05장 사용자 입력과 출력

5.1 표준 입출력

- 사용자 입력은 다음과 같이 input() 함수를 이용해 받을 수 있다.
- 화면으로 출력할 때는 print() 함수를 사용한다.

```
# 사용자 입력
a = input("질문 내용: ")
print(a)

# print 자세히 알기
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]

print("life" "is" "too short")  # 큰따옴표(")로 둘러싸인 문자열은 + 연산과 동일하다.
#lifeistoo short
print("life"+"is"+"too short")
#lifeistoo short

print("life", "is", "too short")  # 문자열 띄어쓰기는 꼼마로 한다.
# life is too short

for i in range(10):
    print(i, end=' ')  # 한 줄에 결과값 출력하기
# 0 1 2 3 4 5 6 7 8 9
```

5.2 파일 입출력

- open() 함수를 통해 파일을 연 후 작업하는 것이 일반적이다.
- open() 함수의 기본형: 파일객체 = open(file, mode)
- pickle 모듈: 리스트나 클래스 등을 저장할 때에 유용하게 사용할 수 있다.

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

# 파일을 쓰기 모드로 열어 출력값 적기
f = open("새파일.txt", 'w', encoding='utf-8')
for i in range(1,11):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()

# 프로그램의 외부에 저장된 파일을 읽는 여러 가지 방법
# 1. readline() 함수 이용하기
```

```
f = open("새파일.txt", 'r', encoding='utf-8')
while True:
    line = f.readline()
    if not line: break
    print(line, end="")
f.close()

# 2. readlines() 함수 이용하기
f = open("새파일.txt", 'r', encoding='utf-8')
lines = f.readlines()
for line in lines:
    print(line, end="")
f.close()

# 3. read() 함수 이용하기
f = open("새파일.txt", 'r', encoding='utf-8')
data = f.read()
print(data)
f.close()

# 파일에 새로운 내용 추가하기
f = open("새파일.txt", 'a')
for i in range(11,20):  # 11부터 19까지 i에 대입
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()

# with문과 함께 사용하기 (파일을 열고 닫는 것을 자동으로 처리한다.)
f = open("새파일.txt", 'w')
f.write("Life is too short, you need python.")
f.close()

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


# pickle 모듈을 이용하여 리스트와 클래스 저장하기
colors = ['red','green','black']
print(colors)
import pickle
f = open('colors','wb')
pickle.dump(colors,f)
f.close()

class test:
    var = None
a = test()
a.var = 'Test'
f = open('test','wb')
pickle.dump(a,f)
f.close()
f = open('test','rb')
b = pickle.load(f)
f.close()
print(b)
print(b.var)
```

5.3 요약

- 표준입출력: 콘솔로 입력과 출력하는 함수, input() / print()
- 파일 입출력: open(), close(), readline(), readlines(), read()

[과제] 연습문제

Q1 다음은 두 개의 숫자를 입력받아 더하여 돌려주는 프로그램이다. 3과 6을 입력했을 때 9가 아닌 36이라는 결과값을 돌려주었다. 이 프로그램의 오류를 수정해 보자.

```
input1 = input("첫번째 숫자를 입력하세요: ")
input2 = input("두번째 숫자를 입력하세요: ")

total = input1 + input2
print("두 수의 합은 %s 입니다." %total)

# 출력결과
36
```

Q2 다음은 "test.txt"라는 파일에 "Life is too short\nyou need java" 문자열을 저장한 후 다시 그 파일을 읽어서 출력하는 프로그램이다. 이 프로그램은 우리가 예상한 "Life is too short"라는 문장을 출력하지 않는다. 우리가 예상한 값을 출력할 수 있도록 프로그램을 수정해 보자.

```
f1 = open("test.txt", 'w')
f1.write("Life is too short\nyou need java")

f2 = open("test.txt", 'r')
print(f2.read())
```

Q3 사용자의 입력을 파일(test.txt)에 저장하는 프로그램을 작성해 보자. (단 프로그램을 다시 실행하더라도 기존에 작성한 내용을 유지하고 새로 입력한 내용을 추가해야 한다.)

```
user_input = input("저장할 내용을 입력하세요: ")
f = open("test.txt", ____)
f.write(____)  ## 입력한 내용을 줄 단위로 구분하기 위해 줄 바꿈 문자 삽입
f.write(user_input)
f.close()
```

Q4 다음과 같은 내용을 지닌 파일 test.txt가 있다. 이 파일의 내용 중 'java'라는 문자열을 'python'으로 바꾸어서 저장해 보자.

```
'''
Life is too short
you need java
'''
f = open('test.txt', 'r')
body = _____________  # test.txt 파일의 내용을 body 변수에 저장
f.close()

body = ________________________  # body 문자열에서 'java'를 'python'으로 변경
```

```
f = open('test.txt', _____)  # 파일을 쓰기 모드로 다시 실행
f.write(body)
f.close()
```

Q5 아래와 같이 총 10줄로 이루어진 sample.txt 파일이 있다. sample.txt 파일의 숫자값을 모두 읽어 총합과 평균값을 구한 후 평균값을 result.txt라는 파일에 쓰는 프로그램을 작성해 보자.

```
# sample.txt
70
60
55
75
95
90
80
80
85
100
```

```
f = open("sample.txt")
lines =  _________  # sample.txt를 줄 단위로 모두 읽는다.
f.close()

total = 0
for line in lines:
    score = _________  # 줄에 적힌 점수를 숫자형으로 변환한다.
    total += __________

average = __________
f = open("result.txt", _________ )
f.____________
f.close()
```