파이썬 프로그래밍

(Python Programming)

- **정수, 실수 데이터 다루기**

[basic] ex01.py

```
#문자형 변수
a = "Hello, World."
print(a, type(a))

#정수형 변수
a=100
print(type(a))

#실수형 변수
b=200.5
print(type(b))

#불린 변수
a = True
print(a, type(a))

#사칙 연산
add = a + b
subtract = a - b
multiply = a * b
divide = a / b
quotient = a // b
remainder = a % b
#사칙 연산 결과 출력
print(a, "+", b, "=", add)
print(a, "-", b, "=", subtract)
print(a, "x", b, "=", multiply)
print(a, "/", b, "=", quotient)
print(a, "%", b, "=", remainder)
print(type(divide))

#관계 연산자
a = 10
b = 20
c = a < b
#관계 연산 결과 출력
print(c, type(c))
print(a > b)
print(a >= b)
print(a < b)
print(a <= b)
print(a == b)
print(a != b)

#논리 연산자
a = True
b = False
#논리 연산 결과 출력
print(a & b)
print(a | b)
print(not a)
print(not b)
```

- math 모듈을 이용한 연산

[basic] ex02.py

```
#절대값
print("절대값=", abs(-10))
#실행 결과: 절대값= 10

#반올림
print("반올림=", round(1.4))
print("반올림=", round(1.5))
#실행 결과: 반올림= 1
#실행 결과: 반올림= 2

import math

#PI
print("파이=", math.pi)
#실행 결과: 파이= 3.141592653589793

#버림
print("버림=", math.trunc(1.5))
#실행 결과: 버림= 1

#올림
print("올림=", math.ceil(1.4))
#실행 결과: 올림= 2

#radian -> degree
print(math.degrees(math.pi))
#실행 결과: 180.0

#degree -> radians
print(math.radians(180))
#실행 결과: 3.141592653589793

#삼각함수
print("sin(90도)=", math.sin(math.radians(90)))
#실행 결과: sin(90도)= 1.0

print("cos(180도)=", math.cos(math.radians(180)))
#실행 결과: cos(180도)= -1.0

#제곱과 제곱근
print("3의 2제곱=", math.pow(3, 2))
#실행 결과: 3의 2제곱= 9.0
print("3의 3제곱=", 3**3)
#실행 결과: 4의 제곱근= 2.0
print("4의 제곱근=", math.sqrt(4))
#실행 결과: 4의 제곱근= 2.0

#팩토리얼
print("5팩토리얼=", math.factorial(5))
#실행 결과: 5팩토리얼= 120

#로그
print("밑수 2에 대한 4의 로그값=", math.log(4, 2))
#실행 결과: 밑수 2에 대한 4의 로그값= 2.0
print("밑수 10에 대한 1000이 로그값=", math.log10(1000))
#실행 결과: 밑수 10에 대한 1000이 로그값= 3.0
```

- **텍스트 다루기**

[basic] ex03.py

```
s = "Good Morning"

#문자열 분리
print("s=", s)
print("s[0:4]=", s[0:4])
print("s[:4]", s[:4])
print("s[0]", s[0])

#포함 결과를 true, false 출력
print('Good in s=', 'Good' in s)
print('X in s=', 'X' in s)

#문자열 길이
print('s 문자열의 길이=', len(s))

#특정 문자열 위치 및 개수 출력
print('M의 위치=', s.find('M'))
print('o의 갯수=', s.count('o'))

#문자열 대, 소문자 변경
s='   Good Morning   '
print('문자열의 앞, 뒤 공백제거=', s.strip())
print('문자열을 모두 대문자로 변경=', s.upper())
print('문자열을 모두 소문자로 변경=', s.lower())

#특정 문자열로 변경
s='Hello, World'
print('World를 Korea로 변경=', s.replace('World', 'Korea'))

s='Apple, Orange, Banna'
b=s.split()
print(',로 분리=', b)
print(type(b))

#문자열 format
a='My name is {0}. I am {1} years old'.format('Chris', 23)
print(a)

#텍스트를 수로 변경
a='1234567890'
print(a, type(a))
b=int(a)
print(b, type(b))
b=float(a)
print(b, type(b))

#숫자를 텍스트로 변경
a=1234567890
print(a, type(a))
b=str(a)
print(b, type(b))
```

- **다음은 두 수를 입력 받아 두 수를 곱하는 예제 코드이다.**

[basic] ex04.py

```
a = input("첫 번째 수를 입력하세요? ")
b = input("두 번째 수를 입력하세요? ")
result = int(a) * int(b)
print("{0} * {1} = {2}".format(a, b, result))
```

