Python 프로그래밍

(기본문법)

01. Python 데이터처리

- 아래 사이트에서 python 프로그램을 다운 받아 설치한다.

```
https://www.python.org/downloads/
```

- 아래 사이트에서 Visual Studio Code 프로그램을 다운받아 설치한다.

```
https://code.visualstudio.com/download
```

- vs-code를 실행한 후 [c:]-[data]-[python]-[기본문법] 폴더를 생성한다.
- Extension 메뉴에서 python을 검색한 후 첫 번째 확장 프로그램을 설치한다. (버전 확인:python --version)
- vs-code [설정]-[Settings] zoom을 검색한 후 zoom메뉴를 체크해 준다.

• Python 데이터타입

Python에서는 크게 숫자 데이터, 문자 데이터, 불린 데이터로 분류된다. 숫자인 경우 정수형 데이터와 실수형 데이터로 분류된다.

[01.데이터처리] ex01.py

```
#숫자 데이터
print(5, type(5))
print(-10)
print(3.14, type(3.14))

#문자 데이터
print('파이썬', type('파이썬'))
print('==========')
print('=' * 10)

#불린 데이터
print(True, type(True))
print(False)
```

• Python 변수

프로그래밍에서 말하는 변수(Variable)의 뜻은 "하나의 값을 저장할 수 있는 저장 공간"이라고 할 수 있다.

[01.데이터처리] ex02.py

```
#숫자 변수
int1=-5
int2=-10
float1=3.14

#변수에 수식 저장도 가능
sum = int1 + int2
avg = sum / 2
print(int1, int2, float1, sum, avg)

#문자 변수
str1='파이썬'
str2='=============='
print(str1, str2)

#불린 변수
bool1 = True
bool2 = False
print(bool1, bool2)
```

- 변수를 사용하지 않은 경우

[01.데이터처리] ex03.py

```
print('저는 홍길동입니다.')
print('나이는 19살입니다.')
print('취미는 컴퓨터게임입니다.')
print('저는 성인일까요? False')
```

- 변수를 사용 한 경우

[01.데이터처리] ex04.py

```
name='홍길동'
age=19
hooby='컴퓨터게임'
isAdult=age >= 20

print('저는 '+ name + '입니다.')
#age, isAdult는 정수형, 불린형 변수이므로 문자열로 변환 후 연결
print('나이는 '+ str(age) + '살입니다.')
print('취미는 '+ hooby + '입니다.')
print('저는 성인일까요? '+ str(isAdult))
```

- 콤마를 사용하여 출력하는 경우

[01.데이터처리] ex05.py

```
name='홍길동'
age=19
hooby='컴퓨터게임'
isAdult=age >= 20

print('저는', name, '입니다.')
#콤마를 사용할 경우 문자열 변환 불필요
print('나이는', age, '입니다.')
print('취미는', hooby, '입니다.')
print('저는 성인일까요?', isAdult)
```

• Python 언어에서 주석

1) 한줄 주석은 #, 여러 줄 주석은 작은따옴표 세 개(''')로 시작해서 작음따옴표 세 개(''')로 끝난다.
2) 단축키 사용 시에는 블록을 지정한 후 ctrl + '/'를 사용한다. 해지할 경우에도 불럭 지정 후 ctrl + '/'를 사용한다.

- 아래 Quiz 결과가 출력되도록 프로그램을 작성하시오.

Quiz : 변수를 이용하여 다음 문장을 출력하시오.

변수 명: station
변수 값: "사당", "신도림", "인천공항" 순서대로 입력
출력문장: xx행 열차가 들어오고 있습니다.

[01.데이터처리] ex06.py

```
station='사당'
print(station + '행 열차가 들어오고 있습니다.')
station='신도림'
print(station, '행 열차가 들어오고 있습니다.')
station ='인천공항'
print(f'{station}행 열차가 들어오고 있습니다.')
```

- **Python의 연산자와 함수**

Python 연산자에는 산술연산자, 관계연산자, 논리연산자와 다양한 함수들이 있다.

[01.데이터처리] ex07.py

```
#산술연산자
print('5+2=', 5+2)
print('5-2=', 5-2)
print('5*2=', 5*2)
print('5/2=', 5/2)
print('5**2=', 5**2)
print('5%2=', 5%2)
print('5//2=', 5//2)

#관계연산자
print('5>3', 5>3)
print('5>=2', 5>=2)
print('5<2', 5<2)
print('5<=2', 5<=2)
print('5==2', 5==2)
print('5!=2', 5!=2)

#논리연산자
print('(5>2) and (5>10)', (5>2) and (5>10))
print('(5>2) & (5>10)', (5>2) & (5>10))
print('(5>2) or (5>10)', (5>2) or (5>2))
print('(5>2) | (5>10)', (5>2) | (5>2))
print('not(5>2)', not(5>2))
print('5>4 >3', 5>4>3)
print('(5>4) and (5>3)', (5>4) and (5>3))
```

- **Python 연산자를 이용한 간단한 수식**

[01.데이터처리] ex08.py

```
num=1
print('num', num)

num=num+2
print('num=num+2', num)

num+=2  #num = num + 2
print('num+=2', num)

num*=2  #num = num * 2
print('num*=2', num)

num/=2  #num = num / 2
print('num/=2', num)

num-=2  #num = num - 2
print('num-=2', num)

num%=5  #num = num % 5
print('num%=5', num)

num//=2  #num = num // 2
print('num//=2', num)
```

- **Python 숫자 처리를 위한 기본함수**

[01.데이터처리] ex09.py

```
print('abs(-5)=', abs(-5))  #절댓값
print('pow(4, 2)=', pow(4, 2)) #4의 2승
print('max(5, 12, 4)=', max(5, 12, 4))  #최솟값
print('round(3.14)=', round(3.14))  #반올림
print('round(3.14159, 3)=', round(3.14159, 3)) #반올림하여 소수점 3째 자리까지 출력

from math import *

print('floor(4.99)=', floor(4.99))  #소수점 버림
print('ceil(3.14)=', ceil(3.14))  #소수점 올림
print('sqrt(16)=', sqrt(16))  #제곱근
```

- **Python 숫자 처리를 위한Random함수**

[01.데이터처리] ex10.py

```
from random import *

#0.0이상 ~ 1.0미만의 임의의 값 생성
print('random()=', random())

#0.0이상 ~ 10미만의 임의의 값 생성
print('random()*10=', random()*10)

#0이상 ~ 10미만의 임의의 정수 생성
print('int(randowm()*10)=', int(random()*10))

#1이상 ~ 10이하의 임의의 정수 생성
print('int(random()*10)+1=', int(random()*10)+1)

#1이상 ~ 46미만의 임의의 정수 생성 (46 미포함)
print('randrange(1, 46)=', randrange(1, 46))

#1이상 ~ 46이하의 임의의 정수 생성 (46 포함)
print('randint(1, 46)=', randint(1, 46))
```

Quiz

당신은 최근에 코딩 스터디 모임을 새로 만들었습니다.
월 4회 스터디를 하는데 3번은 온라인으로 하고 1번은 오프라인으로 하기로 했습니다.
아래 조건에 맞는 오프라인 모임 날짜를 정해주는 프로그램을 작성하시오.

