

Chapter 13

객체와 텍스트 파일 다루기



목차

1. 이 장에서 만들 프로그램
2. 객체 지향 프로그래밍이란?
3. 클래스와 객체
4. 객체를 이용한 회원가입 프로그램
5. 텍스트 파일 다루기

실전 예제 1 주사위 게임

실전 예제 2 한 줄 일기장 프로그램

학습목표

- ‘객체 지향 프로그래밍’이 무엇인지 이해합니다.
- 클래스를 이해하고 선언하는 방법을 알아봅니다.
- 객체를 생성하는 방법을 알아봅니다.
- 객체의 속성을 참조하는 방법과 메서드를 호출하는 방법을 알아봅니다.
- 텍스트 파일에 접근하는 방법을 알아봅니다.
- 텍스트 파일에 텍스트를 쓰고 읽는 방법을 알아봅니다.

Section 01

이 장에서 만들
프로그램



1. 주사위 게임

■ 3명이 주사위를 5번씩 굴려서 나온 합을 구하는 프로그램으로 합이 가장 큰 사람이 이기는 프로그램

```
Gamer1 : [1, 3, 2, 3, 3]
Sum of Gamer1 : 12
-----

Gamer2 : [1, 4, 2, 4, 4]
Sum of Gamer2 : 15
-----

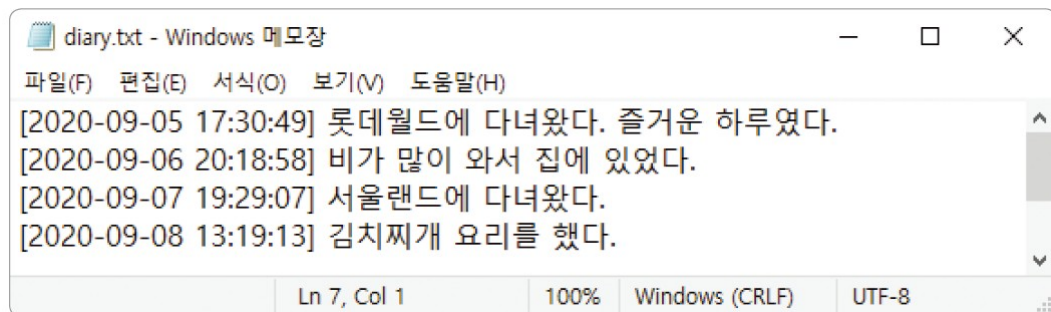
Gamer3 : [4, 3, 2, 1, 3]
Sum of Gamer3 : 13
-----

=====
1st : 15          WIN!!
2st : 13
3st : 12
=====
```



2. 한 줄 일기장 프로그램

- 하루 중 인상 깊었던 일을 파일에 한 줄로 작성하는 ‘한 줄 일기장’ 프로그램
- 결과 화면이 메모장에서 출력됨



Section 02

객체 지향 프로그래밍이란?



■ 객체 지향 프로그래밍(Object Oriented Programming, OOP)

- 객체를 이용하여 프로그램을 만드는 것

■ 객체란?

- 우리 주변의 마우스, 피아노, 자동차 등과 같은 ‘사물’을 의미함
- 객체는 속성(attribute)과 기능(function)으로 구성됨
- 속성 : 객체를 구성하는 요소
- 기능 : 객체가 하는 행위

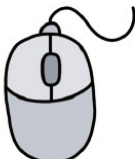


기능	커서 이동하다, 클릭하다	연주하다, 녹음하다	더하다, 빼다, 곱하다, 나누다
속성	색상, 모양 등	건반, 패달 등	숫자 버튼, 화면 등
객체	 마우스	 피아노	 계산기

그림 13-1 객체의 속성과 기능

객체 지향 프로그래밍이란?



