

# 파이썬 프로그래밍 강의 노트 #09

---

## 파일

# 파일(File)이란?

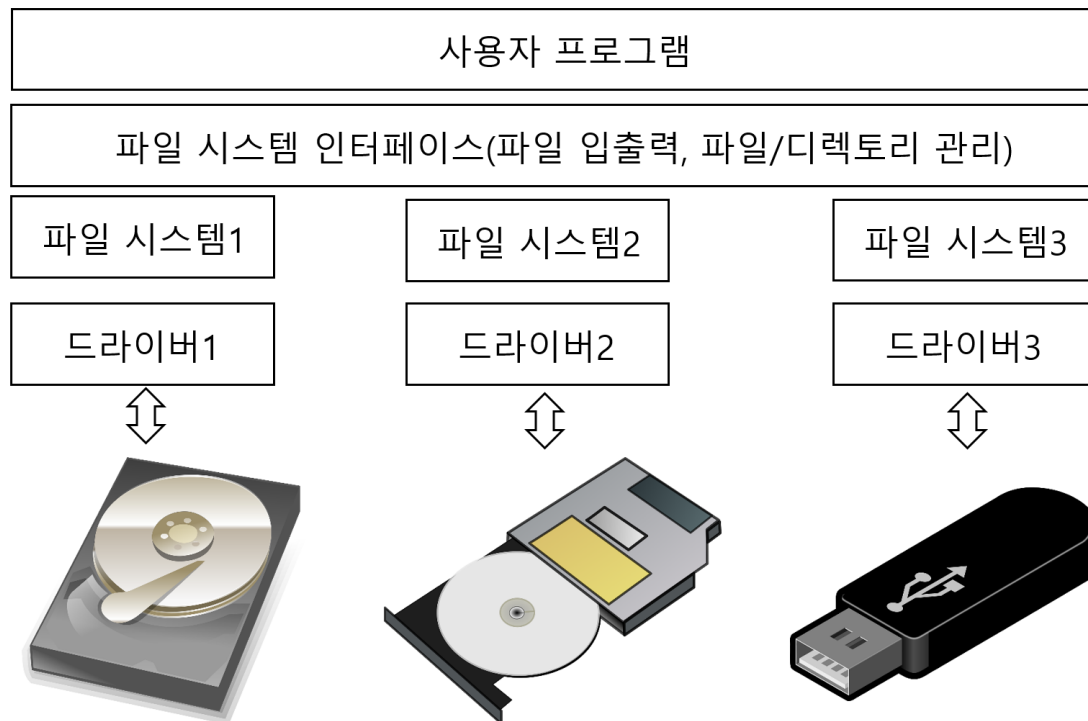
---

- 컴퓨터의 보조 저장 장치(ssd, hdd, odd, usb 등)에 저장되는 데이터를 담는 논리적인 저장 용기 (container)
  - 파일 이름, 크기, 권한, 소유자, 수정 날짜, 생성 날짜 등의 속성이 있음
  - 크게 텍스트 파일/바이너리 파일로 구분
  
- 파일에 담을 수 있는 데이터
  - 종류의 제한이 없음
  - 디지털로 표현 가능한 데이터
  - 사진, 음악, 문서 등

# 파일(File)이란?

## □ 파일 시스템

- 윈도우 같은 운영체제에서 파일과 디렉토리를 생성, 삭제 및 관리하는 방법, 파일에 데이터를 읽고 쓰는 방법 등을 구현한 것



# 파일 (File)

## □ 파일 속성

- 대부분의 운영체제에서 파일들은 다음 표에 보인 속성들을 포함

| 속성    | 설명                                   |
|-------|--------------------------------------|
| 이름    | 파일의 이름                               |
| 크기    | 파일에 포함된 데이터의 크기                      |
| 권한    | 누가 파일을 읽고, 쓰고, 실행할 수 있는 지 등을 나타내는 정보 |
| 소유자   | 파일을 생성한 사용자 정보                       |
| 수정 날짜 | 파일의 내용이 수정된 가장 마지막 날짜와 시각 정보         |
| 생성 날짜 | 파일의 내용이 처음 생성된 날짜와 시각 정보             |