조건1: 랜덤으로 날짜를 뽑아야 함
조건2: 월별 날짜는 다름을 감안하여 최소 일수인 28 이내로 정함
조건3: 매월 1~3일은 스터디 준비를 해야 하므로 제외

(출력문 예제)
오프라인 스터디 모임 날짜는 매월 x일로 선정되었습니다.

[01.데이터처리] ex11.py

```
from random import *

date = randint(4, 28)
print('오프라인 스터디 모임 날짜는 매월 '+ str(date) + ' 일로 선정되었습니다.')
```

- **Python의 문자열**

문자열(string)이란 연속된 문자들의 나열을 의미한다. 따옴표 또는 작은따옴표로 둘러싸여 있으면 모두 문자열이라고 보면 된다.

[01.데이터처리] ex12.py

```
sentence='나는 소년입니다.'
print(sentence)
sentence="파이썬은 쉬워요."
print(sentence)
sentence='''
나는 소년이고
파이썬은 쉬워요.
'''
print(sentence)
```

- **문자열 슬라이싱(slicing)**

[01.데이터처리] ex13.py

```
jumin='990120-2155011'

print('성별:', jumin[7])  #인덱스 7 한문자 출력

print('년:', jumin[0:2])  #인덱스 0부터 인덱스 2직전까지 출력

print('월:', jumin[2:4])  #인덱스 2부터 인덱스 4직전까지 출력

print('일:', jumin[4:6])  #인덱스 4부터 인덱스 6직전까지 출력

print('생년월일:', jumin[:6])  #인덱스 0부터 6직전까지 출력

print('뒤 7자리:', jumin[7:])  #인덱스 7부터 끝까지 출력

print('뒤 7자리부터 끝까지 출력', jumin[-7:])  #맨 뒤에서 7번째부터 끝까지 출력
```

- **문자열 처리함수**

[01.데이터처리] ex14.py

```
python='Python is Amazing'

print(python.lower())  #python 변수의 값을 모두 소문자로 출력

print(python.upper())  #python 변수의 값을 모두 대문자로 출력

print(python[0].isupper())  #인덱스 0의 값이 소문자이면 True 아니면 False 출력

print(len(python))  #문자열의 길이 출력

print(python.replace('Python', '파이썬'))  #Python을 파이썬으로 대체한다.

index = python.index('i')  #처음에 나오는 'i'의 인덱스 값을 출력하고 'i'가 없으면 오류 발생
print(index)
print(python.index('i', index+1))  #index + 1번째 이후에 찾은 'i'의 인덱스 값을 출력

print(python.find('n'))  #'n'을 찾은 경우에 index 함수와 같은 결과가 출력
print(python.find('java'))  #'java'가 없으면 -1이 출력

print(python.count('i'))  #'i'가 몇 개인지 출력
```

- **문자열 출력포맷**

[01.데이터처리] ex15.py

```
#방법1
print('a''b''c')
print('a', 'b', 'c')

#방법2
print('나는 %d살 입니다.'%20)
print('나는 %s을 좋아해요.'%'python')
print('Apple은 %c로 시작해요.'%'A')

#%s는 정수, 한문자 모두 가능하다.
print('나는 %s살 입니다.'%20)
print('Apple은 %s로 시작해요.'%'A')
print('나는 %s색과 %s색을 좋아해요.'%('파랑', '빨강'))

#방법3
print('나는 {}살입니다.'.format(20))
print('나는 {}색과 {}색을 좋아해요.'.format('파랑', '빨강'))
print('나는 {0}색과 {0}색을 좋아해요.'.format('파랑', '빨강'))
print('나는 {1}색과 {0}색을 좋아해요.'.format('파랑', '빨강'))

#방법4
print('나는 {age}살이며, {color}색을 좋아해요.'.format(age=20, color='빨강'))
print('나는 {age}살이며, {color}색을 좋아해요.'.format(color='빨강', age=20))

#방법5 (v3.6이상 사용가능)
age=20
color='빨강'
print(f'나는 {age}살이며, {color}색을 좋아해요.')
pi = 3.14159265359
print(f'{pi:.2f}')
pay = 100000
print(f'{pay:,}')
```

- **탈출(escape)문자 : 화면에 출력하는 문자가 아니라 문자 출력을 제어하는 문자이다.**

[01.데이터처리] ex16.py

```
#문장 내에서 줄 바꿈
print('백문이 불여일견\n백견이 불여일타')

#문장 내에서 작은따옴표 또는 큰 따옴표를 출력
print('저는 "홍길동"입니다.')
print("저는 '홍길동'입니다.")
print('저는 \'홍길동\'입니다.')

#문장 내에서 '\'를 출력
print('c:\\data\\python')

#커서를 맨 앞으로 이동 후 출력
print('Red Apple\rPine')

#키보드 백스페이스 누른 효과(한 글자 삭제)
print('Redd\b Apple')

#키보드 탭(tab)키를 누른 효과
print('Red\tApple')
```

Quiz

사이트별로 비밀번호를 만들어 주는 프로그램을 작성하시오.

예) http://naver.com

규칙1: http://부분은 제외 => naver.com

규칙2: 처음 만나는 점(.) 이후 부분을 제외 => naver

규칙3: 남은 글자 중 처음 세 자리 + 글자 개수 + 글자 내 "e" 개수 + "!"로 구성

결과) 생성된 비밀번호 : nav51!

[01.데이터처리] ex17.py

```
url1='http://naver.com'
url2='http://doum.com'
url3='http://google.com'
url4='http://youtube.com'

#규칙1
mystr=url1.replace('http://', '')
print(mystr)

#규칙2
mystr=mystr[:mystr.index('.')]
print(mystr)

#규칙3
password =mystr[:3]
password+=str(len(mystr))
password+=str(mystr.count('e'))
password+='!'
print('{0}의 비밀번호는 {1}입니다.'.format(url1, password))
print(f'{url1}의 비밀번호는 {password}입니다.')
```

- **Python의 리스트(list)**

리스트(list)란 원소들이 연속적으로 저장되는 형태의 자료형이다. 저장되는 요소들이 모두 같은 자료형일 필요는 없다. 대괄호:[]를 사용한다.