■ 두 수를 입력 받아 사칙연산을 하는 계산기 프로그램에 객체 지향을 적용하기

▪ [1단계] 변수 만들기 → 속성

- 사용자가 입력한 숫자를 임시로 저장하는 변수를 선언함

▪ [2단계] 함수 만들기 → 기능

- 사용자가 원하는 연산(+, -, *, /)을 실행하기 위한 각각의 함수를 선언함

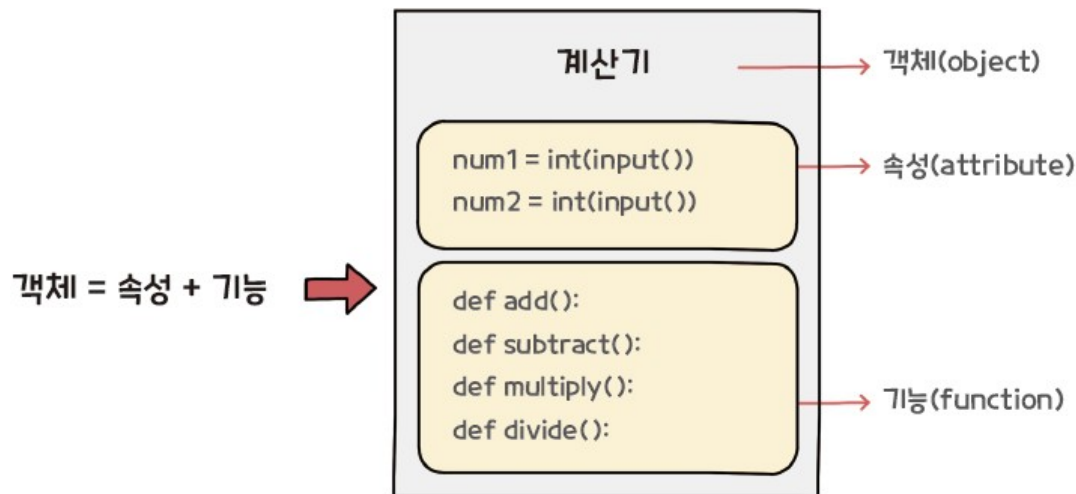


그림 13-2 프로그램 속 객체의 속성과 기능



■ 객체의 유기적인 동작

- 객체는 단독으로 사용되기도 하지만 서로 유기적으로 관계를 맺고 사용되기도 함
- 곱셈 게임 프로그램에 포함된 '계산기' 객체와 '프린트' 객체의 경우
 - '계산기' 객체에서 사용자의 점수를 계산하면
 - '프린트' 객체에서 그 결과를 받아 모니터에 출력하는 작업을 수행함

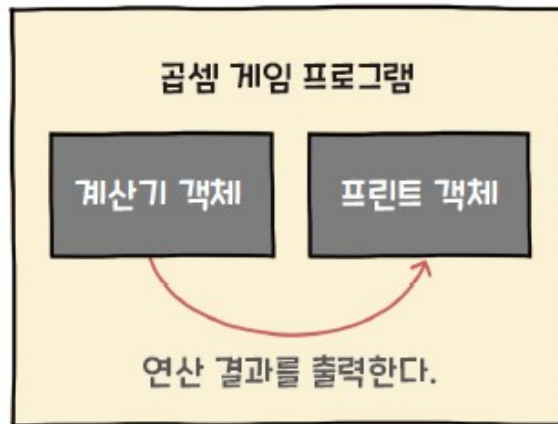


그림 13-3 서로 다른 객체의 유기적인 동작

11/49

Section 03

클래스와 객체



■ 클래스

- 객체를 생성하기 위해서는 클래스가 필요함
- 클래스란 객체를 생성하기 위한 틀(mold)로 마치 붕어빵 틀과 같음
 - 클래스(붕어빵 틀)에 속성과 기능을 설정하면 객체(붕어빵)가 생성됨
 - 즉 붕어빵이라는 '객체'를 만들기 위해서는 붕어빵 틀에 해당하는 '클래스'가 필요함
 - 붕어빵 틀이 있으면 원하는 만큼 붕어빵을 만들 수 있듯이 클래스가 있으면 원하는 만큼 객체를 생성할 수 있음

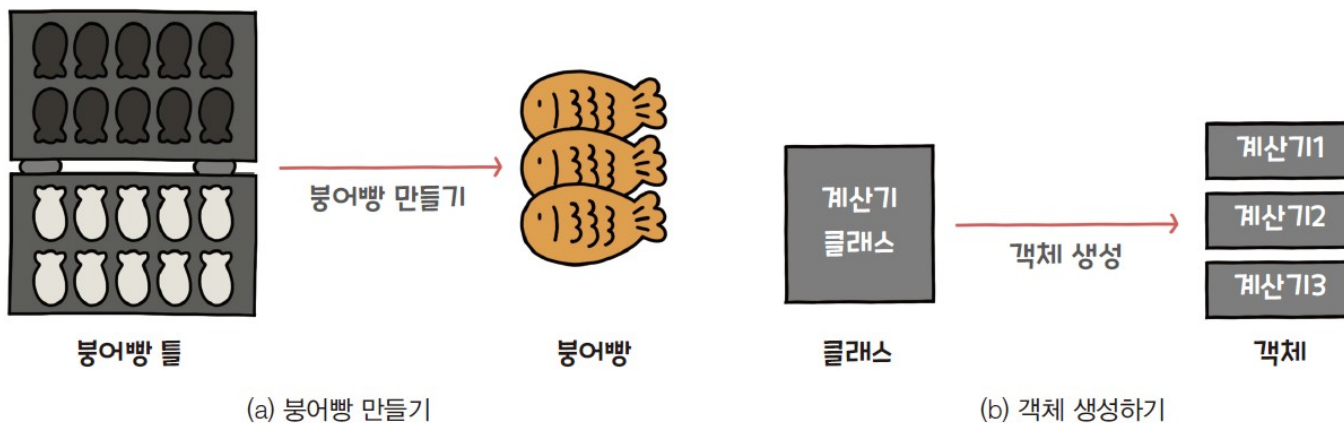


그림 13-4 클래스를 통한 객체 생성



■ 클래스의 구조

- class 키워드, 클래스 이름, 콜론(:)을 순서대로 적고 그 아래에 클래스 속성과 기능을 정의함

```
class 키워드   class 이름   콜론
  ↑           ↑           ↑
class FishBread: → 클래스 선언
    def __init__(self, f, b):
        self.flour = f
        self.bean = b → 클래스 속성 정의
    def makeFishBread(self):
        print('붕어빵 제조') → 클래스 기능 정의
```

그림 13-5 클래스의 기본 구조



■ 클래스 선언

- 클래스를 선언할 때는 class 키워드를 사용함
- class 키워드 다음에는 클래스의 이름과 콜론(:)이 옴
- 클래스 이름은 원하는 대로 지을 수 있지만 함수와 구분하기 위해서 첫 글자는 대문자로 함