- **List**

[basic] ex05.py

```
#리스트 생성
a =['홍길동', '심청이', '강감찬']
print(a, type(a))
print(a[0])
print(a[1])

#리스트 슬라이싱
a=[1, 2, 2, 4, 5, 6, 7, 8, 9, 10]
print(a[0:5])
print(a[5:])
print(a[:3])

#리스트 결합
a=[1, 2, 3, 4, 5]
b=[5, 6, 7]
print(a + b)

#특정 위치의 리스트 값 변경
a[2] = 300
print(a[2])

#리스트의 길이
print(len(a))

#리스트 끝에 새 요소를 추가
a.append(4)
print(a)

#기존 리스트에 다른 리스트를 이어 붙인다.
a.extend([4, 5, 6])
print(a)

#리스트 내의 위치에 새 요소를 삽입한다.
a.insert(0, 1)
print(a)

#데이터를 찾아 발견한 첫 번째 요소를 제거한다.
a = ['홍길동', '이순신', '강감찬']
a.remove('홍길동')
print(a)

#리스트의 마지막 요소를 제거하거나 특정 위치의 값을 제거한다.
a=[1, 2, 3, 4, 5]
a.pop()
print(a)
a.pop(1)
print(a)

#리스트 내에서 특정 값과 일치하는 첫 번째 요소의 첨자를 알려준다.
a = ['홍길동', '이순신', '강감찬']
print(a.index('이순신'))

#특정 값과 일치하는 요소가 몇 개 있는지 카운트한다.
a = [1, 100, 2, 100, 3, 100]
print(a.count(100))

#리스트 내의 요소를 정렬한다. (revers=True)는 내림차순, 생략은 오름차순
a = [3, 4, 5, 1, 2]
a.sort()
print(a)
a.sort(reverse=True)
print(a)

#리스트 내 요소의 순서를 반대로 뒤집는다.
a = [3, 4, 5, 1, 2]
a.reverse()
print(a)
```

- **Tuple**

데이터 값 변경이 불가능하고 위경도, 좌표값, RGB 색상처럼 작은 규모의 자료 구조를 구성하기에 좋다.

[basic] ex06.py

```
#Tuple 생성 및 슬라이싱
a = (1, 2, 3, 4, 5, 6)
print(type(a))
#실행 결과: <class 'tuple'>
print(a[0])
print(a[:3])
#실행 결과: (1, 2, 3)
print(a[4:6])

#Tuple간의 결합
a = (1, 2, 3)
b = (4, 5, 6)
c = a + b
print(c)
#실행 결과: (1, 2, 3, 4, 5, 6)

#Tuple의 길이 구하기
a = (1, 2, 3)
print(len(a))
#실행 결과: 3

#Tuple 패킹: 여러 가지 데이터를 묶는다.
a = 1, 2, 3
print(a)
#실행 결과: (1, 2, 3)
print(type(a))
#실행 결과: <class 'tuple'>

#Tuple 언 패킹: 각 요소를 여러 개의 변수에 할당한다.
a = 1, 2, 3
one, two, three = a
print(one)
#실행 결과: 1
print(type(one))
#실행 결과: <class 'int'>
print(two)
print(three)

city, latitude, longitude = 'Seoul', 37.541, 126.986
print(city)
print(latitude)
print(longitude)

#특정 값과 일치하는 인덱스 값을 알려준다.
a = ('홍길동', '심청이', '이순신')
print(a.index('심청이'))
#실행 결과: 1

#특정 값과 일치하는 요소의 수 출력
a = (1, 100, 2, 100, 3, 100)
print(a.count(100))
```

- **Dictionary**

첨자는 키(key)라고 하고 이 키가 가리키는 슬롯에 저장되는 데이터를 값(value)라고 한다. Dictionary는 키-값 쌍으로 구성된다.

[basic] ex07.py

```
#Dictionary 생성
dic = {}
print(type(dic))
#실행결과: <class 'dict'>

#Dictionary 값 저장
dic['파이썬'] = 'www.python.org'
dic['마이크로소프트'] = 'www.microsoft.com'
dic['애플'] = 'www.apple.com'

print(dic['파이썬'])
print(dic['마이크로소프트'])
print(dic['애플'])

#Dictionary 값 출력
print(dic.get('애플'))
#실행 결과: www.apple.com
print(dic.get('자바'))
#실행 결과: None
print(dic.get('자바', '과목 없음'))
#실행 결과: 과목 없음
print(dic)
#실행결과: {'파이썬': 'www.python.org', '마이크로소프트': 'www.microsoft.com', '애플': 'www.apple.com'}

#Dictionary에 저장되어 있는 키의 목록과 값의 목록 출력
print(dic.keys())
#실행 결과: dict_keys(['파이썬', '마이크로소프트', '애플'])
print(dic.values())
#실행 결과: dict_items([('파이썬', 'www.python.org'), ('마이크로소프트', 'www.microsoft.com'), ('애플', 'www.apple.com')])

#키와 값의 쌍으로 이루어진 전체 목록 반환
a = dic.items()
print(a)
#실행 결과: dict_items([('파이썬', 'www.python.org'), ('마이크로소프트', 'www.microsoft.com'), ('애플', 'www.apple.com')])
print(type(a))
#실행 결과: <class 'dict_items'>

#특정키가 목록 안에 존재하는지 확인
print('애플' in dic.keys())
#실행 결과: True
print('www.microsoft.com' in dic.values())

#Dictionary 안에 키-값 쌍을 제거
dic.pop('애플')
print(dic)
#실행 결과: {'파이썬': 'www.python.org', '마이크로소프트': 'www.microsoft.com'}

#Dictionary 데이터를 남김없이 정리
dic.clear()
print(dic)
#실행결과: {}
```