[01.데이터처리] ex18.py ①

```
subway = [10, 20, 30] #지하철 칸별로 10명, 20명, 30명
print(subway)

subway = ['유재석', '조세호', '박명수']
print(subway)

print(subway.index('조세호')) #조세호가 몇 번째 칸에 타고 있는가?

subway.append('하하') #'하하'가 다른 정류장에서 마지막 칸에 탄 경우
print(subway)

subway.insert(1, '정형돈') #'정형돈'을 '유재석'과 '조세호' 사이 칸에 태움
print(subway)

subway.pop() #지하철에 있는 사람을 뒤에서 한명 꺼냄
print(subway)

subway.append('유재석')
print(subway)
print(subway.count('유재석')) #같은 이름의 사람이 몇 명이 있는지 확인
```

[01.데이터처리] ex18.py ②

```
nums=[5, 2, 4, 3, 1]
nums.sort()
print(nums)

nums.reverse() #순서 뒤집기
print(nums)

nums.clear() #모두지우기
print(nums)

mix_list = ['조세호', 20, True] #다양한 자료형 함께 사용
print(mix_list)

nums=[5, 2, 4, 3, 1] #리스트 확장
mix_list.extend(nums)
print(mix_list)
```

- **Python의 딕션너리(dictionary)**

Python 딕셔너리(dictionary)는 각 키가 고유해야 하는 키-값 쌍의 변경 가능하고 정렬되지 않은 자료형이다. 중괄호:{}를 사용한다.

[01.데이터처리] ex19.py

```
cabinet = {3: '유재석', 100: '김태호'}
print(cabinet[100]);
#print(cabinet[5]) 키가 없는 경우 오류발생 후 프로그램 종료

#get 함수를 이용한 값 추출
print(cabinet.get(3))
print(cabinet.get(5, '값없음'))  #키가 없는 경우 Default로 'None' 출력

#in을 사용하여 키가 있는지 유무(True, False) 체크기능
print(3 in cabinet)
print(5 in cabinet)

#키 이름을 문자로도 가능
cabinet={'A': '유재석', 'B':'김태호'}
print(cabinet['A'])

#새로운 키에 새로운 데이터 할당
cabinet['C'] = '조세호'
print(cabinet)

#기존키에 새로운 데이터 할당
cabinet['A'] = '김종국'
print(cabinet)

#키 삭제
del cabinet['A']
print(cabinet)

#dictionary에 키와 값을 따로 출력
print(cabinet.keys(), type(cabinet.keys()))
print(cabinet.values(), type(cabinet.values()))

#키 값을 쌍으로 출력
print(cabinet.items(), type(cabinet.items()))

#모든 데이터 삭제
cabinet.clear()
print(cabinet)
```

- **Python의 튜플(tuple)**

튜플(tuple)은 몇 가지 점을 제외하곤 리스트와 거의 비슷한 자료형이며 리스트와 다른 점은 다음과 같다.

- **리스트는 [], 튜플(tuple)은 ()으로 둘러싼다.**
- **리스트는 요소 값의 생성, 삭제, 수정이 가능하지만, 튜플(tuple)은 요소 값을 바꿀 수 없다.**
- **사용 빈도가 적은반면 데이터 처리 속도가 빠르다.**

[01.데이터처리] ex20.py

```
menu=('돈까스', '치즈까스')
print(menu[0])
print(menu[1])
print(menu)
print(type(menu))
#print(menu[3]) 없는 index를 지정하면 오류가 발생

name = '김종국'
age = 20
hobby = '코딩'
print(name, age, hobby)

#튜플(tuple)을 이용하여 값 선언
(name, age, hobby) = ('김종국', 20, '코딩')
print(name, age, hobby)
```

- **Python의 집합세트(set)**

세트(set)는 집합에 관련된 것을 쉽게 처리하기 위해 만든 자료형이다. 집합 자료형은 중괄호{ }또는 set 키워드를 사용해 만들 수 있다.

[01.데이터처리] ex21.py

```
java = {'유재석', '김태호', '양세형'}
print(1, java, type(java))

python = set(['유재석', '박명수'])
print(2, python, type(python))

#교집합
print(java & python)
print(3, java.intersection(python))

#합집합
print(java | python)
print(4, java.union(python))

#차집합
print(java - python)
print(5, java.difference(python))

#python을 할 줄 아는 사람이 늘어남
python.add('김태호')
print(6, python)

#java 언어를 잊어버린 경우
java.remove('김태호')
print(7, java)

#모든 학생들이 언어를 잊어버린 경우
python.clear()
java.clear()
print(8, python, java)
```

- **자료구조의 변경**

[01.데이터처리] ex22.py

```
menu = {'커피', '우유', '주스'}
print(1, menu, type(menu))

#menu를 list 타입으로 변환한다.
menu = list(menu)
print(2, menu, type(menu))

#menu를 tuple 타입으로 변환한다.
menu = tuple(menu)
print(3, menu, type(menu))
#menu를 set 타입으로 변환한다.
menu = set(menu)
print(4, menu, type(menu))

#내가 좋아하는 음식들
foods = '짜장면 피자 돈까지 만두 탕수육'
print(5, foods, type(foods))

#foods 문자열 변수의 값을 빈칸으로 분리해 리스트 타입으로 변환한다.
foods = foods.split(' ')
print(6, foods, type(foods))

#foods 리스트 변수의 값들을 콤마로 연결해 문자열 타입으로 변환한다.
foods = ','.join(foods)
print(7, foods, type(foods))
print(f'내가 좋아하는 음식은 {foods}입니다.')

#names 변수 값들 중 이씨 성을 가진 이름만 출력하시오.
names = '홍길동,이몽룡,강감찬,이순신,성춘향'
print(8, names, type(names))

#names 문자열 변수의 값을 콤마로 분리해 리스트 타입으로 변환한다.
names = names.split(',')
print(9, names, type(names))

new_names=[]
for name in names:
    if name.startswith('이'):
        new_names.append(name)

#names 리스트 변수의 값들을 ' '로 연결해 문자열 타입으로 변환한다.
names = ','.join(new_names)
print(10, names, type(names))
print(f'이씨 성을 가진 이름들은 {names}입니다.')
```