■ 속성과 기능 정의

- 클래스의 속성 : 'def __init__():' 형태로 정의함
 - self는 클래스 자신을 가리킨다는 의미로 __init__()의 첫 번째 매개변수를 self로 지정
 - 기능 : 'def 함수명():' 형태로 정의함
- 클래스의 속성과 기능은 self를 이용함



■ 계산기 클래스를 만들며 클래스의 기본 구조를 익히기

- 클래스를 만들기 위해서는 먼저 클래스가 가지고 있는 속성과 기능을 정의해야 함
- 계산기 클래스는 숫자를 저장하는 속성과, 덧셈/뺄셈 기능을 하는 메서드로 구성됨

표 13-1 Calculator 클래스의 속성과 메서드

구분	이름	내용
클래스	Calculator	-
속성	num1	숫자1 저장
	num2	숫자2 저장
메서드	add()	덧셈 연산
	subtract()	뺄셈 연산



■ 계산기 클래스를 만들며 클래스의 기본 구조를 익히기

- 코드를 작성하고 F5를 눌러 실행함
- 클래스는 봉어빵 틀과 같이 객체를 만들기 위한 틀일 뿐 아직 객체를 만든 것은 아니기 때문에 실행 결과에 아무 것도 안나옴

코드 13-1

ch13_01.py

```
01 class Calculator:                                # 클래스 선언
02
03     def __init__(self, n1, n2):
04         print('\n=== __init__() START ===')
05         self.num1 = n1
06         self.num2 = n2
07
08     def add(self):
09         print('\n=== add() START ===')
10         print('num1 + num2 =', self.num1 + self.num2)
11
12     def subtract(self):
13         print('\n=== subtract() START ===')
14         print('num1 - num2 =', self.num1 - self.num2)
```

속성 정의

메서드 정의

1. 클래스와 객체에 대한 설명으로 옳은 것은 무엇인가?

- ① 클래스는 객체를 생성하기 위한 틀이다. ② 객체는 클래스를 생성하기 위한 틀이다.
③ 객체는 클래스 없이 생성할 수 있다. ④ 클래스를 만들면 객체는 자동으로 만들어 진다.

2. 클래스 이름으로 가장 적절한 것은 무엇인가?

- ① mycalculator ② clock
③ Bicycle ④ student

1. ① 2. ③



■ 계산기 객체 만들기

- ‘객체 이름=계산기클래스(속성1, 속성2 ... 속성n)’ 형태로 작성

- 계산기 객체를 만드는 선언문

```
calc1 = Calculator(10, 20)
```

- 위 명령이 실행되면 객체 선언문의 10과 20이 Calculator 클래스의 `__init__()` 내부의 `num1`과 `num2` 속성에 각각 할당됨

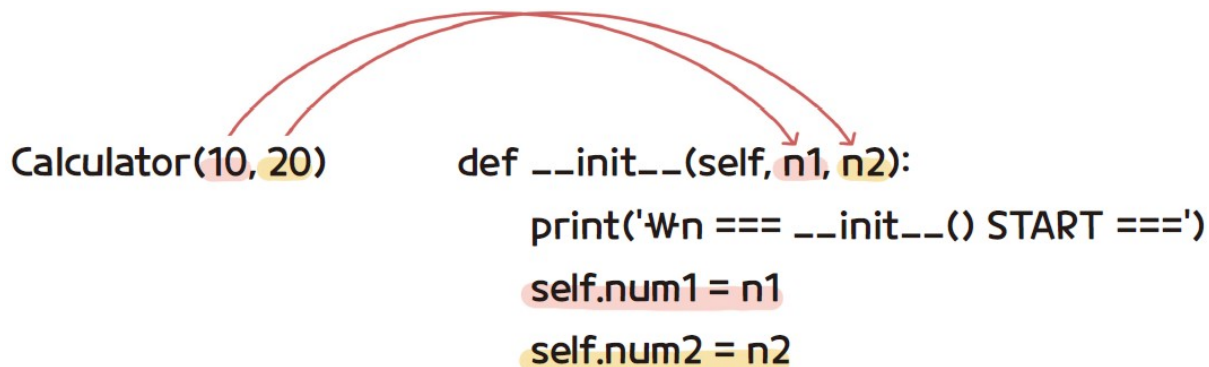


그림 13-6 num1과 num1 속성에 할당되는 매개변수 10, 20



■ __init__() 메서드의 역할

- __init__() 메서드는 객체를 메모리에 할당하여 생성하는 기능을 담당함
- 즉 객체를 생성할 때 클래스 이름을 사용하면 자동으로 __init__() 메서드가 호출되고 이것을 '생성자 호출'이라고 함