# 파일 (File)

---

## □ 텍스트/바이너리 파일의 분류

### ■ 텍스트 파일

- 일반적으로 사람들이 사용하는 글자들로 구성된 데이터를 담고 있음 (예: 소스코드)
- 메모장 또는 코딩용 에디터(텍스트 편집기)를 이용해서 수정 가능

### ■ 바이너리 파일

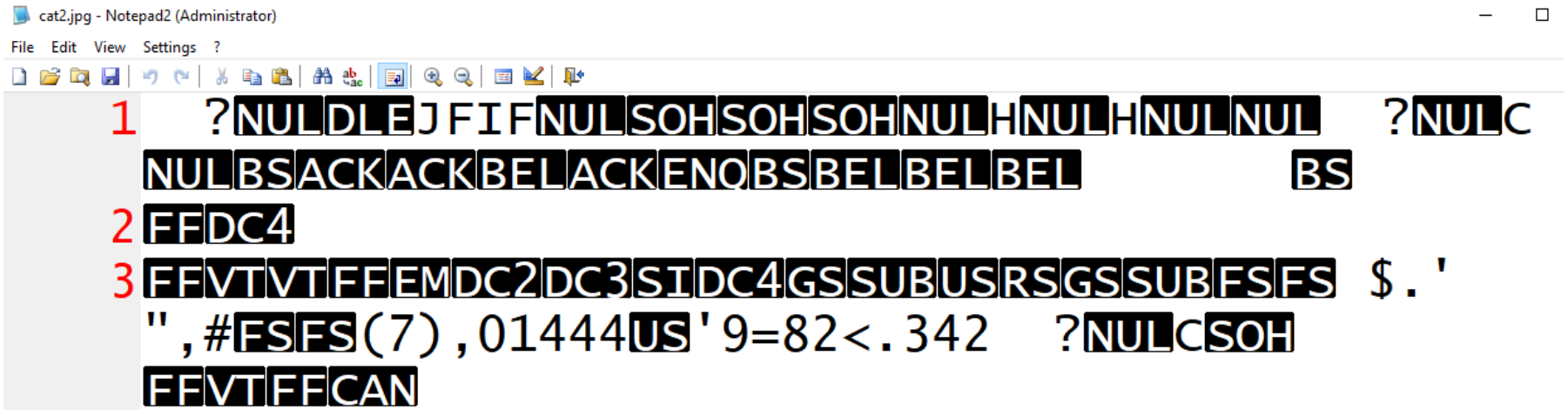
- 이진화된 데이터를 담고 있음 (예: 이미지/음원 파일 등)
- 수정하려면 전용 프로그램 또는 특화된 프로그램이 필요할 수 있음

## □ 구별 방법

- 확장자를 보는 것이 제일 쉬움

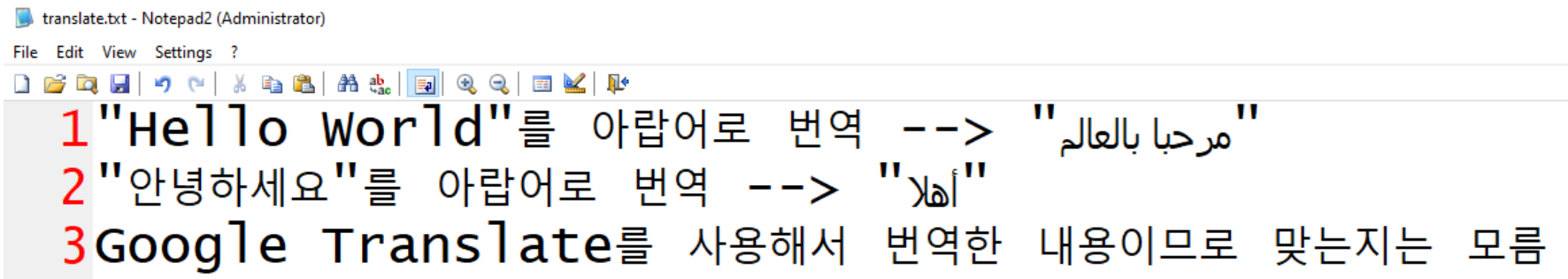
# 파일 (File)