[01.데이터처리] ex22.py 실행결과

```
1 {'커피', '주스', '우유'} <class 'set'>
2 ['커피', '주스', '우유'] <class 'list'>
3 ('커피', '주스', '우유') <class 'tuple'>
4 {'커피', '주스', '우유'} <class 'set'>
5 짜장면 피자 돈까지 만두 탕수육 <class 'str'>
6 ['짜장면', '피자', '돈까지', '만두', '탕수육'] <class 'list'>
7 짜장면,피자,돈까지,만두,탕수육 <class 'str'>
내가 좋아하는 음식은 짜장면,피자,돈까지,만두,탕수육입니다.
8 홍길동,이몽룡,강감찬,이순신,성춘향 <class 'str'>
9 ['홍길동', '이몽룡', '강감찬', '이순신', '성춘향'] <class 'list'>
10 이몽룡,이순신 <class 'str'>
이씨 성을 가진 이름들은 이몽룡,이순신입니다.
```

- **아래 조건을 만족하는 프로그램을 작성하시오.**

Quiz

당신의 학교에서는 파이썬 코딩 대회를 주최합니다.
참석률을 높이기 위해 댓글 이벤트를 진행하기로 하였습니다.
댓글 작성자들 중에 추첨을 통해 1명은 치킨, 3명은 커피 쿠폰을 받게 됩니다.
추첨 프로그램을 작성하세요.

(조건)
조건1: range 함수를 이용해 댓글 아이디는 1~20까지 생성
조건2: 댓글 내용과 상관없이 무작위로 추첨하되 중복불가
조건3: random 모듈의 shuffle 과 sample을 활용

(출력예제)
---당첨자 발표---
치킨 당첨자: 1
커피 당첨자: [2, 3, 4]
---축하합니다.---

(활용예제)
```
from random import *
lst=[1,2,3,4,5]
print(lst)
shuffle(lst)
print(sample(lst, 1))
```

[01.데이터처리] ex23.py

```
from random import *

#1부터 20까지의 숫자를 생성
users = range(1, 21)
print(users, type(users))

#list 타입으로 변환
users=list(users)
print(users, type(users))

#주어진 배열의 원소를 무작위로 섞는다.
shuffle(users)
print(users)

#지정한 숫자만큼의 요소들을 Random으로 뽑아 리스트로 반환
winner = sample(users, 4)
print(winner)

print('---당첨자 발표---')
print('치킨 당첨자: {0}'.format(winner[0]))
print('커피 당첨자: {0}'.format(winner[1:]))
print('---축하합니다.---')
```

[01.데이터처리] ex23.py 실행결과

```
---당첨자 발표---
치킨 당첨자: 5
커피 당첨자: [1, 15, 18]
---축하합니다.---
```

---

02. 프로그램의 흐름 제어하기

- **IF 제어문**

[02.흐름제어] ex01.py

```
#input 함수는 키보드에서 입력받는다. 입력한 데이터는 문자형이다.
num1 = int(input('숫자를 입력하세요?'))
num2 = int(input('숫자를 입력하세요?'))

if num1 > num2:
    print(f'{num1}은 {num2}보다 큽니다.')
else:
    print(f'{num1}은 {num2}보다 작습니다.')
```

[02.흐름제어] ex02.py

```
temp = int(input('기온은 어때요?'))

if temp >= 30:
    print('너무 더워요. 나가지 마세요.')
elif temp >= 10 and temp < 30:
    print('괜찮은 날씨예요.')
elift temp >= 0 and temp < 10:
    print('외투를 챙기세요.')
else:
    print('너무 추워요. 나가지 마세요.')
```

[02.흐름제어] ex03.py

```
score = int(input('점수를 입력하세요?'))
grade = ''

if score>=90:
    if score>=95:
        grade = 'A+'
    else:
        grade = 'A0'
elif score>=80 and score<90:
    if score>=85:
        grade = 'B+'
    else:
        grade = 'B0'
elif score>=70 and score<80:
    if score>=75:
        grade = 'C+'
    else:
        grade = 'C0'
elif score>=60 and score<70:
    if score>=65:
        grade = 'D+'
    else:
        grade = 'D0'
else:
    grade = 'F'

print(f'점수 {score}의 학점은 {grade}입니다.')
```

- **FOR 반복문**

[02.흐름제어] ex04.py

```
for i in range(10):
    print(i)

for i in range(1, 10, 2):
    print(i)

for i in range(10, 0, -1):
    print(i)
```

[02.흐름제어] ex05.py

```
sum = 0
for i in range(1, 101):
    sum += i
print(f'1~100까지의 합은 {sum}')

sum = 0
for i in range(0, 101, 2):
    sum +=i
print(f'1~100까지의 짝수의 합은 {sum}')

sum = 0
for i in range(1, 101, 2):
    sum +=i
print(f'1~100까지의 홀수의 합은 {sum}')
```

[02.흐름제어] ex05.py 실행결과

```
1~100까지의 합은 5050
1~100까지의 짝수의 합은 2550
1~100까지의 홀수의 합은 2500
```

[02.흐름제어] ex06.py

```
starbucks=['유재석', '하하', '정형돈', '김종국']
for customer in starbucks:
    print('{0}, 커피가 준비되었습니다.'.format(customer))

starbucks=('유재석', '하하', '정형돈', '김종국')
for customer in starbucks:
    print('{0}, 커피가 준비되었습니다.'.format(customer))

starbucks={'유재석', '하하', '정형돈', '김종국'}
for customer in starbucks:
    print('{0}, 커피가 준비되었습니다.'.format(customer))

starbucks={'1번':'유재석', '2번':'하하', '3번':'정형돈', '4번': '김종국'}
for no, customer in starbucks.items():
    print(f'{no}{customer}님 커피가 준비되었습니다.')

starbucks=['유재석', '하하', '정형돈', '김종국']
for index in range(len(starbucks)):
    print(f'{index + 1}번 {starbucks[index]}님 커피가 준비되었습니다.')

starbucks=['유재석', '하하', '정형돈', '김종국']
for index, customer in enumerate(starbucks):
    print(f'{index+1}번 {customer}님 커피가 준비되었습니다.')
    print(f'대기 중인 손님은 {len(starbucks)-(index+1)}명입니다.')
```

[02.흐름제어] ex07.py

```
students=[
    {"no":"1번", "이름":"홍길동", "국어":90, "영어":95, "수학": 98},
    {"no":"2번", "이름":"강감찬", "국어":80, "영어":99, "수학": 96},
    {"no":"3번", "이름":"이순신", "국어":92, "영어":85, "수학": 78},
    {"no":"4번", "이름":"성춘향", "국어":78, "영어":55, "수학": 67},
    {"no":"5번", "이름":"이몽룡", "국어":95, "영어":85, "수학": 87},
]

for student in students:
    tot = student.get('국어') + student.get('영어') + student.get('수학')
    avg = tot / 3
    name = student.get('이름')
    print(f'{name}의 평균점수는 {avg:.2f}입니다.')
```

