codingOn X posco

K-Digital Training 스마트팩토리 개발자 입문

codingOn

Python 파일입출력

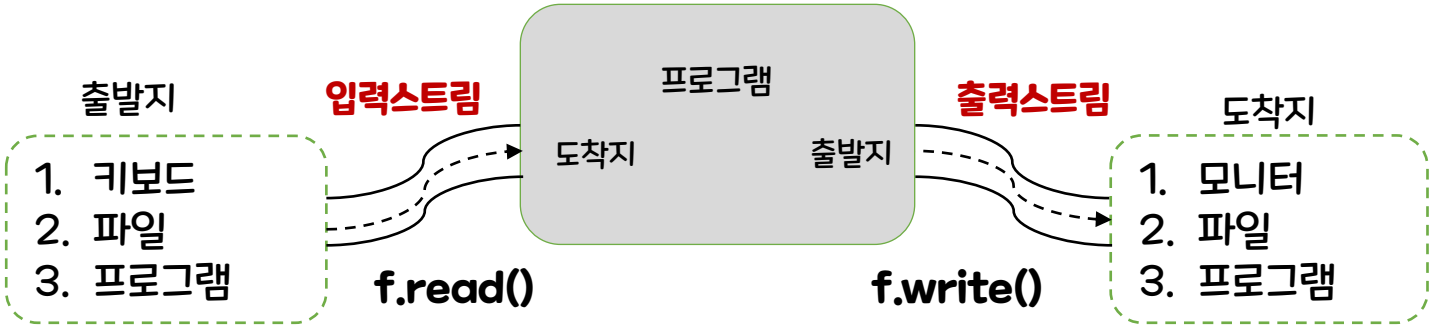
입,출력 스트림

스트림(stream)

자료흐름이 물의 흐름과 같다는 뜻이다. 입출력 장치는 매우 다양하기 때문에 프로그램 호환성이 떨어짐

입력 스트림 - 동영상을 재생하기 위해 동영상 파일에서 자료를 읽을때 사용함

출력 스트림 - 사용자가 쓴 글을 파일에 저장할 때는 출력 스트림 사용함

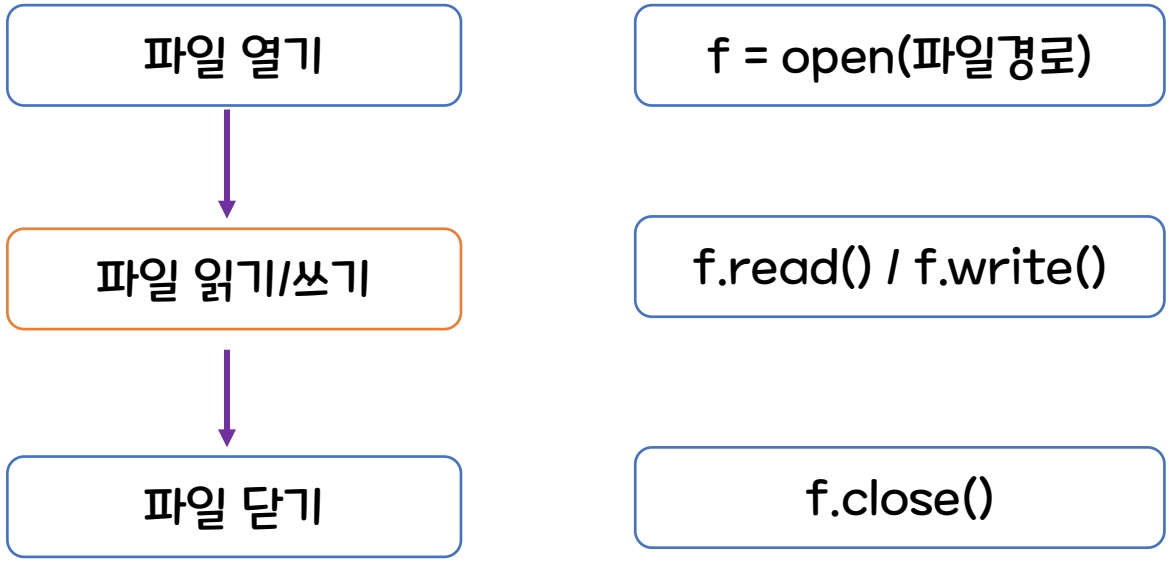


입,출력 스트림

- 파일 입출력의 필요성

  프로그램 실행 중에 메모리에 저장된 데이터는 프로그램이 종료되면 사라진다.데이터를 프로그램이 종료된 후에도 계속해서 사용하려면 파일에 저장하고 필요할 때 파일을 읽어서 데이터를 사용할 수 있다.