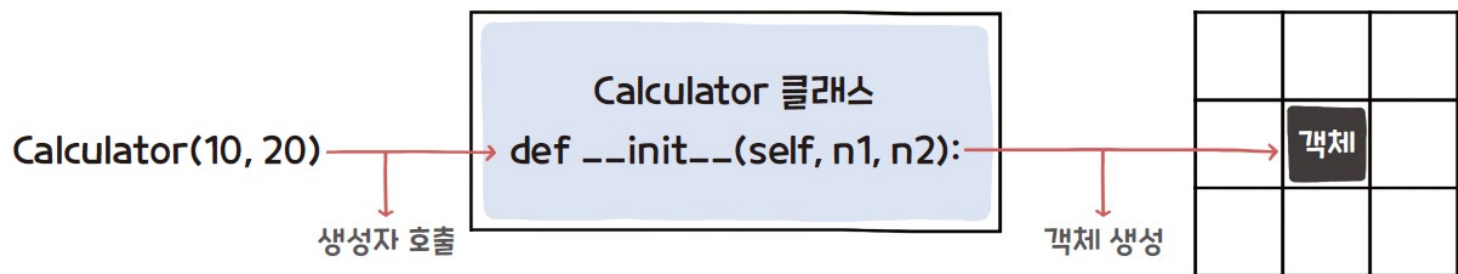


그림 13-7 클래스의 생성자 호출을 통해서 메모리에 생성되는 객체



■ __init__() 메서드의 역할

- [코드 13-1]에서 만든 Calculator 클래스를 이용하여 계산기 객체 3개 생성

코드 13-2

ch13_02.py

```
01 ...생략([코드 13-1]의 내용)...           # 클래스 선언
02
03 calc1 = Calculator(10, 20)
04 calc2 = Calculator(100, 200)
05 calc3 = Calculator(1000, 2000)
```

객체 생성

- Calculator 클래스를 이용해서 생성된 객체들을 메모리에 나타낸 모습

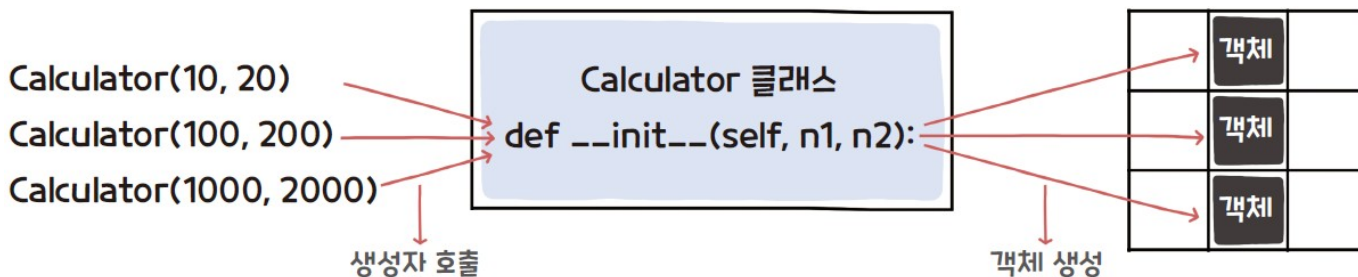


그림 13-8 메모리에 생성된 세 개의 객체



■ 도트 연산자(.)

- 모듈의 함수에 접근할 때 도트 연산자(.)를 사용했듯이 객체의 메서드를 호출하기 위해서도 도트 연산자(.)를 이용함

코드 13-3

ch13_03.py

```
01 ...생략([코드 13-1]의 내용)...
02
03 calc1 = Calculator(10, 20)
04 calc1.add()                # calc1의 add() 호출
05 calc1.subtract()           # calc1의 subtract() 호출
```

=== add() START ===

num1 + num2 = 30

calc1의 add()호출

=== subtract() START ===

num1 - num2 = -10

calc1의 subtract()호출



■ 도트 연산자(.)

- 객체의 속성에도 접근할 수 있음

코드 13-4

ch13_04.py

```
01 ...생략([코드 13-2]의 내용)...
02
03 print('calc1.num1 :', calc1.num1)      # calc1의 num1에 접근
04 print('calc1.num2 :', calc1.num2)      # calc1의 num2에 접근
05
06 print('calc2.num1 :', calc2.num1)      # calc2의 num1에 접근
07 print('calc2.num2 :', calc2.num2)      # calc2의 num2에 접근
08
09 print('calc3.num1 :', calc3.num1)      # calc3의 num1에 접근
10 print('calc3.num2 :', calc3.num2)      # calc3의 num2에 접근
```

```
calc1.num1 : 10
calc1.num2 : 20
calc2.num1 : 100
calc2.num2 : 200
calc3.num1 : 1000
calc3.num2 : 2000
```



■ 도트 연산자(.)

- 속성에 접근할 수 있으니 변경도 가능함

코드 13-5

ch13_05.py

```
01  ...생략([코드 13-2]의 내용)...
02
03  print('calc1.num1 :', calc1.num1)      # num1 출력
04  calc1.num1 = 123                       # calc1의 num1을 123으로 변경
05  print('calc1.num1 :', calc1.num1)      # 변경 후 num1 출력
```

calc1.num1 : 10 ● ————— 변경 전 데이터

calc1.num1 : 123 ● ————— 변경 후 데이터



다음 조건을 참고하여 자동차 객체를 정의하고 색상을 변경하는 프로그램을 만들어봅시다.

- ① 자동차 객체를 정의한다. 속성은 색상(color)과 길이(length), 기능은 전진(moveForward)과 후진(moveBackward)으로 정의한다.
- ② car1, car2 객체를 만들고 car1 객체의 색상과 길이는 'blue', 500, car2 객체의 색상과 길이는 'black', 450으로 할당한다.
- ③ car1과 car2 객체의 color 속성을 자신이 원하는 색상으로 변경한다.

```
01 class Car:
02     def __init__(self, c, l):
03         self.color = c
04         self.length = l
05     def moveForward(self):
06         print('=== moveForward() 호출 ===')
07
08     def moveBackward(self):
09         print('=== moveBackward() 호출 ===')
10
11 car1 = Car('blue', 500)
12 car2 = Car('black', 450)
13
14 print('car1의 색상 :', car1.color)
15 print('car2의 색상 :', car2.color)
```

```
16
17 car1.color = input('원하는 색상을 입력하세요. ')
18 car2.color = input('원하는 색상을 입력하세요. ')
19
20 print('car1의 색상 :', car1.color)
21 print('car2의 색상 :', car2.color)
```

```
car1의 색상 : blue
car2의 색상 : black
원하는 색상을 입력하세요. Yellow
원하는 색상을 입력하세요. Orange
car1의 색상 : Yellow
car2의 색상 : Orange
```