[02.흐름제어] ex07.py 실행결과

```
홍길동의 평균점수는 94.33입니다.
강감찬의 평균점수는 91.67입니다.
이순신의 평균점수는 85.00입니다.
성춘향의 평균점수는 66.67입니다.
이몽룡의 평균점수는 89.00입니다.
```

[02.흐름제어] ex08.py

```
products = [
    {"no":"1번", "name":"비스코프 냉장고", "price":3500000, "qnt": 5},
    {"no":"2번", "name":"오브제 스타일러", "price":2230000, "qnt": 4},
    {"no":"3번", "name":"비스코프 세탁기", "price":2400000, "qnt": 6},
    {"no":"4번", "name":"전자레인지", "price":150000, "qnt": 8},
    {"no":"5번", "name":"오브제 텔레비젼", "price":350000, "qnt": 7},
]

for product in products:
    sum = product.get('price') * product.get('qnt')
    no = product.get('no')
    name = product.get('name')
    price = product.get('price')
    qnt = product.get('qnt')

    print(f'번호: {no:<5}\t', end='')
    print(f'이름: {name:<10}\t', end='')
    print(f'단가: {price:>9,}원\t', end='')
    print(f'수량: {qnt:>4,}개\t', end='')
    print(f'금액: {sum:>15,}원', end='')
    print()
    print('-' * 110)
```

[02.흐름제어] ex08.py 실행결과

```
번호: 1번       이름: 비스코프 냉장고   단가: 3,500,000원       수량:    5개    금액:      17,500,000원
--------------------------------------------------------------------------------------------------------------
번호: 2번       이름: 오브제 스타일러   단가: 2,230,000원       수량:    4개    금액:       8,920,000원
--------------------------------------------------------------------------------------------------------------
번호: 3번       이름: 비스코프 세탁기   단가: 2,400,000원       수량:    6개    금액:      14,400,000원
--------------------------------------------------------------------------------------------------------------
번호: 4번       이름: 전자레인지        단가:   150,000원       수량:    8개    금액:       1,200,000원
--------------------------------------------------------------------------------------------------------------
번호: 5번       이름: 오브제 텔레비젼   단가:   350,000원       수량:    7개    금액:       2,450,000원
--------------------------------------------------------------------------------------------------------------
```

- **1~10번 학생들을 반복하면서 결석한 학생은 스킵하고 출석한 학생은 책을 읽히고 책이 없으면 수업을 종료한다.**

[02.흐름제어] ex09.py

```
absent = [2, 5]  #결석 학생 목록
no_book = [7]  #책이 없는 학생 목록

#1번부터 10번까지 반복
for student in range(1, 11):
    if student in absent:
        continue
    elif student in no_book:
        print(f'오늘 수업 여기까지, {student}번은 교무실로 따라와.')
        break
    print(f'{student}번, 책을 읽어봐.')
```

Quiz

당신은 Cocoa 서비스를 이용하는 택시 기사님입니다.
50명의 승객과 매칭 기회가 있을 때, 총 탑승 승객 수를 구하는 프로그램을 작성하시오.

조건1 : 승객별 운행 소유 시간은 5분 ~ 50분 사이의 난수로 정해집니다.
조건2 : 당신은 소요시간 5분 ~ 15분 사이의 승객만 매칭 해야 합니다.

(출력문 예제)
[O] 1번째 손님 (소요시간 : 15분)
[X] 2번째 손님 (소요시간 : 50분)
[O] 3번째 손님 (소요시간 : 5분)
...
[O] 48번째 손님 (소요시간 : 4분)
[X] 49번째 손님 (소요시간 : 18분)
[X] 50번째 손님 (소요시간 : 16분)

총 탑승 승객 : 2명

[02.흐름제어] ex10.py

```
from random import*

#탑승한 승객수를 누적할 변수
cnt = 0

#1번 승객부터 50번 승객까지 반복
for i in range(1, 51):
    #5분부터 50분까지 random 소요시간 발생
    time = randrange(5, 51)

    #소요 시간이 5분에서 15분 이내이면 탑승 승객 수 증가 처리
    if 5<= time <= 15:
        print(f'[O] {i}번째 손님 (소요시간: {time}분)')
        cnt += 1
    else:
        print(f'[X] {i}번째 손님 (소요시간: {time}분)')

print(f'총 탑승 승객: {cnt}')
```

- **다중 FOR문**

▪ **다중 for문을 사용하여 아래와 같이 출력하도록 프로그램을 작성하시오.**

[02.흐름제어] ex11.py

```
for i in range(5):  #i=0~4
    for j in range(1, 11):  #(i=0, j=1,11), (i=1, j=1,11), (i=2, j=1,11), (i=3, j=1,11), (i=4, j=1,10)
        print(j, end=' ')  #j값 출력 후 한 칸(end=' ') 띄우고 줄 바꿈 하지 않는다.
    print()  #줄 바꿈 한다.
```

[02.흐름제어] ex11.py 실행결과

```
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
```

▪ **다중 for문을 사용하여 아래와 같이 출력하도록 프로그램을 작성하시오.**

[02.흐름제어] ex12.py

```
for i in range(5):  #i=0~4
    for j in range(0, i+1): #(i=0, j=0,1), (i=1, j=0,2), (i=2, j=0,3), (i=3, j=0,4), (i=4, j=0,5)
        print('*', end='')  #'*'를 출력하고 줄 바꿈 하지 않는다.
    print()
```

[02.흐름제어] ex12.py 실행결과

```
*
**
***
****
*****
```

▪ **다중 for문을 사용하여 아래와 같이 출력하도록 프로그램을 작성하시오.**

[02.흐름제어] ex13.py

```
for i in range(1, 6):  #i=1~5
    for j in range(6-i, 0, -1):  #(i=1, j=5,0), (i=2, j=4,0), (i=3, j=3,0), (i=4, j=2,0), (i=5, j=1,0)
        print('*', end='')  #'*'를 출력하고 줄 바꿈 하지 않는다.
    print()
```

[02.흐름제어] ex13.py 실행결과

```
*****
****
***
**
*
```

- **다중 for문을 사용하여 아래와 같이 출력하도록 프로그램을 작성하시오.**

[02.흐름제어] ex14.py

```
for i in range(1, 6):  #i=1~5
    for j in range(5-i):  #(i=1, j=0,4), (i=2, j=0,3), (i=3, j=0,2), (i=4, j=0,1), (i=5, j=0,0)
        print(' ', end='')  #한 칸 띄우고 줄 바꿈 하지 않는다.
    for j in range(i):  #(i=1, j=0,1), (i=2, j=0,2), (i=3, j=0,3), (i=4, j=0,4), (i=5, j=0,5)
        print('*', end='')  #'*'를 출력하고 줄 바꿈 하지 않는다.
    print()
```