## □ 바이너리 파일을 메모장으로 열어본 것



```
cat2.jpg - Notepad2 (Administrator)
File Edit View Settings ?
1 ?NULDLEJFIFNULSOH SOH SOH NULHNULHNULNUL ?NULC
NULBSACKACKBELACKENQBSBELBELBEL BS
2 FFDC4
3 FFVTVTFFEMDC2DC3SIDC4GSSUBUSRSGSSUBFSFS $. '
",#FSFS(7),01444US'9=82<.342 ?NULCSOH
FFVTFFCAN
```

## □ 텍스트 파일을 메모장으로 열어본 것

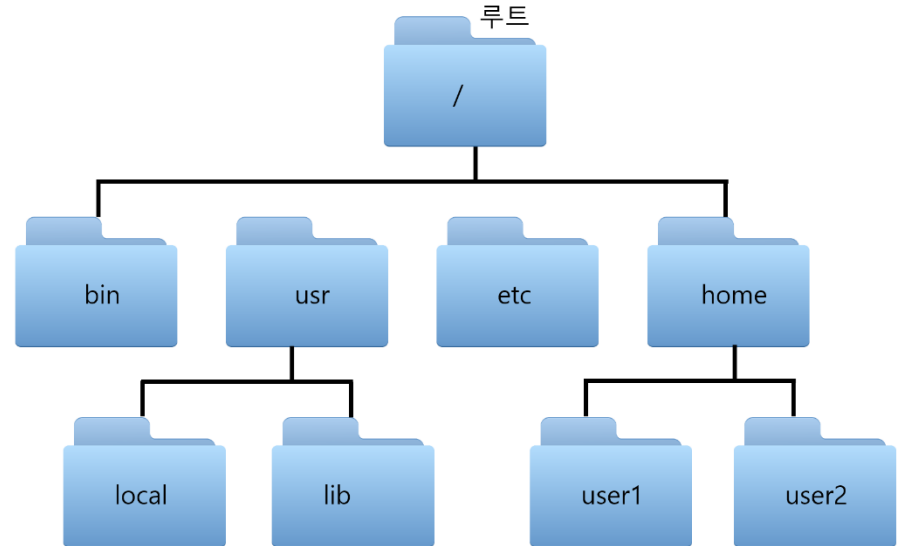
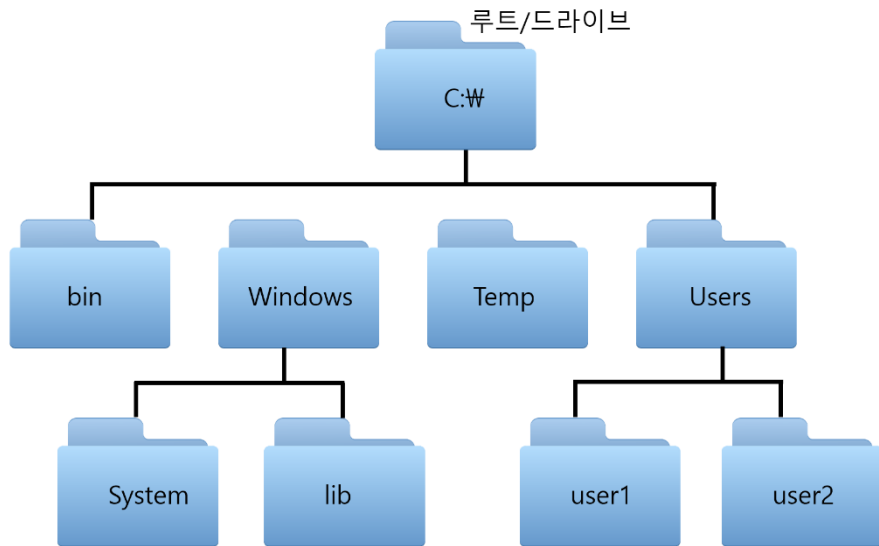


```
translate.txt - Notepad2 (Administrator)
File Edit View Settings ?
1 "Hello world"를 아랍어로 번역 --> "مرحبا بالعالم"
2 "안녕하세요"를 아랍어로 번역 --> "أهلا"
3 Google Translate를 사용해서 번역한 내용이므로 맞는지는 모름
```