Section 04

객체를 이용한 회원가입 프로그램



■ 회원과 회원 관리 시스템

■ 회원(Member) 클래스

- 회원 개인의 아이디와 비밀번호를 담고 있는 객체를 생성

■ 회원 관리 시스템(MemberManage) 클래스

- 회원 등록 및 삭제 메서드 : addMember(), removeMember()
- 로그인 메서드 : loginMember()

표 13-2 각 클래스에 필요한 속성과 메서드

클래스	구분	이름	내용
회원 (Member)	속성	mId	회원 아이디
		mPw	회원 비밀번호
회원 관리 시스템 (MemberManage)	속성	members	전체 회원 리스트(dictionary 형)
	메서드	addMember()	회원 등록
		loginMember()	로그인
		removeMember()	회원 삭제
		printMembers()	전체 회원 정보 출력



코드 13-6

ch13_06.py

```
01 class Member:
02     def __init__(self, i, p):
03         self.mId = i
04         self.mPw = p
05
06 class MemberManage:
07     def __init__(self):
08         self.members = {}
09
10     def addMember(self, m):
11         self.members[m.mId] = m.mPw
12
13     def loginMember(self, i, p):
14         isMember = i in self.members
15
16         print('==== 로그인 결과 ====')
17         if isMember and self.members[i] == p:
18             print(i, '님 로그인 성공')
19         else:
20             print(i, '님 ID와 PW를 다시 확인하세요!')
21
22     def removeMember(self, i, p):
23         del self.members[i]
24
```

객체를 이용한 회원 관리 시스템



```
25     def printMembers(self):
26         print('==== 전체 회원 ====')
27         for member in self.members.keys():
28             print('ID :', member)
29             print('PW :', self.members[member])
30             print('-----')
31
32 mm = MemberManage()
33
34 mm.addMember(Member('chanho@gmail.com', '1234a!'))
35 mm.addMember(Member('seri@gmail.com', '5678b^'))
36 mm.addMember(Member('heungmin@gmail.com', '9852c#'))
37
38 mm.printMembers()           # 전체 회원 출력
39
40 mm.loginMember('chanho@gmail.com', '1234a!')
41 mm.loginMember('seri@gmail.com', '99999')
42
43 mm.removeMember('chanho@gmail.com', '1234a!')
44 mm.removeMember('heungmin@gmail.com', '9852c#')
45
46 mm.printMembers()           # 회원 삭제 후 전체 회원 출력
```

```
==== 전체 회원 ====
ID : chanho@gmail.com
PW : 1234a!
-----
ID : seri@gmail.com
PW : 5678b^
-----
ID : heungmin@gmail.com
PW : 9852c#
-----
==== 로그인 결과 ====
chanho@gmail.com 님 로그인 성공
==== 로그인 결과 ====
seri@gmail.com 님 ID와 PW를 다시 확인하세요!
==== 전체 회원 ====
ID : seri@gmail.com
PW : 5678b^
-----
```

Section 05

텍스트 파일 다루기



■ 텍스트 파일을 다루는 3단계

▪ [1단계] 파일 열기

- 첫 번째, 파일을 여는 단계입니다.
- 파일을 열기 위해서는 `open()` 함수를 이용함
- 파일 열기에 성공하면 파일은 객체로 만들어져 메모리에 생성됨