[basic] ex08.py

```
#집합 생성
my_set = { 6, 3, 5, 1, 2, 3, 4 }
print(my_set)
#출력 결과
#{ 1, 2, 3, 4, 5, 6 }


print(type(my_set))
#집합의 교집합, 합집합, 차집합
java = {"홍길동", "이순신", "강감찬"}
python = set(["홍길동", "성춘향"])
print(java)
print(python)

print(java & python)
print(java.intersection(python))

print(java | python)
print(java.union(python))

print( java - python)
print( java.difference(python))

python.remove("성춘향")
print(python)

#자료구조 변경
menu = {"커피", "우유", "주스"}
print(menu, type(menu))
#실행 결과: {'우유', '주스', '커피'} <class 'set'>

menu = list(menu)
print(menu, type(menu))
#실행 결과: ['우유', '주스', '커피'] <class 'list'>

menu = tuple(menu)
print(menu, type(menu))
#실행 결과: ('우유', '주스', '커피') <class 'tuple'>

menu = set(menu)
print(menu, type(menu))
#실행 결과: {'주스', '우유', '커피'} <class 'set'>
```

- **예제 프로그램**

조건1: 편의상 댓글은 20명 작성하였고 아이디는 1~20 이라고 가정한다.
조건2: 댓글 내용과 상관없이 무작위로 추첨하되 중복 불가 한다.
조건2: 댓글 작성자 20명중 추첨을 통해 1명은 치킨, 2명은 커피 쿠폰을 받는 프로그램을 작성하시오.

[basic] ex09.py

```
from random import *

users = list(range(1, 21))
print(users)
shuffle(users)
print(users)
winners = sample(users, 4)
print(winners)
print("치킨 당첨자: {0}".format(winners[0]))
print("커피 당첨자: {0}".format(winners[1:]))
```

- **분기문**

[basic] ex10.py

```
import sys

kor = input("국어 점수를 입력하세요? ")

#단순 if ~ else 문
if(kor.isnumeric() != True):
    print("국어 점수를 숫자로 입력하세요!")
    sys.exit(0)
else:
    kor = int(kor)

eng = input("영어 점수를 입력하세요? ")
if(eng.isnumeric() == False):
    print("영어 점수를 숫자로 입력하세요!")
    sys.exit(0)
else:
    eng = int(eng)

mat = input("수학 점수를 입력하세요? ")
if(mat.isnumeric() == False):
    print("수학 점수를 숫자로 입력하세요!")
    sys.exit(0)
else:
    mat = int(mat)

avg = (kor + eng + mat)/3

#if ~ elif ~ else 문
if(avg >= 90):
    print("{0}의 학점은 A입니다.".format(avg))
elif(avg >= 80):
    print("{0}의 학점은 B입니다.".format(avg))
elif(avg >= 70):
    print("{0}의 학점은 C입니다.".format(avg))
elif(avg >= 60):
    print("{0}의 학점은 D입니다.".format(avg))
else:
    print("{0}의 학점은 F입니다.".format(avg))

#다중 if 문
if(avg >=90):
    if(avg >= 95):
        grade="A+"
    else:
        grade="A0"
elif(avg >= 80):
    if(avg >= 85):
        grade="B+"
    else:
        grade = "B0"
elif(avg >= 70):
    if(avg >= 75):
        grade="C+"
    else:
        grade="C0"
elif(avg >= 60):
    if(avg >= 65):
        grade="D+"
    else:
        grade="D0"
else:
    grade ="F"

print("{0}의 학점은 {1}입니다.".format(avg, grade))
```

- **반복문**

[basic] ex11.py

```
limit = int(input("숫자를 입력하세요? "))

sum = 0
count = 0
while count < limit:
    count = count + 1
    sum = sum + count
    print("{0}회 반복".format(count))

print("{0}까지의 합계는 {1}입니다.".format(count, sum))
```

[basic] ex12.py

```
while True:
    number = input("숫자를 입력하세요(종료: 0) ? ")
    if(not number.isnumeric()) :
        print("입력하신 값이 숫자가 아닙니다.")
    elif( number == "0" ):
        break
    else:
        print("입력하신 값은 {0}입니다.".format(number))

print("프로그램을 종료합니다.")
```

[basic] ex13.py

```
#1 ~ 10까지의 합계 출력
sum=0
for i in range(1, 11, 1):
    print("i={0}".format(i))
    sum = sum + i
print("1부터 10까지의 합은 {0}입니다.".format(sum))



#1 ~ 10 짝수 출력
for i in range(0, 10, 2):
    if(i % 2 == 1):
        continue
    print("{0}".format(i))

#이름 리스트 출력
for name in ["홍길동", "심청이", "강감찬", "이순신"]:
    print(name)

#문자열 출력
for str in "안녕하세요?":
    print(str)

#Dictionary 출력
dic = {"애플": "www.apple.com", "파이썬": "www.python.org", "마이크로소프트": "www.microsoft.com"}
for key, value in dic.items():
    print("키: {0}, 값: {1}".format(key, value))
```