- 파일 입출력 프로세스



with문

- 파일, 네트워크 연결, 데이터베이스 연결 등 리소스를 사용한 작업이 끝난 후 이를 자동으로 정리(cleanup)합니다.
- 파일을 열고 작업한 후 파일을 자동으로 닫거나, 잠금을 해제하는 등의 처리를 보장
- 작업 중 예외가 발생하더라도 자동으로 리소스를 정리하므로, 예외로 인해 리소스가 닫히지 않는 문제를 방지

수동닫기와 자동닫기

codingOn

- 파일을 open()을 이용하여 열고 작업이 끝나면 close()를 이용하여 파일을 닫아줘야함
- with문을 사용하면 파일을 알아서 정리해주기 때문에 작업효율이 향상

```
# 수동닫기
file = open("example.txt", "w")  # 파일 열기
file.write("Hello, Python!")     # 파일에 쓰기
file.close()                     # 파일 닫기

# 자동닫기
with open("example.txt", "w") as file:
    file.write("Hello, Python!")  # 파일에 쓰기
# 파일은 자동으로 닫힘
```

파일 입출력

파일 객체 = open(파일이름, 파일 모드)

[파일 모드]

r : 읽기 모드(파일이 존재해야 함)

w : 쓰기 모드(파일이 없으면 새로 생성, 기존 파일 내용 삭제)

a : 추가 모드(파일 끝에 내용 추가)

b : 바이너리 데이터

파일쓰기

codingOn

- write() 메서드사용, 문자열만 가능, 숫자는 문자열형태로 변환 후 사용

```
# 파일 생성
with open("test.txt", "w") as file:
    file.write("안녕하세요!\n")
    file.write("파이썬 파일 쓰기 예제입니다.\n")
print("파일에 데이터가 작성되었습니다.")

# 내용 추가
with open("test.txt", "a") as file:
    file.write("새로운 내용을 추가합니다.\n")
    file.write("여기에 또 다른 줄을 추가합니다.\n")
print("파일에 데이터가 추가되었습니다.")
```

- 만약 한글이 깨진다면 3번째에 encoding="utf-8" 입력

파일쓰기

codingOn

- writelines() 메서드사용. 문자열리스트 형태

```
lines = ["첫 번째 줄\n", "두 번째 줄\n", "세 번째 줄\n"]

# 리스트 내용을 파일에 쓰기
with open("testlist.txt", "w") as file:
    file.writelines(lines)
print("리스트 데이터가 파일에 작성되었습니다.")
```

파일쓰기

codingOn

- 사용자로부터 내용입력 받기

```
with open("user_input.txt", "w") as file:
    while True:
        line = input("파일에 저장할 내용을 입력하세요 (종료하려면 '종료' 입력): ")
        if line == "종료":
            print("입력을 종료합니다.")
            break
        file.write(line + "\n")
print("사용자 입력이 파일에 저장되었습니다.")
```

파일읽기

codingOn

파일 읽기 모드 메서드

- read() : 파일의 내용 전체를 문자열로 반환, 메모리 사용량이 높음
- readline(): 파일에서 한 줄을 읽어옴. 메모리 사용량이 가장 적음
- readlines() : 파일의 모든 줄을 읽고, 각각의 줄을 요소로 갖는 리스트를 반환, 메모리 사용량이 보통
  - 예)
  - 큰 파일: readline()을 사용하여 한 줄씩 처리
  - 작은 파일: read() 또는 readlines()를 사용하여 빠르게 처리

파일읽기

codingOn

- 파일쓰기할때 인코딩을 했다면 3번째에 encoding="utf-8" 입력

```
# 파일 내용 전체가져오기
with open("test.txt", "r") as file:
    content = file.read()
    print("파일 내용:")
    print(content)

# 파일을 한줄씩 읽기
with open("test.txt", "r") as file:
    print(file.readline())  # 첫 번째 줄
    print(file.readline())  # 두 번째 줄

# 모든 줄을 읽고 각 줄을 리스트에 담기
with open("test.txt", "r" ) as file:
    lines = file.readlines()
    print(lines)
```

파일읽기

- 내장함수: enumerate(리스트) - 리스트의 요소들을 튜플형태로 (인덱스, 요소값) 으로 반환

```
with open("test.txt", "r") as file:
    line = file.readline()
    while line:
        print(line.strip())  # 양쪽공백 제거
        line = file.readline()

with open("test.txt", "r") as file:
    lines = file.readlines()
    for idx, line in enumerate(lines):
        print(f"{idx + 1}번째 줄: {line.strip()}")
```

바이너리 파일

바이너리 파일이란 영상, 이미지, 음성 등이 대부분인 파일로 0과 1로 이루어진 파일이다.

[파일 모드]

rb : 바이너리 읽기 모드

wb : 바이너리 쓰기 모드(파일을 새로 생성)

ab : 바이너리 추가 쓰기 모드(파일 끝에 추가)

바이너리 읽기

```python
with open("owl.png", "rb") as file:
    data = file.read()
    print(f"{len(data)} 바이트")

with open("owl.png", "rb") as file:
    header = file.read(10)
    print(f"{header}")
# PNG 파일은 항상 b'\x89PNG\r\n\x1a\n'로 시작
# JPEG 파일은 항상 b'\xff\xd8\xff'로 시작

def identify_file(file_path):
    with open(file_path, "rb") as file:
        header = file.read(4)
        print(header)
        if header[:2] == b'\xff\xd8':  # JPEG
            return "JPEG"
        elif header == b'\x89PNG':  # PNG
            return "PNG"
        else:
            return "Unknown format"

print(identify_file("owl.png"))
```

바이너리 쓰기

```
with open("owl.png", "rb") as source_file:
    data = source_file.read()

with open("owl_copy.png", "wb") as dest_file:
    dest_file.write(data)

print("이미지가 성공적으로 복사되었습니다!")
```

복.습.철.저

수고하셨습니다