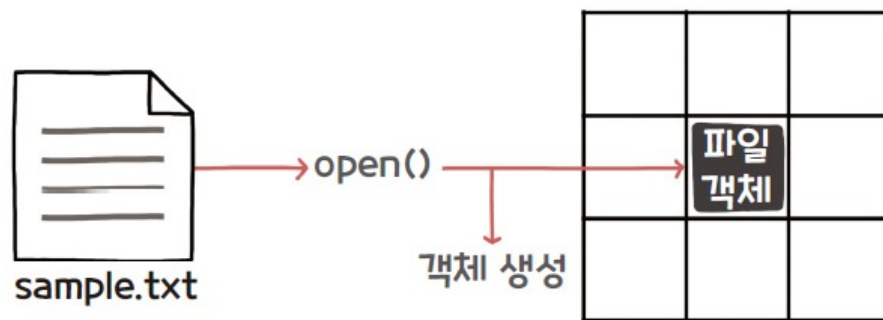


그림 13-9 `open()`에 의해서 객체로 생성됨



■ 텍스트 파일을 다루는 3단계

■ [2단계] 파일 쓰기/읽기

- 두 번째, 문자열을 쓰거나 읽는 단계
- 문자열을 쓸 때는 `write()` 함수를, 읽을 때는 `read()` 함수를 이용함

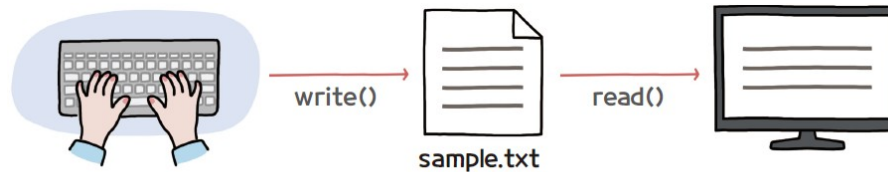


그림 13-10 `write()`와 `read()`를 이용한 쓰기, 읽기

■ [3단계] 파일 닫기

- 세 번째, 파일을 닫는 단계
- 쓰기 또는 읽기가 끝난 파일은 `close()` 함수를 이용해서 연결을 해제함

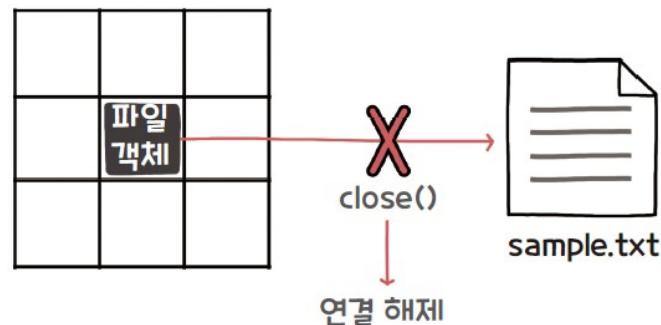


그림 13-11 `close()`에 의해서 연결 해제됨



■ 텍스트 파일을 다루는 3단계 정리

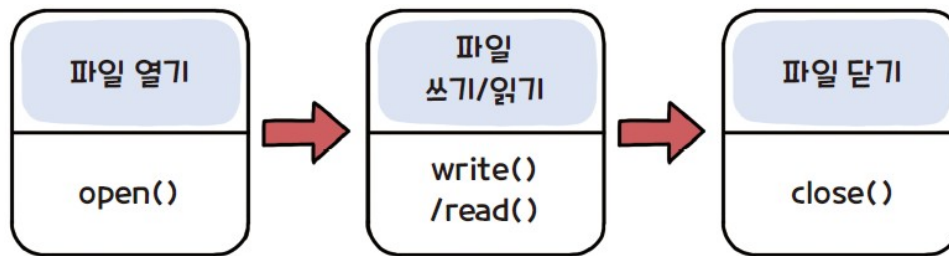


그림 13-12 파일을 다루는 3단계



■ 파일 열기

- 파일을 열 때 사용하는 함수 open()은 두 개의 인수가 필요함
- 첫 번째 인수는 파일의 경로이고, 두 번째 인수는 파일 모드임

```
open('C:\python\test.txt', 'w')
```

①파일의 경로

②파일 모드

그림 13-13 open() 함수의 형식

- 파일 모드는 사용 목적에 따라서 파일을 여는 모드를 설정함

표 13-3 파일 모드

모드	사용 목적	특징
w	쓰기	기존 파일이 존재하면 새로운 파일로 덮어쓴다.
a	쓰기	기존 파일이 존재하면 기존 파일에 덧붙인다.
x	쓰기	기존 파일이 존재하면 에러가 발생한다.
r	읽기	파일을 읽는다. 파일이 없으면 에러가 발생한다.



■ 파일 열기

- 쓰기(w) 모드로 파일을 열 때, 만약 해당 경로에 '*.txt' 파일이 없다면 새로운 파일(*.txt)을 생성함

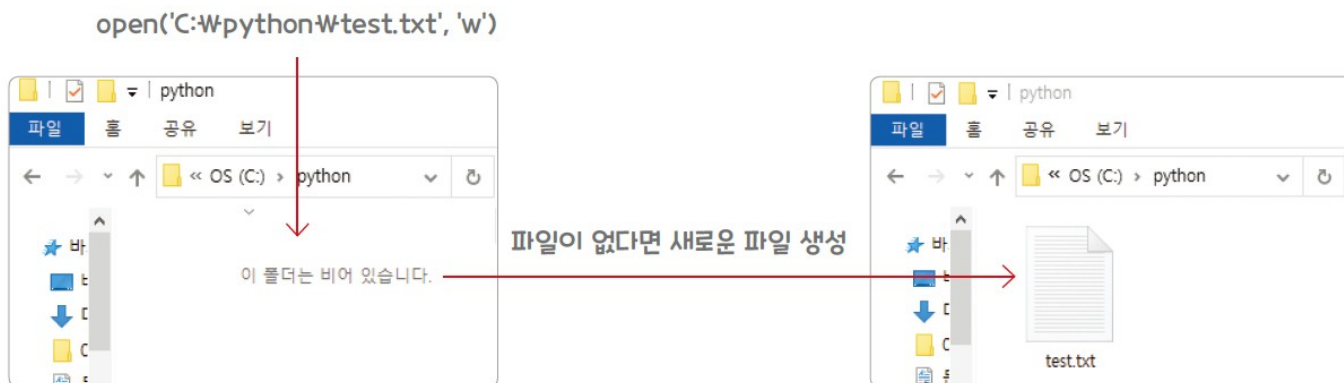


그림 13-14 오픈할 파일이 없다면 새로운 파일 생성



■ 파일 닫기

- 작업을 마치고 파일을 닫을 때는 `close()` 함수를 사용함

`file.close()`

그림 13-15 `close()` 함수의 형식

하나 더 알기 ✓

파일 작업이 끝난 후에 `close()`를 호출하지 않으면 어떻게 되나요?