- **함수 정의 및 사용하기**

[basic] ex14.py

```
#기본 함수 만들기
def my_abs(number):
    if(number < 0):
        result = number * -1
    else:
        result = number
    return result
print("결과: ", my_abs(-5))
print("결과: ", my_abs(10))

#매개변수에 초기 값 설정
def print_string(text, number=1):
    for i in range(number):
        print(text)
print_string("안녕하세요!", 5)
print_string("반갑습니다!")

#가변 매개변수 Tuple
def merge_string(*text_list):
    print(type(text_list))
    result = ""
    for text in text_list:
        result = result + text
    return result
print(merge_string("홍길동", "이순신", "심청이"))

#가변 매개변수 Dictionary
def print_words(**words):
    print(type(words))
    for key in words.keys():
        print("{0} = {1}".format(key, words[key]))
print(print_words(고양이="cat", 강아지="dog", 소년="boy", 소녀="girl"))

#가변 매개변수와 일반 매개변수 함께 사용하기
def print_students(*names, count):
    for i in range(count):
        print(names[i])
print_students("홍길동", "심청이", "강감찬", count=3)

#일반 매개변수와 가변 매개변수 함께 사용하기
def print_students(count, *names):
   for i in range(count):
        print(names[i])
print_students(3, "홍길동", "심청이", "강감찬")

#함수를 변수에 담아 사용하기
def print_something(text):
    print(text)
something = print_something
something("안녕하세요!")
```

- **모듈과 패키지**

Python에서는 각각의 소스 파일을 일컬어 모듈이라고 한다. 모듈은 제공자에 따라 표준모듈, 사용자 생성 모듈, 서드파티 모듈로 나눈다. 표준 모듈은 Python과 함께 따라오는 모듈, 사용자 생성 모듈은 프로그래머가 직접 작성한 모듈이다. 그리고 서드파티(3rd Party) 모듈은 Python 재단도 프로그래머도 아닌 다른 프로그래머 또는 업체에서 제공한 모듈을 말한다.

1) 두 개의 소스 파일로 만드는 하나의 프로그램 예제

[basic]-[module] calculator.py

```
def plus(a, b):
    return a + b

def minus(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    return a / b
```

[c:]-[module] calc_tester.py

```
#불러올 모듈의 이름 calculator.py에서 .py는 생략한다.
import calculator


#모듈이름.함수()의 꼴로 calculator 모듈의 함수를 호출한다.
print(calculator.plus(10, 5))
print(calculator.minus(10, 5))
print(calculator.multiply(10, 5))
print(calculator.divide(10, 5))
```

2) import에 대해

import의 역할은 정확하게는 '다른 모듈 내의 코드에 대한 접근'을 가능하게 하는 것이다. import가 접근 가능하게 하는 코드에는 변수, 함수, 클래스 등이 모두 포함된다. 아래 표는 앞에서 만든 calculator.py 모듈을 불러오는 두 가지 스타일의 코드예제이다.

import 모듈	form 모듈 import 변수 또는 함수
import calculator print(calculator.plus(10, 5)) print(calculator.minus(10, 5))	from calculator import plus from calculator import minus print(plus(10, 5)) print(minus(10, 5))

'import calculator' 코드를 통해 모듈을 불러와서 calculator 안에 정의되어 있는 함수를 호출할 때 calculator라는 모듈 이름을 일일이 입력하기가 귀찮을 경우에는 아래와 같이 as 키워드를 사용하여 해결할 수 있다.

[c:]-[module] calc_tester.py

```
#calculator 모듈을 c라는 이름으로 불러온다.
import calculator as c


#calculator라는 이름 대신 c를 이용해 함수 이름에 접근한다.
print(c.plus(10, 5))
print(c.minus(10, 5))
print(c.multiply(10, 5))
print(c.divide(10, 5))
```

- **클래스**

1) 클래스 정의

[basic]-[class] Car.py

```
class Car:
    def __init__(self):  #객체가 생성될 때 호출되는 메서드로써 객체의 초기화를 담당한다.
        self.color = "검정"
        self.wheel_size = 16
        self.displacement = 2000

    def forward(self):  #매개변수 self는 메서드를 정의할 때는 명시적으로 정의하지만 호출할 때는 자동으로 해당 매개변수를 넘겨준다.
        print("전진합니다.")

    def backword(self):
        pass

    def turn_left(self):
        print("좌회전 합니다.")

    def turn_right(self):
        print("우회전 합니다.")


if __name__ == '__main__':
    my_car = Car()
    print("색상: {0}".format(my_car.color))
    print("바퀴 사이즈: {0}".format(my_car.wheel_size))
    print("배기량: {0}".format(my_car.displacement))
```

2) __init__() 메서드를 이용한 초기화

__init__() 메서드를 이용하면 Instance가 생성될 때에만 __init__() 메서드가 실행되므로 이 메서드 안에 변수의 정의/초기화 코드를 넣으면 모든 Instance가 변수를 공유하게 되는 불상사를 피할 수 있다. 아래는 __init__() 메서드를 이용해 클래스를 초기화하는 예제이다.