# 파일 이름

- 많은 운영체제에서는 파일 이름에 해당 파일에 있는 데이터의 종류를 나타냄
  - 이러한 경우 주로 '.'과 확장자(extension)을 사용하는 경우가 많음
  - 예: 텍스트 파일(.txt), jpeg 이미지 파일(.jpg), 워드 파일(.doc 또는 .docx) 등
- 디렉토리 또는 폴더
  - 운영체제에서 많은 수의 파일을 관리하기 위한 목적으로 폴더(디렉토리)를 이용해서 파일을 넣는 공간을 분리함
  - 디렉토리에는 파일들 또는 또 다른 서브 폴더(서브 디렉토리)들이 존재할 수 있음
  - 윈도우에서는 드라이브라는 또 다른 폴더 개념이 있고 여기에는 여러 개의 폴더나 파일이 존재할 수 있음

# 디렉토리





# 파일 이름

## □ 파일 경로 (file path)

### ■ 절대 경로(absolute path)

- 최상위 폴더부터 시작해서 파일이 있는 위치까지 모두 표현하는 방법
- 윈도우에서는 드라이브부터 시작 (대소문자 구별 안함)

```
c:\users\ycho\temp\a.txt  
d:/users/ycho/temp/a.txt
```

- 맥 또는 리눅스에서는 root 폴더(/)부터 시작
  - 대문자/소문자 구별

```
/users/ycho/temp/a.txt
```

# 파일 이름

## ■ 상대 경로

- 현재 작업 디렉토리(current working directory 또는 cwd)
  - 현재 운영체제에서 기억하고 있는 위치(폴더/디렉토리)
- 상대경로는 현재 작업 디렉토리를 기준으로 상대적인 위치를 나타냄
- 예: cwd가 c:/users/ycho 또는 /users/ycho인 경우, temp 서브 폴더의 a.txt는 temp/a.txt (또는 윈도우는 temp\wa.txt)로 표현 가능
- 상대 경로를 표시할 때에는 '.' 또는 '..'로 폴더를 지정

|              |                            |
|--------------|----------------------------|
| ./temp/a.txt | (c:/users/ycho/temp/a.txt) |
| ../a.txt     | (c:/users/a.txt)           |

# 파일 이름

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19043
(c) Microsoft Corporation. All rights
C:\Users\ycho>
```

```
ycho@ycho-ubuntu-vm:~$ pwd
/home/ycho
ycho@ycho-ubuntu-vm:~$
```

- IDLE에서 파이썬 코드를 이용해서 현재 작업 디렉토리를 찾는 방법

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

```
IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import os
>>> print(os.getcwd())
C:\DevTools\Python
>>>
```

# 파일 입출력

- 파일 입출력 순서
  - 파일 열기
  - 파일 입출력 (read, write)
  - 파일 닫기 (생략가능)
- 물이나 편지가 있는 병과 비슷함



# 파일 입출력

## □ 파일 열기

- 어떤 파일을 열까?
    - 파일 경로 제공
  - 어떤 목적으로 열까?
    - 읽기, 쓰기 등 파일 열기 모드(mode) 제공
  - 열린 파일은 어떻게 접근할까?
    - 파일 객체 반환 (열린 파일에 대해 이름표를 붙일 수 있음)
  - 파이썬에서 제공하는 open() 함수 사용
- 파일\_변수 = open(파일\_이름)
- 예

```
f = open("C:\temp\data.txt")  
f = open("../..\\temp\\data.txt")
```

# 파일 입출력

## □ 파일 닫기

- 파일은 스스로 닫는 방법을 알고 있음

- 파일 객체의 `close()` 함수 사용

- 열린 파일의 이름표를 사용

```
파일_변수.close()          # 파일을 닫음
```

- 예

```
f.close()
```

# 텍스트 파일 읽기

---

- 함수들을 이용해서 텍스트 파일의 내용을 읽는 방법
  - 파일 내용을 한꺼번에 읽기
  - 한 줄씩 읽어서 문자열에 저장
  - 파일의 반복자를 이용해서 한 줄씩 처리
  - 파일의 내용을 한 줄씩 읽어 리스트로 구성

# 파이썬에서 파일 입출력

## ▣ 파일 내용 한꺼번에 읽기