파일을 제대로 닫지 않으면 파일을 필요로 하는 다른 곳에서 사용하지 못 할 수 있습니다. 또한 사용이 끝난 외부 파일이 메모리에 계속 남아 있기 때문에 시스템에 부하가 걸려 최악의 경우 시스템이 멈추는 현상까지 발생할 수 있습니다. 따라서 파일을 열어 쓰거나 읽기한 후에는 반드시 파일을 닫아야 합니다.





확인문제

1. 텍스트 파일을 읽기 또는 쓰기 위한 과정으로 옳은 것은 무엇인가?

- ① save → read → open → close
- ② read → open → close
- ③ open → write / read → close
- ④ close → write / read → open

2. 다음 중 파일을 다루는 함수에 대한 설명으로 옳은 것은 무엇인가?

- ① open() : 파일을 저장하기 위한 함수이다.
- ② open() : 파일을 열기 위한 함수이다.
- ③ close() : 파일을 삭제하기 위한 함수이다.
- ④ close() : 파일을 암호화 처리하기 위한 함수이다.

정답

1. ③

2. ②



■ 파일 쓰기

- 파일에 문자열을 쓸 때는 write() 함수를 사용함

```
file.write('Hello world~')
```

그림 13-16 write() 함수의 형식

■ write() 함수

- 파일에 '쓰기'를 실행한 후 데이터를 반환함
- 반환값은 파일에 '쓰기'한 문자의 개수로 공백 문자까지 포함됨



■ 파일 쓰기

코드 13-7

ch13_07.py

```
01 file = open('C:/python/test.txt', 'w')    # 파일 열기(쓰기 모드)
02 result = file.write('Hello python~')      # 쓰기
03 print('type of result :', type(result))    # 반환값의 데이터 타입 출력
04 print('result :', result)                  # 반환값 출력
05 file.close()                              # 파일 닫기
```

```
type of result : <class 'int'>
result : 13
```

write()의 반환값은 꼭 사용해야 하는 것은 아닙니다. 주로 문자열이 파일에 정상적으로 쓰였는지 확인하는 용도로 사용합니다.

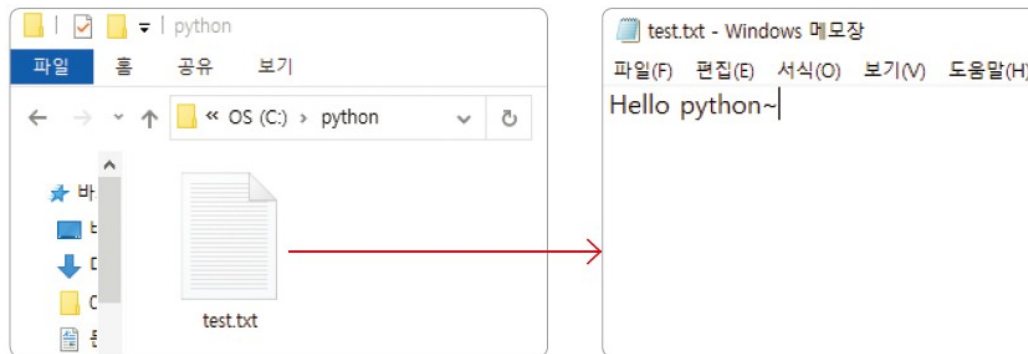


그림 13-17 test.txt에 기록된 문자열 확인



■ 파일 읽기

- 파일을 읽을 때는 read() 함수를 사용함

file.read()

그림 13-18 read() 함수의 형식

- read()를 이용해서 파일에 문자열을 읽기 위한 코드

코드 13-8

ch13_08.py

```
01 file = open('C:/python/test.txt', 'r')    # 파일 열기(읽기 모드)
02 result = file.read()                      # 읽기
03 print('type of result :', type(result))    # 반환값의 데이터 타입 출력
04 print('result :', result)
05 file.close()
```

```
type of result : <class 'str'>
result : Hello python~
```




■ 파일 읽기

- read()로 읽어 들인 데이터 타입은 항상 str(문자열)임

코드 13-9

ch13_09.py

```
01 file = open('C:/python/number.txt', 'r')
02 result = file.read()
03 print('result :', result)
04 sum = result + 1
05 print('sum :', sum)
06 file.close()
```

result : 123

Traceback (most recent call last):

sum = result + 1

TypeError: can only concatenate str (not "int") to str

- 123을 read()로 읽어 들이면 숫자 123이 아닌 문자열 '123'으로 읽기 때문에 문자열에 정수 1을 더할 수 없어 에러가 발생함



■ 파일 읽기

- [코드 13-9]의 문자열을 정수로 변환하고 1을 더해 에러를 없앤 코드

코드 13-10

ch13_10.py

```
01 file = open('C:/python/number.txt', 'r')
02 result = file.read()
03 print('result :', result)
04 sum = int(result) + 1
05 print('sum :', sum)
06 file.close()
```