[basic]-[module]-[class] Instance.py

```
class Instance:
    def __init__(self):
        self.text_list = []

    def add(self, text):
        self.text_list.append(text)

    def print_list(self):
        print(self.text_list)

if __name__ == '__main__':
    a = Instance()
    a.add("홍길동")
    a.print_list()

    b = Instance()
    b.add("심청이")
    b.print_list()
```

3) 메인 모듈과 하위 모듈

프롬프트 창이나 탐색기를 이용하여 실행하는 Python 모듈을 '최상위 수준'의 모듈이라고 하고 이 모듈이 메인 함수에 해당한다. Python에는 내장 전역 변수인 __name__이 있는데 이 변수는 모듈이 최상위 수준으로 실행될 때 '__main__'으로 지정된다.

최상위 수준으로 실행되는 모듈을 메인 모듈이라고 한다면 이 메인 모듈이 import 문을 이용하여 불러오는 모듈은 하위 모듈(Sub Module)이라고 한다. 하위 모듈의 __name__ 변수는 메인 모듈에서와는 달리 모듈 자체의 이름을 담고 있다. 아래 예제를 통해 확인해 보자.

[basic]-[module] sub.py

```
print('beginning of sub.py...')
print('name : {0}'.format(__name__))
print('end of sub.py....')
```

[basic]-[module]main.py

```
import sub

print('beginning of main.py...')
print('name :{0}'.format(__name__))
print('end of main.py...')
```

[basic]-[module] main.py 실행결과

```
beginning of sub.py...  import sub를 통해 'sub.py' 모듈의 코드가 실행되었다.
name : sub
end of sub.py....
beginning of main.py...  'main.py'의 코드가 실행되었다.
name :__main__
end of main.py...
```

'sub.py'의 코드를 최상위 수준일 때만 실행되도록 하고 싶을 때는 __name__변수가 '__main__'으로 지정되는 것을 이용한다. 'sub.py'의 출력문은 최상위 수준으로 실행할 때가 아니면 실행되지 않는다.

[basic]-[module] sub.py

```
#앞의 예제와 비교해서 달라진 점은 이 첫 번째 코드 한 줄 뿐이다.
if __name__ == '__main__':
    print('beginning of sub.py...')
    print('name : {0}'.format(__name__))
    print('end of sub.py....')
```

- main.py 모듈을 실행하면 sup.py의 출력문은 실행되지 않는다.

[basic]-[module] main.py 실행결과

```
beginning of main.py...
name :__main__
end of main.py...
```

- sub.py 모듈을 실행하면 __name__변수의 값이 '__main__'이 되어 출력문은 실행된다.

[basic]-[module] sub.py 실행결과

```
begining of sub.py...
name : __main__
end of sub.py....
```

4) 패키지

모듈의 수가 늘어나면 모듈의 이름이 충돌하거나 찾기 쉽게 디렉터리 별로 정리를 하는 것이 좋다. Python에서는 모듈을 모아놓은 디렉터리를 패키지(package)라고 한다. 평범한 디렉터리가 'Python의 패키지'로 인정받으려면 __init__.py 파일을 그 경로에 갖고 있어야 한다.

'__init__.py' 파일을 가지고 있는 디렉터리가 패키지이다. 이 파일은 보통 비워두지만 이 파일을 손대는 경우는 __all__이라는 변수를 조정할 때 정도이다. __all__은 아래와 같은 코드를 실행할 때 패키지로부터 반입할 모듈의 목록을 정의하기 위해 사용한다.

```
from 패키지 import *
```

- 패키지가 다음과 같이 구성되어 있다고 가정해보자

```
[basic]-[module] print_names.py
[basic]-[module]-[name_package]-kim.py
[basic]-[module]-[name_package]-lee.py
[basic]-[module]-[name_package]-park.py
[basic]-[module]-[name_package]-hong.py
```

먼저 [name_package] 디렉터리를 만들고 다음에 네 모듈(kim, lee, park, hong)의 코드를 작성한다. 각 파일의 내용이 동일하므로 하나만 작성한 다음 다른 파일에 복사/붙여넣기해도 좋다.

[basic]-[module]-[name_package] __init__.py

```
def print_name():
    print("name: {0}".format(__name__))
```

[basic]-[module]-[name_package] kim.py

```
def print_name():
    print("name: {0}".format(__name__))
```

[basic]-[module]-[name_package] lee.py

```
def print_name():
    print("name: {0}".format(__name__))
```

[basic]-[module]-[name_package] park.py

```
def print_name():
    print("name: {0}".format(__name__))
```

[basic]-[module]-[name_package] hong.py

```
def print_name():
    print("name: {0}".format(__name__))
```

import *가 kim, lee, park, hong 네 개의 모듈을 반입하도록 하려면 __init__.py에서 __all__를 아래와 같이 지정한다.

