txt",'w')
f.write("""안녕하세요. 학습자님!
이번 회차에서는 파이썬의 파일 입출력에 대해서 학습했습니다.수고하셨습니다.
""")
f.close()
```

학습정리

1. 파일 입출력

파일 입출력이란?	• open() 함수를 활용해 파일을 읽고, 쓰고, 수정할 수 있음
파일 쓰기	• w, x, a 모드로 파일을 엶 • write, writelines 함수를 활용해 파일에 내용을 쓸 수 있음
파일 읽기	• r 모드로 파일을 엶 • read, readline, readlines 함수를 활용해 파일의 내용을 읽어 올 수 있음

2. 파일 입출력의 활용

표준 출력 전환	• 파이썬의 sys 모듈을 활용 • 표준 입출력을 파일로 변경하고 파일로 프로그램 실행 결과를 출력할 수 있음
다른 자료형의 파일 입출력	• 텍스트가 아닌 객체의 자료형 그대로 파일에 저장하고 싶을 땐 pickle 모듈을 활용해 객체 그대로를 파일에 저장할 수 있음

학습정리

3. 파일 및 디렉토리 다루기

파일 다루기	• 파이썬의 OS 모듈을 활용 • 파일의 위치, 목록, 확장자 등을 확인할 수 있고, 파일명을 변경할 수 있음
디렉토리 다루기	• 디렉토리 작업 환경을 변경할 수 있음 • 새로운 폴더 생성, 폴더 삭제 등을 할 수 있음