문자열을 정수로 변환

result : 123

sum : 124

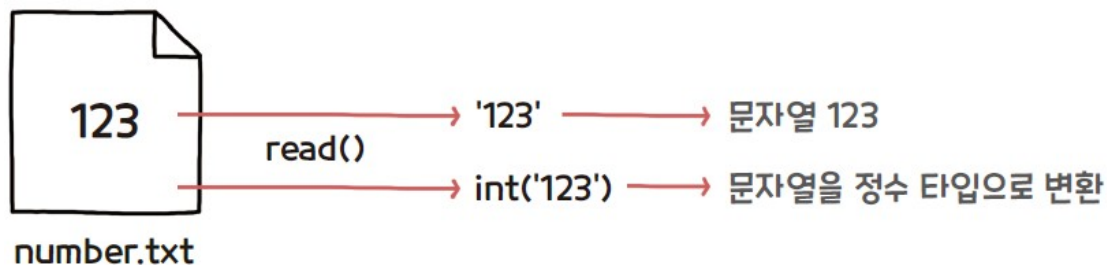


그림 13-19 텍스트의 기본 반환 값인 문자열



문제 해결 13-2

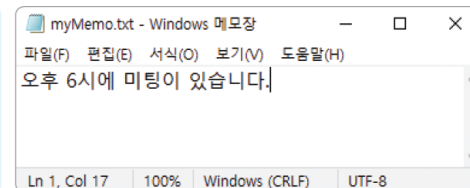
일정 관리 메모를 입력하여 텍스트 파일에 저장하기

ch13_sol_02.py

길동이는 미팅 약속을 잊어버리지 않기 위해 메모장에 기록으로 남겨두려고 합니다. 길동이가 입력한 내용을 텍스트 파일로 저장하는 프로그램을 만들어봅시다.

```
01 memo = input('메모를 입력하세요. ')
02
03 file = open('C:/python/myMemo.txt', 'a')
04 file.write(memo)
05 file.close()
```

메모를 입력하세요. 오후 6시에 미팅이 있습니다.



실전 예제

실전 예제1 – 주사위 게임



문제

3명이 주사위를 5번씩 굴려서 나온 합을 구하는 프로그램을 만들어봅시다. 합이 가장 큰 사람이 이기는 게임입니다.



```
Gamer1 : [1, 3, 2, 3, 3]
```

```
Sum of Gamer1 : 12
```

```
Gamer2 : [1, 4, 2, 4, 4]
```

```
Sum of Gamer2 : 15
```

```
Gamer3 : [4, 3, 2, 1, 3]
```

```
Sum of Gamer3 : 13
```

```
=====
```

```
1등 : 15점      WIN!!
```

```
2등 : 13점
```

```
3등 : 12점
```

```
=====
```

실전 예제1 – 주사위 게임



해결

```
01 import random
02
03 class Dice:
04     def __init__(self):
05         self.numbers = []
06
07     def playDice(self):
08         self.numbers.append(random.randint(1, 6));
09
10     def getNumbers(self):
11         return self.numbers
12
13     def getSum(self):
14         return sum(self.numbers)
15
16 def sortNumbers(*numbers):
17     list = sorted(numbers)
18     list.sort(reverse = True)
19     return list
20
21
```

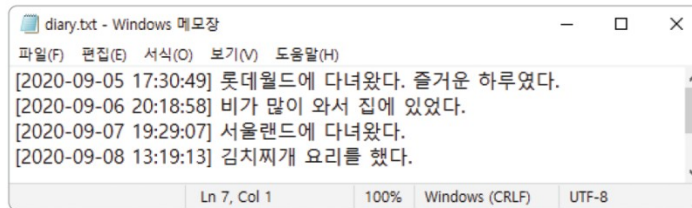
ch13_appEx_01.py

```
22 gamer1Dice = Dice()
23 gamer2Dice = Dice()
24 gamer3Dice = Dice()
25
26 for i in range(5):
27     gamer1Dice.playDice()
28     gamer2Dice.playDice()
29     gamer3Dice.playDice()
30
31 print('Gamer1 :', gamer1Dice.getNumbers())
32 print('Sum of Gamer1 :', gamer1Dice.getSum())
33 print('-'*25)
34
35 print('Gamer2 :', gamer2Dice.getNumbers())
36 print('Sum of Gamer2 :', gamer2Dice.getSum())
37 print('-'*25)
38
39 print('Gamer3 :', gamer3Dice.getNumbers())
40 print('Sum of Gamer3 :', gamer3Dice.getSum())
41 print('-'*25)
42
43 sortedNumbers = sortNumbers(gamer1Dice.getSum(), gamer2Dice.getSum(), gamer3Dice.getSum())
44 print('='*25)
45 print('1등 :', sortedNumbers[0], '점\tWIN!!')
46 print('2등 :', sortedNumbers[1], '점')
47 print('3등 :', sortedNumbers[2], '점')
48 print('='*25)
```



문제

하루 중 인상 깊었던 일을 파일에 한 줄로 작성하는 ‘한 줄 일기장’ 프로그램을 만들어봅시다.





해결

ch13_appEx_02.py

```
01 from time import strftime, localtime
02
03 def getDay():
04     return strftime('%Y-%m-%d', localtime())
05
06 def getTime():
07     return strftime('%H:%M:%S', localtime())
08
09 print('***** 한 줄 일기 *****')
10
11 dFlag = True
12 while dFlag:
13     print('\n다음 항목을 선택하세요.')
14     selectItem = int(input('1.일기 작성\t2.일기 보기\t3.종료 '))
15
16     if selectItem == 1:
17         print('\n[' + getDay() + '] 한 줄 일기를 작성하세요.')
18         todayDiary = input()
19
20         with open('C:/python/diary.txt', 'a') as f:
21             f.write([' + getDay() + ' ' + getTime() + '] ')
22             f.write(todayDiary + '\n')
23
24     elif selectItem == 2:
25
26         with open('C:/python/diary.txt', 'r') as f:
27             str = f.read()
28             print(str)
29
30     elif selectItem == 3:
31         print('\nBye~~')
32         dFlag = False
```


Thank you!