[basic]-[module]-[name_package] __init__.py

```
__all__ = ["kim", "lee", "park", "hong"]
```

[basic]-[module] print_names.py

```
from name_package import *  #import * 사용은 __all__ 변수를 확인하기 전까지는 어떤 모듈이 반입되는지 알 길이 없기 때문에 사용하지 않는 것이 좋다.
kim.print_name()
lee.print_name()
park.print_name()
hong.print_name()
```

- **파일에 데이터 읽고 쓰기**

[basic]-[file] write.py

```
with open("./file/test.txt", "w") as file:
    file.write("Hello")
```

[basic]-[file] read.py

```
with open("./file/test.txt", "r") as file:
    str = file.read()
    print(str)
```

[basic]-[file] writeLine.py

```
lines =[
    "그래도 지구는 돈다.: 갈릴레오\n",
    "내 사전에 불가능은 없다.: 나폴레옹\n",
    "너 자신을 알라.", "소크라테스\n"
]

with open("./file/famous.txt", "w", encoding="utf-8") as file:
    for line in lines:
        file.write(line)
```

[basic]-[file] readLine.py

```
with open("./file/famous.txt", "r", encoding="utf-8") as file:
    lines = file.readlines()
    line = ""
    for line in lines:
        print(line, end="")
```

[basic]-[file] writeAppend.py

```
with open("./file/names.txt", "a", encoding="utf-8") as file:
    while True:
        name = input("이름을 입력하세요(종료: 0)? ")
        if( name == "0"):
            break
        else:
            file.write(name + "\n")

print("프로그램을 종료합니다.")
```

[basic]-[file] readAppend.py

```
with open("./file/names.txt", "r", encoding="utf-8") as file:
    names = file.readlines()
    name = ""
    for name in names:
        print(name, end="")
```

- **파일관리 예제 프로그램**

[basic]-[file] address.py

```
import db

while True:
    print("------------------------------------------------------------------------")
    print("  1.주소입력 | 2.주소목록 | 3.주소검색 | 4.주소삭제 | 0.프로그램종료")
    print("------------------------------------------------------------------------")

    menu = input("메뉴선택> ")

    if(not menu.isnumeric()):
        print("숫자로 입력하세요!")
        break
    else:
        menu=int(menu)

    if(menu == 0):
        break
    elif(menu == 1):
        db.insert()
        continue
    elif(menu == 2):
        db.list()
        continue
    elif(menu == 3):
        db.read()
        continue
    elif(menu == 4):
        db.delete()
        continue
    else:
        print("0 ~ 4 숫자로 입력하세요!")

print("프로그램을 종료합니다!")
```

[basic]-[file] db.py ①

```
def insert():
    with open("address.txt", "a", encoding="utf-8") as file:
        name = input("이름> ")
        tel = input("전화> ")
        address = input("주소> ")
        file.write("{0},{1},{2}\n".format(name, tel, address))


def list():
    with open("address.txt", "r", encoding="utf-8") as file:
        lines = file.readlines()
        line = ""
        for line in lines:
            items = line.split(",")
            name = items[0]
            tel = items[1]
            address = items[2]
            print("{0}\t{1}\t{2}".format(name, tel, address), end="")
```

[basic]-[file] db.py ②

```
def read():
    with open("address.txt", "r", encoding="utf-8") as file:
        search = input("검색할 이름> ")
        lines = file.readlines()
        find = False

        for line in lines:
            items = line.split(",")
            if(search == items[0]):
                find = True
                name = items[0]
                tel = items[1]
                address = items[2]
                print("{0}\t{1}\t{2}".format(name, tel, address), end="")

        if(not find):
            print("{0} 이름은 존재하지 않습니다!".format(search))

def delete():
    with open("address.txt", "r", encoding="utf-8") as file:
        lines = file.readlines()
        search = input("삭제할 이름> ")
        find = False

        newines =""
        for line in lines:
            items = line.split(",")
            if(search == items[0]):
                find = True
            else:
                newines = newines + line

        sel = input("정말로 삭제하실래요(예:y)? ")

        if(sel == "y" or sel == "Y" or sel == "ㅛ"):
            with open("address.txt", "w", encoding="utf-8") as writeFile:
                writeFile.write(newines)
                print("{0}가(이) 삭제되었습니다!".format(search))

        if(not find):
            print("삭제할 이름이 존재하지 않습니다!")
```

[basic]-[file] address.txt

```
심청이,010-0002-0002,서울 강서구 화곡동
홍길동,010-0001-0001,인천 서구 청라 3동
성춘향,010-0008-0008,경기도 수원시 장안동
강감찬,010-0003-0003,인천 부평구 계산동
이순신,010-0004-0004,서울 강남구 압구정동
```

- **데이터베이스**

1) SQLite3 데이터베이스 설치하기

아래 주소로 접속해서 'Precompiled Binaries for Windows' 항목의 'SQLite-tools-win32-x86~.zip' 파일을 내려 받는다. 내려 받는 파일을 압축해제하면 SQLite3.exe 파일이 생긴다. 이 실행 파일을 더블클릭하면 셀이 실행되고 SQL문을 실습할 수 있다.

```
https://www.sqlite.org/download.html
```

2) 테이블 생성 및 데이터 입력

아래와 같이 'phonebook'이라는 이름의 테이블을 만들고 각 레코드를 식별할 수 있는 기본키(Primary key) 필드는 email로 지정한다.

name char(32)	phone char(32)	email char(64) primary key
홍길동	021-322-1542	hong@test.com
심청이	021-445-2424	shim@test.com
강감찬	026-542-7576	kang@test.com