```
f = open("t.txt") # 열기, f = open("t.txt", "r")
s = f.read()      # 파일 내용 전체 읽기
print(s)          # 읽은 내용 출력
f.close()          # 파일 닫기

f = open("t.txt", "w")
f.write("hello world")
f.close()
```



# 파이썬에서 텍스트 파일 읽기

## □ 줄 단위로 파일 읽기

### ■ readline() 함수 쓰기

- 한 줄(line)의 텍스트를 읽어서 반환
- 파일에 끝에 도달하면 False (빈 문자열) 가 됨
- 주로 while 문과 함께 쓰임

```
f = open("t.txt")      # f = open("t.txt", "r")
line = f.readline()    # 한 줄 읽어옴
while line:            # 파일의 끝에 도달하면 False
    print(line, end = ' ')
    line = f.readline()
f.close()               # 파일 닫기
```

# 파이썬에서 텍스트 파일 읽기

- 파일 객체의 반복자 활용하기
- 텍스트 파일이 순서가 있는 객체 역할을 함

```
f = open("t.txt") # f = open("t.txt", "r")
for line in f:
    # line에는 이미 줄바꿈 문자가 들어있음
    print(line, end = '')
f.close()          # 파일 닫기
```

# 파이썬에서 텍스트 파일 읽기

- 파일 내용을 리스트로 구성
- readlines() 함수 활용

```
f = open("t.txt") # f = open("t.txt", 'r')  
# 파일 전체의 내용을 리스트로 만듦  
lines = f.readlines()  
for line in lines:  
    print(line, end = ' ')  
f.close()          # 파일 닫기
```

# 텍스트 파일 쓰기

- 파일에 내용을 쓰는 것은 두 가지 방법이 있음
    - 파일의 내용을 지우고 새로 쓰기
    - 파일의 기존 내용에 덧붙이기
  - 파일 내용을 지우고 새로 쓰기
    - 쓰기 모드로 파일 열기
      - open() 함수를 호출할 때 두 번째 인자로 'w'를 전달
- ```
파일_변수 = open("파일 이름", 'w')
```
- 예
- ```
>>> f = open("C:\\temp\\data2.txt", "w")
```

# 텍스트 파일 쓰기

## ■ 파일에 내용 쓰기

- 파일에 포함되어 있는 write() 함수에 문자열을 전달

```
>>> f.write("Fusce est.")
```

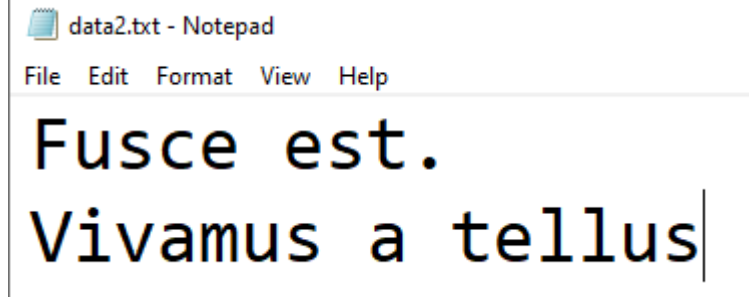
- write() 함수는 파일에 저장한 글자의 개수를 반환

```
>>> f.write("\nVivamus a tellus")
17
```

- 내용 저장 후에 파일 닫기

```
>>> f.close()
```

## 저장된 파일



# 텍스트 파일 쓰기

## □ 기존 파일 내용에 덧붙이기

### ■ 추가 모드로 파일 열기

- open() 함수를 호출할 때 두 번째 인자로 'w'를 전달

```
파일_변수 = open("파일 이름", 'a')
```

### □ 예제 코드

```
>>> f = open("C:\\temp\\data2.txt", "a")
>>> f.write("Nunc viverra imperdiet enim.")
28
>>> f.write("Lorem ipsum dolor sit amet,
consectetuer adipiscing elit.")
57
>>> f.close()
```

## with 문

- 파이썬의 with문은 두 개의 관련된 연산들 사이에서 작업을 수행할 때 유용함
- 두 개의 관련된 연산이란 open()-close(), save()-restore() 등과 같은 작업을 의미
- 다음은 파일을 처리할 때 with문을 활용하는 대표적인 예를 보여줌