[02.흐름제어] ex14.py 실행결과

```
    *
   **
  ***
 ****
*****
```

- **다중 for문을 사용하여 아래와 같이 출력하도록 프로그램을 작성하시오.**

[02.흐름제어] ex15.py

```
for i in range(5):  #i=0~4
    for j in range(i):  #(i=0, j=0,0), (i=1, j=0,1), (i=2, j=0,2), (i=3, j=0,3), (i=4, j=0,4)
        print(' ', end='')  #한 칸 띄우고 줄 바꿈 하지 않는다.
    for j in range(5-i):  #(i=0, j=0,4), (i=1, j=0,3), (i=2, j=0,2), (i=3, j=0,1), (i=4, j=0,0)
        print('*', end='')  #'*'를 출력하고 줄 바꿈 하지 않는다.
    print()
```

[02.흐름제어] ex15.py 실행결과

```
*****
 ****
  ***
   **
    *
```

- **다중 for문을 사용하여 아래와 같이 출력하도록 프로그램을 작성하시오.**

[02.흐름제어] ex16.py

```
for i in range(1, 6):  #i=1~5
    for j in range(5-i, 0, -1):  #(i=1, j=4,0), (i=2, j=3,0), (i=3, j=2,0), (i=4, j=1,0), (i=5, j=0,0)
        print(' ', end='')  #한 칸 띄우고 줄 바꿈 하지 않는다.
    for j in range(1, i*2):  #(i=1, j=1,2), (i=2, j=1,4), (i=3, j=1,6), (i=4, j=1,8), (i=5, j=1,10)
        print('*', end='')  #'*'를 출력하고 줄 바꿈 하지 않는다.
    print()
```

[02.흐름제어] ex16.py 실행결과

```
    *
   ***
  *****
 *******
*********
```

- **한줄 FOR 반복문**

[02.흐름제어] ex17.py

```
#리스트 안에 for문을 한 줄로 작성하여 새로운 리스트를 작성이 가능하다.
temp = [i for i in range(10)]
print('1.....', temp)

#'결과 for 반복문'을 리스트 안에 중첩하여 다중 for문이 가능하다.
#(i=1, j=1,4), (i=2, j=1,4)
temp = [i * j for i in range(1, 3) for j in range(1, 4)]
print('2.....', temp)

list = [2, 3, 5, 10, 7]

temp = [i for i in list]
print('3.....', temp)

#for문을 한 줄에 쓰면서 if 조건문을 같이 적용할 수 있다
temp = ['짝수' if i%2==0 else '홀수' for i in list]
print('4.....', temp)

#만약 else가 필요 없다면
temp = [i for i in list if i%2==0]
print('5.....', temp)
```

[02.흐름제어] ex17.py 실행결과

```
1..... [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
2..... [1, 2, 3, 2, 4, 6]
3..... [2, 3, 5, 10, 7]
4..... ['짝수', '홀수', '홀수', '짝수', '홀수']
5..... [2, 10]
```

[02.흐름제어] ex18.py

```
#출석번호가 1, 2, 3, 4 앞에 100을 붙이기로 함 -> 101, 102
students=[1, 2, 3, 4, 5]
students=[i+100 for i in students]
print(students)

#학생이름을 길이로 변환
students=['Iron man', 'chris', 'justin hwang']
students=[len(i) for i in students]
print(students)

#학생이름을 대문자로 변환
students=['Iron man', 'chris', 'justin hwang']
students=[i.upper() for i in students]
print(students)
```

[02.흐름제어] ex18.py 실행결과

```
[101, 102, 103, 104, 105]
[8, 5, 12]
['IRON MAN', 'CHRIS', 'JUSTIN HWANG']
```

- **WHILE 반복문**

[02.흐름제어] ex19.py

```
i=0
sum=0
while i<100:
    i = i + 1
    sum = sum + i

print('1~100까지 합:', sum)
```

[02.흐름제어] ex20.py

```
sum=0
while True:  #무한 반복한다.
    num = input('숫자를 입력하세요(종료:엔터)>')
    if num == '':
        print('sum:', sum)
        break  #while문을 빠져나간다.
    else:
     sum = sum + int(num)

print('프로그램을 종료합니다.')
```

- **고객 이름을 5번 호출하고 받아가지 않으면 폐기처분 메시지를 출력하는 프로그램을 작성한다.**

[02.흐름제어] ex21.py

```
customer='홍길동'
index=5

while index >= 1:  #index가 1보다 크거나 작은 동안에 while문 안에 문장을 반복 실행한다.
    print(f'{customer}님 커피가 준비되었습니다. {index}번 남았어요.')
    index -=1
    if index == 0:
        print('커피가 폐기 처분되었습니다.')
```

- **반복문이 무한 반복되고 중지하려면 Ctrl + C키를 이용한다.**

[02.흐름제어] ex22.py

```
customer = '홍길동'
index = 1

while True:  #while 문의 조건식이 항상 True이므로 while문 안에 문장이 무한 반복한다.
    print(f'{customer}님 커피가 준비되었습니다. 호출 {index}회')
    index += 1
```

- **고객번호가 일치할 때까지 반복 호출하는 프로그램을 작성한다.**

[02.흐름제어] ex23.py

```
customer=20
person=0

while person != customer:  #손님의 번호가 20이 아닌 동안에 반복된다.
    print('{0}번 손님, 커피가 준비되었습니다.'.format(customer))
    person=int(input('몇 번이세요?'))
```

---

03. 함수로 코드 간추리기

파이썬 함수는 특정 작업을 수행하는 코드 블록을 의미하며, 재사용성과 코드 관리 편의성을 높이는 중요한 요소이다. 함수는 입력 값을 받아 처리한 후 결과를 반환할 수 있다. 간단히 말해, 함수는 반복적으로 사용되는 코드를 묶어서 하나의 이름으로 관리하는 것이다.