- **SQLite 셀에 다음과 같이 입력한다. 마지막에 세미콜론(;)을 잊지 말고 입력해야한다.**

```
sqlite> .help
sqlite> .open phone.db
sqlite> create table phonebook(nae char(32), phone char(32), email char(64) primary key);
sqlite> .quit
```

- **SQLite 셀에 '.schema phonebook' 이라고 입력해서 phonebook 테이블의 스키마를 확인한다.**

```
sqlite> .open phone.db
sqlite> .schema phonebook
create table phonebook(nae char(32), phone char(32), email char(64) primary key);
sqlite> .quit
```

- **drop table 은 테이블을 삭제하고자 할 때 이용하는 구문이다. 매개변수는 삭제할 테이블 이름뿐이다.**

```
sqlite> .open phone.db
sqlite> drop table phonebook
sqlite> .schema phonebook
sqlite> .quit
```

- **앞에서 만들어준 phonebook 테이블에 insert문을 이용해서 데이터를 입력해 본다.**

```
sqlite> .open phone.db
sqlite> insert into phonebook(name, phone, email) values('홍길동', '021-322-1542', 'hong@test.com');
sqlite> insert into phonebook(name, phone, email) values('심청이', '021-445-2424', 'shim@test.com');
sqlite> insert into phonebook(name, phone, email) values('강감찬', '026-542-7576', 'kang@test.com');
sqlite> insert into phonebook(name, phone, email) values('이순신', '026-502-8586', 'lee@test.com');
sqlite> .quit
```

3) SQLite의 파이썬 API 사용

Python3에는 SQLite 라이브러리가 기본 탑재되어 있다. import문으로 sqlite3 모듈을 반입하면 SQLite API를 사용할 수 있다.

```
import sqlite3
```

- SQLite API를 사용할 때는 다음과 같은 과정을 거친다.

순서	작업
1	커넥션(Connection) 열기
2	커서(Cursor) 열기
3	커서를 이용하여 데이터 추가/조회/수정/삭제
4	커서 닫기
5	커넥션 닫기

- 데이터베이스 생성

[basic]-[database] create_table.py

```
import sqlite3

con = sqlite3.connect("phone.db")
cursor = con.cursor()

cursor.execute(
    "create table phonebook(name char(32), phone char(32), email char(64) primary key)"
)

cursor.close()
con.close()
```

- 샘플 데이터 입력, 수정 삭제

[basic]-[database] insert_record.py

```
import sqlite3

con = sqlite3.connect("phone.db")
cursor = con.cursor()

cursor.execute(
    "insert into phonebook(name, phone, email) values('홍길동', '021-322-1542', 'hong@test.com')"
)
id = cursor.lastrowid
print(id)

cursor.execute(
    "insert into phonebook(name, phone, email) values('심청이', '021-445-2424', 'shim@test.com');"
)
id = cursor.lastrowid
print(id)

cursor.execute(
    "insert into phonebook(name, phone, email) values('강감찬', '026-542-7576', 'kang@test.com')"
)
id = cursor.lastrowid
print(id)

con.commit()
cursor.close()
con.close()
```

[basic]-[database] delete_record.py

```
import sqlite3

con = sqlite3.connect("phone.db")
cursor = con.cursor()
cursor.execute(
    "select name, phone, email from phonebook"
)

rows = cursor.fetchall()
for row in rows:
    print("NAME: {0}, PHONE: {1}, EMAIL: {2}".format(row[0], row[1], row[2]))

cursor.close()
con.close()
```

[basic]-[database] update_record.py

```
import sqlite3

con = sqlite3.connect("phone.db")
cursor = con.cursor()
cursor.execute(
    "update phonebook set phone=?, name=? where email=?",
    ("010-0007-0007", "김길동", "hong@test.com",)
)

con.commit()
cursor.close()
con.close()
```

[basic]-[database] read_record.py

```
import sqlite3

con = sqlite3.connect("phone.db")
cursor = con.cursor()
cursor.execute(
    "select name, phone, email from phonebook where name=?",
    ("심청이",)
)

rows = cursor.fetchall()
for row in rows:
    print("NAME: {0}, PHONE: {1}, EMAIL: {2}".format(row[0], row[1], row[2]))

cursor.close()
con.close()
```

[basic]-[database] delete_table.py

```
import sqlite3

con = sqlite3.connect("phone.db")
cursor = con.cursor()
cursor.execute(
    "delete from phonebook where email=?",
    ("hong@test.com",)
)

con.commit()
cursor.close()
con.close()
```