```
f = open("t.txt") # 파일 열기
# 파일 읽기
for line in f:
    print(line, end = '')
f.close()          # 파일 닫기
```

## with 문

---

- 아래 코드는 앞에서 본 코드와 같은 작업을 함
  - 단 close()함수를 따로 부르지 않아도 됨

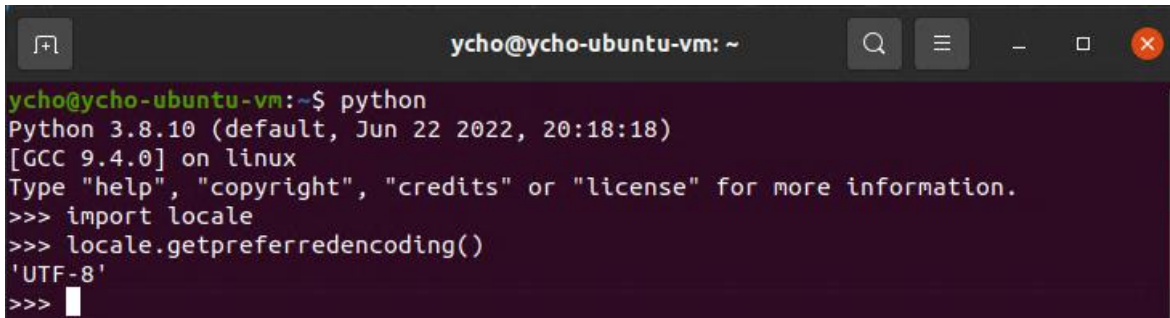
```
with open("t.txt") as f:  
    for line in f:  
        print(line, end = '')
```



# 파일 입출력과 문자 인코딩

- 파일에 저장할 때에도 인코딩 방법을 알아야 함
- 파이썬은 운영체제에서 주로 사용하는 방법으로 인코딩함
  - 기본 인코딩 방식이 아닌 다른 형태로 파일을 읽고 쓰려면 파일을 열 때 인코딩 방식을 지정해야 함
- 시스템에서 사용하는 기본 인코딩 방법 확인

```
>>> import locale  
>>> locale.getpreferredencoding()  
'cp949'
```

A terminal window titled 'ycho@ycho-ubuntu-vm: ~' with search, menu, and window control icons. It shows the execution of a Python script. The output of 'python' is 'Python 3.8.10 (default, Jun 22 2022, 20:18:18) [GCC 9.4.0] on linux'. The interactive prompt shows 'import locale' and 'locale.getpreferredencoding()' returning 'UTF-8'.

```
ycho@ycho-ubuntu-vm: ~$ python  
Python 3.8.10 (default, Jun 22 2022, 20:18:18)  
[GCC 9.4.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import locale  
>>> locale.getpreferredencoding()  
'UTF-8'  
>>>
```

# 파일 입출력과 문자 인코딩

- 일반적으로 플랫폼에서 사용하는 인코딩 방식을 이용해서 파일 입출력하기

```
f = open(filename)
```

- 오류 발생 가능

- UnicodeDecodeError

- 텍스트 파일의 내용이  
locale.getpreferredencoding() 형태로 되어 있지 않으면 발생됨

- UnicodeEncodeError

- 정해진 인코딩 방법으로 저장하려 할 때 저장할 수 없는 문자들이 있는 경우 발생됨

# 파일 입출력과 문자 인코딩

- utf-8로 저장한 파일을 cp949로 읽을 때 오류 발생

```
>>> with open("C:\\temp\\python.utf8.txt") as f:  
...     s = f.read()  
...     print(s)  
...  
Traceback (most recent call last):  
  File "<stdin>", line 2, in <module>  
UnicodeDecodeError: 'cp949' codec can't decode  
byte 0xed in position 0: illegal multibyte  
sequence
```

# 파일 입출력과 문자 인코딩

- 특정 인코딩 형태로 저장되어 있는 파일 읽기
  - `open()` 함수를 호출하며 "encoding" 인자의 값을 원하는 방식으로 지정
- UTF-8 형식으로 저장된 파일 읽기

```
f = open(filename, encoding="utf-8")
```

- 예제(파일 읽기)

```
>>> with open("python.utf8.txt", encoding="utf-8") as f:  
...     s = f.read()  
...     print(s)  
...  
파이썬
```

# 파일 입출력과 문자 인코딩

## ▣ 예제(파일 쓰기)