[03.함수] ex01.py

```
def grade(score):
    if score>=90:
        return 'A'
    elif score>=80 and score<90:
        return 'B'
    elif score>=70 and score<80:
        return 'C'
    elif score>=60 and score<70:
        return 'D'
    else:
        return 'F'

for i in range(1, 6):
    score = int(input(f'{i}번 학생의 점수를 입력하세요?'))
    if score==0:
        print('프로그램을 종료합니다.')
        break
    print(f'{i}번 학생의 학점은 {grade(score)}입니다.\n')
```

[03.함수] ex02.py

```
#계좌생성 함수
def open_account():
    print('새로운 계좌가 생성되었습니다.')

#입금 함수
def deposit(balance, money):
    balance += money
    print('입금이 완료되었습니다. 잔액은 {0}원입니다.'.format(balance))
    return balance

#출금 함수
def withdraw(balance, money):
    if balance >= money:
        balance -=money
        print(f'출금이 완료되었습니다. 잔액은 {balance}원입니다.')
        return balance
    else:
        print(f'잔액이 부족하여 출금이 완료되지 않았습니다. 잔액은 {balance}원입니다.')
        return balance

#저녁에 출금 함수
def withdraw_night(balance, money):
    commission = 100
    balance2 = withdraw(balance, money + commission)
    if (balance != balance2):
        print(f'수수료는 {commission}원입니다.')
        return commission, balance2

balance=0
open_account()  #계좌생성
balance = deposit(balance, 1000)  #1000원 입금
balance = withdraw(balance, 2000)  #2000원 출금
balance = withdraw_night(balance, 500)  #저녁에 500원 출금
```

- **함수의 기본 값 예제**

[03.함수] ex03.py

```
def profile(name, age=17, lang='자바'):
    print(f'이름:{name}, 나이:{age}, 주 사용언어:{lang}')

profile('유재석')
profile('강호동', 20)
profile('김태호', 21, '파이썬')
```

[03.함수] ex03.py 실행결과

```
이름:유재석,나이:17,주사용언어:자바
이름:강호동,나이:20,주사용언어:자바
이름:김태호,나이:21,주사용언어:파이썬
```

- **키워드 값**

[03.함수] ex04.py

```
def profile(name, age=17, lang='자바'):
    print(f'이름:{name}, 나이:{age}, 주 사용언어:{lang}')

profile(age=20, name='유재석')
profile(lang='HTML', age=15, name='김태호')
```

- **가변인자**

[03.함수] ex05.py

```
def profile(name, age, *langs):
    print(f'이름:{name}, 나이:{age}, 주사용언어:', end='')
    for lang in langs:
        print(f'{lang} ', end='')
    print()

profile('유재석', 20, 'Java', 'C', 'C++', 'C#', 'Javascript')
profile('김태호', 25, 'Kotlin', 'HTML')
```

- **지역변수와 전역변수**

[03.함수] ex06.py

```
#전역변수
gun=10
def checkpoint(soldiers):
    global gun
    #gun=12 #지역변수
    gun -= soldiers
    print(f'[함수내] 남은 총:{gun}')

print(f'전체 총:{gun}')

#2명이 경계근무 나감
checkpoint(2)
print(f'남은 총:{gun}')
```

[03.함수] ex07.py

```
#파라미터 변수 사용
gun=10
def checkpoint(gun, soldiers):
    #파라미터로 받은 지역변수
    gun -= soldiers
    print(f'[함수내] 남은 총:{gun}')
    return gun

print(f'전체 총:{gun}')

#함수에 반환한 변수 값을 받는다.
gun = checkpoint(gun, 2)
print(f'남은 총:{gun}')
```

[03.함수] ex07.py실행결과

```
전체 총:10
[함수내] 남은 총:8
남은 총:8
```

Quiz

표준 체중을 구하는 프로그램을 작성하시오. (표준 체중: 각 개인의 키에 적당한 체중)

(성별에 따른 공식)
남자: 키(m) x 키(m) x 22
여자: 키(m) x 키(m) x 21
조건1: 표준 체중은 별도의 함수 내에서 계산
*함수명: std_weight
*전달값: 키(height), 성별(gender)
조건2: 표준 체중은 소수점 둘째자리까지 표시

(출력 예제)
키 175cm 남자의 표준 체중은 67.38kg 입니다.

[03.함수] ex08.py

```
#키:(단위:cm 실수), 성별('남자','여자')
def std_weight(height, gender):
    if gender=='남자':
        std = height/100 * height/100 * 22
    else:
        std = height/100 * height/100 * 21
    return std

height=165
gender='여자'

weight=std_weight(height, gender)
weight=round(weight, 2)

print(f'키 {height}cm {gender}의 표준체중은 {weight}kg입니다.')
```

[03.함수] ex08.py 실행결과

```
키 165cm 여자의 표준체중은 57.17kg입니다.
```

• 람다(lambda) 함수

람다(lambda) 함수는 함수형 프로그래밍에서 중요한 개념 중 하나로, 익명 함수(anonymous function)라고도 부른다. 람다 함수는 이름이 없는 함수로, 일반적으로 함수를 한 번만 사용하거나 함수를 인자로 전달해야 하는 경우에 매우 유용하게 사용된다.

- **다음은 일반함수와 람다함수를 비교한 예제이다.**

[03.함수] ex09.py

```
#일반 함수
def add(x, y):
    return x + y

sum = add(100, 200)
print('일반함수:', sum)

#람다 함수
add = lambda x, y: x + y
sum = add(100, 200)
print('람다함수:', sum)
```

```
일반함수: 300
람다함수: 300
```

- **map() 함수는 시퀀스(리스트, 튜플 등)의 모든 요소에 함수를 적용한 결과를 반환합니다.**

[03.함수] ex10.py

```
myList = [5, 3, 2, 7, 4]
print('1.....', myList)

#myList 배열 아이템 값*2 결과를 리턴 한다.
myList2 = list(map(lambda x: x*2, myList))
print('2.....', myList2)

#myList 배열 아이템이 짝수인지 홀수인지 리턴 한다.
myList2 = list(map(lambda x:'짝수' if x%2==0 else '홀수', myList))
print('3.....', myList2)
```

[03.함수] ex10.py 실행결과

```
1..... [5, 3, 2, 7, 4]
2..... [10, 6, 4, 14, 8]
3..... ['홀수', '홀수', '짝수', '홀수', '짝수']
```

- **filter() 함수는 시퀀스(리스트, 튜플 등)의 모든 요소 중에서 조건에 맞는 요소만을 반환한다.**

[03.함수] ex11.py

```
myList = [1, 2, 3, 4, 5]

#홀수 값만 추출하여 리턴 한다.
myList2 = list(filter(lambda x:x%2==1, myList))
print('1.....', myList2)

#'C'로 시작하는 값만 추출하여 리턴 한다.
myList = ['java', 'python', 'C', 'C++', 'C+']
myList2 = list(filter(lambda x:x.startswith('C'), myList))
print('2.....', myList2)
```

[03.함수] ex11.py 실행결과

```
1..... [5, 3, 2, 7, 4]
2..... [10, 6, 4, 14, 8]
3..... ['홀수', '홀수', '짝수', '홀수', '짝수']
```