- **데이터베이스 예제 프로그램**

[basic]-[database] db.py

```
import sqlite3

def connection():
    con = sqlite3.connect("phone.db")
    return con

def insert(phone):
    con = connection()
    coursor = con.cursor()
    coursor.execute(
        "insert into phonebook(name, phone, email) values(?, ?, ?)",
        (phone.name, phone.phone, phone.email)
    )

    con.commit()
    coursor.close()
    print("등록이 완료 되었습니다.")

def list():
    cursor = connection().cursor()
    cursor.execute(
        "select id, name, phone, email from phonebook"
    )
    rows = cursor.fetchall()
    return rows

def read(name):
    con = connection()
    cursor = con.cursor()
    cursor.execute(
        "select id, name, phone, email from phonebook where name like ?",
        ("%" + name +"%",)
    )
    rows = cursor.fetchall()
    return rows

def delete(id):
    con = connection()
    cursor= con.cursor()
    cursor.execute(
        "delete from phonebook where id=?",
        (id, )
    )
    con.commit()
    print("삭제가 완료되었습니다!")

def getPhone(id):
    con = connection()
    cursor = con.cursor()
    cursor.execute(
        "select id, name, phone, email from phonebook where id=?",
        (id, )
    )
    rows = cursor.fetchall()
    cursor.close()
    return rows

def update(phone):
    con = connection()
    cursor = con.cursor()
    cursor.execute(
        "update phonebook set name=?, phone=?, email=? where id=?",
        (phone.name, phone.phone, phone.email, phone.id)
    )
    con.commit()
    print("수정이 완료되었습니다!")
```

[basic]-[database] main.py ①

```
import db

class Phone:
    def __init__(self):
        self.id = 0
        self.name = "홍길동"
        self.phone = "010-0001-10101"
        self.email = "hong@test.com"

    def print_phone(self):
        print("ID: {0}, NAME: {1}, Phone: {2}, EMAIL: {3}".format(self.id, self.name, self.phone, self.email))

def main_menu():
    while True:
        print("------------------------------------------------------------------------")
        print("  1. 입력 | 2. 목록 | 3. 검색 | 4. 삭제 |  5.수정 | 0. 프로그램종료")
        print("------------------------------------------------------------------------")
        menu = input("메뉴선택> ")

        if(not menu.isnumeric()):
            print("숫자로 입력하세요!")
            continue
        else:
            menu=int(menu)

        if(menu == 0):
            print("프로그램을 종료합니다!")
            break

        elif(menu == 1):  #1. 입력
            try:
                phone = Phone()
                phone.name = input("이름> ")
                phone.phone = input("전화> ")
                phone.email = input("이메일> ")
                sel = input("등록하실래요(예:y)? ")
                if(sel == "y" or sel =="Y" or sel =="ㅛ"):
                    db.insert(phone)
            except Exception as err:
                print("에러가 발생했습니다.", err)

        elif(menu == 2):  #2. 목록
            try:
                rows = db.list()
                for row in rows:
                    phone = Phone()
                    phone.id = row[0]
                    phone.name = row[1]
                    phone.phone = row[2]
                    phone.email = row[3]
                    phone.print_phone()
            except Exception as err:
                print("에러가 발생했습니다.", err)

        elif(menu == 3):  #3. 검색
            try:
                name = input("검색할 이름> ")
                rows = db.read(name)
                for row in rows:
                    phone = Phone()
                    phone.id = row[0]
                    phone.name = row[1]
                    phone.phone = row[2]
                    phone.email = row[3]
                    phone.print_phone()
                if(len(rows) == 0):
                    print("{0} 이름은 존재하지 않습니다.".format(name))
            except Exception as err:
                print("에러가 발생했습니다!", err)
```

[basic]-[database] main.py ②

```
        elif(menu == 4):  #4. 삭제
            try:
                id = input("삭제할 ID> ")
                if(not id.isnumeric()):
                    print("ID는 숫자로 입력하세요!")
                    continue
                rows = db.getPhone(id)
                if(len(rows) == 0):
                    print("{0}번 데이터가 존재하지 않습니다!".format(id))
                else:
                    row = rows[0]
                    phone = Phone()
                    phone.id = row[0]
                    phone.name = row[1]
                    phone.phone = row[2]
                    phone.email = row[3]
                    phone.print_phone()
                    sel = input("삭제하실래요(예:y)? ")
                    if(sel == "y" or sel =="Y" or sel =="ㅛ"):
                        db.delete(id)
            except Exception as err:
                print("에러가 발생했습니다.", err)

        elif(menu == 5):  #5. 수정
            try:
                id = input("수정할 ID> ")
                if(not id.isnumeric()):
                    print("ID는 숫자로 입력하세요!")
                    continue

                rows = db.getPhone(id)
                if(len(rows) == 0):
                    print("{0}번 데이터가 존재하지 않습니다!".format(id))
                else:
                    row = rows[0]
                    phone = Phone()
                    phone.id = row[0]
                    phone.name = row[1]
                    phone.phone = row[2]
                    phone.email = row[3]

                    sname = input("이름:{0}> ".format(phone.name))
                    if(sname != ""):
                        phone.name = sname
                    sphone = input("전화:{0}> ".format(phone.phone))
                    if(sphone != ""):
                        phone.phone = sphone
                    semail = input("이메일:{0}> ".format(phone.email))
                    if(semail != ""):
                        phone.email = semail

                    phone.print_phone()
                    sel = input("수정하실래요(예:y)? ")
                    if(sel == "y" or sel =="Y" or sel =="ㅛ"):
                        db.update(phone)
            except Exception as err:
                print("에러가 발생했습니다.", err)

        else:  #1 ~ 5가 아닌 경우
            print("0 ~ 4 숫자로 입력하세요!")

if(__name__ == '__main__'):
    main_menu()
```