```
>>> with open("python.txt", "w", encoding="utf-8") as f:  
...     f.write("파이썬을 이용한 파일 입출력")  
...  
15
```

# 실습문제 1

---

## □ 문제

- 사용자로부터 입력받은 문자열을 텍스트 파일에 저장하는 프로그램을 작성하세요.

## □ 요구사항

- 사용자로부터 문자열 입력 받기
- 파일은 기본 인코딩 방식을 사용해서 저장되었다고 가정
- "output.txt" 파일에 쓰는 프로그램
- 파일을 열고 내용을 작성한 후 자동으로 파일을 닫기

# 실습문제 1

## ▣ 최종 코드

```
def write_to_file():  
    # 사용자 입력 받기  
    text = input("저장할 문자열을 입력하세요: ")  
    # 파일에 쓰기  
    with open("output.txt", "w") as file:  
        file.write(text)  
    print("파일 저장 완료")  
  
write_to_file()
```

# 실습문제 2

---

## □ 문제

- 생성된 "output.txt" 파일의 내용을 읽고 화면에 출력하는 프로그램을 작성하세요.

## □ 요구사항

- `file.read()`



# 실습문제 2

## □ 최종 코드

```
def read_from_file():  
    # 파일 읽기  
    with open("output.txt", "r") as file:  
        content = file.read()  
    print("파일 내용:")  
    print(content)  
  
# 함수 호출  
read_from_file()
```

## 실습문제 3

---

### □ 문제

- 사용자로부터 여러 줄의 텍스트를 입력받아 기존 파일에 추가하는 프로그램을 작성하세요. 사용자가 "stop"이라고 입력하면 입력을 중단합니다.

### □ 요구사항

- # 사용자로부터 "stop"을 입력받을 때까지 반복하여 내용을 입력받고,
- # 각 입력받은 내용을 "output.txt" 파일에 추가합니다.

# 실습문제 3

## □ 최종 코드

```
def append_to_file():  
    with open("output.txt", "a") as file:  
        while True:  
            line = input("추가할 내용을 입력하세요  
(종료하려면 'stop' 입력): ")  
            if line == "stop":  
                break  
            file.write(line + "\n")  
        print("내용 추가 완료")  
  
# 함수 호출  
append_to_file()
```

## 실습문제 4

---

### □ 문제

- "output.txt" 파일 내에서 사용자가 지정한 단어가 몇 번 나타나는지 카운트하는 프로그램을 작성하세요.

### □ 요구사항

# 실습문제 4

## □ 최종 코드

```
def count_word_in_file():  
    word = input("카운트할 단어를 입력하세요: ")  
    count = 0  
    with open("output.txt", "r") as file:  
        content = file.read()  
        count = content.count(word)  
    print(f"'{word}'의 출현 횟수: {count}")  
  
# 함수 호출  
count_word_in_file()
```

# 실습문제 5

---

## □ 문제

- "output.txt" 파일의 각 행을 읽고 번호를 매겨서 출력하는 프로그램을 작성하세요.

## □ 요구사항

# 실습문제 5

## ▣ 최종 코드

```
def print_lines_with_numbers():  
    with open("output.txt", "r") as file:  
        lines = file.readlines()  
        for index, line in enumerate(lines):  
            print(f"{index + 1}: {line.strip()}")  
  
# 함수 호출  
print_lines_with_numbers()
```

# 실습문제 6

## □ 문제

- 사용자로부터 이름, 나이, 이메일을 입력받아 "contacts.csv" 파일에 저장하는 프로그램을 작성하세요.

## □ 요구사항

- `Imprt csv` 활용
- `csv.writer()` 활용
- `writerow()` 활용

```
이름을 입력하세요 (종료하려면 'stop' 입력): b  
나이를 입력하세요: 10  
이메일을 입력하세요: b@a.com  
이름을 입력하세요 (종료하려면 'stop' 입력): c  
나이를 입력하세요: 10  
이메일을 입력하세요: 10@a.com  
이름을 입력하세요 (종료하려면 'stop' 입력): stop  
연락처가 csv 파일에 저장되었습니다.
```



# 실습문제 6

## ▣ 최종 코드

```
def write_csv():
    import csv
    with open('contacts.csv', 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["이름", "나이", "이메일"])

        while True:
            name = input("이름을 입력하세요 (종료하려면 'stop' 입력): ")
            if name.lower() == "stop":
                break
            age = input("나이를 입력하세요: ")
            email = input("이메일을 입력하세요: ")
            writer.writerow([name, age, email])
        print("연락처가 CSV 파일에 저장되었습니다.")

write_csv()
```

# 실습문제 7

---

## □ 문제

- "contacts.csv" 파일에 저장된 모든 연락처 정보를 읽어서 출력하는 프로그램을 작성하세요.

## □ 요구사항

- csv.reader(file) 활용
- "이름: , 나이: , 이메일: 형식 출력
- next(reader) # 첫 번째 행 건너뛰기

# 실습문제 7

## ▣ 최종 코드

```
def read_csv():
    import csv
    with open('contacts.csv', 'r') as file:
        reader = csv.reader(file)
        next(reader) # 첫 번째 행 건너뛰기
        for row in reader:
            print(f"이름: {row[0]}, 나이: {row[1]},
이메일: {row[2]}")

# 함수 호출
read_csv()
```

# 실습문제 8

---

## □ 문제

- 사용자로부터 검색어를 입력받아 " output.txt" 파일 내에서 해당 검색어가 포함된 모든 라인을 출력하는 프로그램을 작성하세요.

## □ 요구사항

# 실습문제 8

## ▣ 최종 코드

```
def search_in_file():  
    keyword = input("검색할 키워드를 입력하세요: ")  
    with open("output.txt", "r") as file:  
        lines = file.readlines()  
        for line in lines:  
            if keyword in line:  
                print(line.strip())  
  
# 함수 호출  
search_in_file()
```

# 실습문제 9

---

## □ 문제

- 임의의 바이너리 데이터를 파일에 쓰는 프로그램을 작성하세요.

## □ 요구사항

# 실습문제 9

## ▣ 최종 코드

```
def write_binary_data():  
    # 바이너리 데이터  
    data = bytes([255, 0, 127, 64, 32, 16])  
    # 파일에 바이너리 데이터 쓰기  
    with open('binary_data.bin', 'wb') as file:  
        file.write(data)  
    print("바이너리 데이터가 저장되었습니다.")  
  
# 함수 호출  
write_binary_data()
```

# 실습문제 10

---

## □ 문제

- 바이너리 파일에서 데이터를 읽어와서 각 바이트를 화면에 출력하는 프로그램을 작성하세요.

## □ 요구사항



# 실습문제 10

## ▣ 최종 코드

```
def read_binary_data():  
    with open('binary_data.bin', 'rb') as file:  
        data = file.read()  
        print("읽은 바이너리 데이터:")  
        for byte in data:  
            print(byte)  
  
# 함수 호출  
read_binary_data()
```

# 실습문제 11

---

## □ 문제

- 기존 바이너리 파일에서 특정 위치의 바이트를 수정하여 저장하는 프로그램을 작성하세요.

## □ 요구사항

# 실습문제 11

## ▣ 최종 코드

```
def modify_binary_data():  
    # 파일 읽기  
    with open('binary_data.bin', 'rb') as file:  
        data = bytearray(file.read())  
  
    # 데이터 수정 (예: 3번째 바이트를 255로 변경)  
    data[2] = 255  
  
    # 수정된 데이터 쓰기  
    with open('binary_data.bin', 'wb') as file:  
        file.write(data)  
  
    print("바이너리 데이터가 수정되었습니다.")  
  
# 함수 호출  
modify_binary_data()
```

# 실습문제 12

---

## □ 문제

- 이미지 파일을 읽어 다른 이름으로 복사 저장하는 프로그램을 작성하세요.

## □ 요구사항

- 이미지 파일을 'rb' 모드로 열어 데이터를 읽고, 'wb' 모드로 새 파일을 열어 읽은 데이터를 저장합니다
- Image.jpg 파일 준비

# 실습문제 12

## □ 최종 코드

```
def copy_image_file():  
    # 이미지 파일 읽기  
    with open('image.jpg', 'rb') as file:  
        data = file.read()  
  
    # 이미지 파일 쓰기  
    with open('copy_image.jpg', 'wb') as file:  
        file.write(data)  
  
    print("이미지 파일이 복사되었습니다.")  
  
# 함수 호출  
copy_image_file()
```