- **sorted() 함수는 시퀀스(리스트, 튜플 등)의 요소를 정렬한 결과를 반환한다.**

[03.함수] ex12.py

```
myList = ['chris', 'tom', 'justin']

#오름차순 정렬한 결과를 리턴 한다.
myList2 = sorted(myList)
print('1.....', myList2)

#아이템 글자 수가 작은 순으로 리턴 한다.
myList2 = sorted(myList, key=lambda x:len(x))
print('2.....', myList2)

#아이템 글자 수가 큰 순으로 리턴 한다.
myList2 = sorted(myList, key=lambda x:len(x), reverse=True)
print('3.....', myList2)

myList = [
    {'번호':'01', '이름':'chris'},
    {'번호':'02', '이름':'tom'},
    {'번호':'03', '이름':'justin'},
]
#아이템 번호 순으로 내림차순 정렬한 결과를 리턴 한다.
myList2 = sorted(myList, key=lambda x:x['번호'], reverse=True)
print('4.....', myList2)
```

[03.함수] ex12.py 실행결과

```
1..... ['chris', 'justin', 'tom']
2..... ['tom', 'chris', 'justin']
3..... ['justin', 'chris', 'tom']
4..... [{'번호': '03', '이름': 'justin'}, {'번호': '02', '이름': 'tom'}, {'번호': '01', '이름': 'chris'}]
```

- **reduce() 함수는 시퀀스(리스트, 튜플 등)의 모든 요소를 누적적으로 계산한 결과를 반환한다.**

[03.함수] ex13.py

```
from functools import reduce
myList = [5, 3, 2, 7, 4]

#모든 아이템들을 곱한 결과를 리턴 한다.
result = reduce(lambda x, y:x * y, myList)
print('1.....', result)

#모든 아이템들을 더한 결과를 리턴 한다.
result = reduce(lambda x, y:x + y, myList)
print('2.....', result)
```

[03.함수] ex13.py 실행결과

```
1..... 840
2..... 21
```

04. 모듈과 패키지

파이썬에서 모듈은 관련된 코드(함수, 클래스, 변수 등)를 담고 있는 .py 파일 하나를 의미한다. 패키지는 이러한 모듈들을 논리적으로 묶어 폴더 형태로 구성한 것으로, 관련 모듈들을 계층적으로 관리할 수 있게 해준다. 패키지는 종종 라이브러리라고도 불린다.

• 모듈(Module)

영화 관람료를 계산하는 3개의 함수를 정의하여 여러 방법으로 모듈안의 함수를 실행한다.

▪ **다음은 영화 관람료를 계산하는 3개의 함수를 정의한 모듈(theater_module)이다.**

[04.모듈과 패키지] theater_module.py

```
#일반 가격
def price(people):
 print(f'{people}명 가격은 {people * 10000:,}원 입니다.')

#조조할인 가격
def price_morning(people):
    print(f'{people}명 조조할인 가격은 {people*6000:,}원 입니다.')

#군인 할인 가격
def price_soldier(people):
    print(f'{people}명 군인 할인 가격은 {people*4000:,}원 입니다.')
```

▪ **모듈을(theater_module) import하여 모듈 명으로 함수를 호출하여 실행한다.**

[04.모듈과 패키지] ex01.py

```
import theater_module

#3명이서 영화 보러 갔을 때
theater_module.price(3)

#4명이 조조할인 영화 보러 갔을 때
theater_module.price_morning(4)

#5명의 군인이 영화 보러 갔을 때
theater_module.price_soldier(5)
```

[04.모듈과 패키지] ex01.py 실행결과

```
3명 가격은 30,000원 입니다.
4명 조조할인 가격은 24,000원 입니다.
5명 군인 할인 가격은 20,000원 입니다.
```

▪ **모듈(theater_module)의 이름을 별명(theater)으로 지정하여 별명으로 함수를 호출하여 실행한다.**

[04.모듈과 패키지] ex02.py

```
import theater_module as theater

#3명이서 영화 보러 갔을 때
theater.price(3)

#4명이 조조할인 영화 보러 갔을 때
theater.price_morning(4)

#5명의 군인이 영화 보러 갔을 때
theater.price_soldier(5)
```

[04.모듈과 패키지] ex02.py 실행결과

```
3명 가격은 30,000원 입니다.
4명 조조할인 가격은 24,000원 입니다.
5명 군인 할인 가격은 20,000원 입니다.
```

- **모듈(theater_module)에서 모든 함수들을 import 한다. 모듈 명을 생략할 수 있다.**

[04.모듈과 패키지] ex03.py

```
from theater_module import *

#3명이서 영화 보러 갔을 때
price(3)

#4명이 조조할인 영화 보러 갔을 때
price_morning(4)

#5명의 군인이 영화 보러 갔을 때
price_soldier(5)
```

[04.모듈과 패키지] ex03.py 실행결과

```
3명 가격은 30,000원 입니다.
4명 조조할인 가격은 24,000원 입니다.
5명 군인 할인 가격은 20,000원 입니다.
```

- **모듈(theater_module)에서 일부 함수(price, price_morning)만 import한다.**

[04.모듈과 패키지] ex04.py

```
from theater_module import price, price_morning

#5명이서 영화 보러 갔을 때
price(5)

#4명이 조조할인 영화 보러 갔을 때
price_morning(6)
```

[04.모듈과 패키지] ex04.py 실행결과

```
5명 가격은 50,000원 입니다.
6명 조조할인 가격은 36,000원 입니다.
```

- **모듈(theater_module)에서 특정 함수(price_soldier)에 별명(price)을 지정하여 함수를 실행한다.**

[04.모듈과 패키지] ex05.py

```
from theater_module import price_soldier as price

#5명의 군인이 영화 보러 갔을 때
sprice(5)
```

[04.모듈과 패키지] ex05.py 실행결과

```
5명 군인 할인 가격은 20,000원 입니다.
```

- **패키지(Package)**

travel 패키지 폴더를 생성하여 travel 패키지에 thailand, vietnam 모듈을 생성한다.

- **travel 패지지에 thailand 모듈을 생성한다.**

[04.모듈과 패키지]-[travel] thailand.py

```
class ThailandPackage:
    def detail(self):
        print('[태국 패지지 3박5일] 방콕, 파타야 여행(야시장 투어) 50만원')
```

- **travel 패키지에 vietnam 모듈을 생성한다.**

[04.모듈과 패키지]-[travel] vietnam.py

```
class VietnamPackage:
    def detail(self):
        print('[베트남 패지지 3박 5일] 다낭 효도 여행 60만원')
```

- **travel 패키지 thailand 모듈을 import 한 후 ThailandPage 클래스로 객체를 생성한다. import는 패키지나 모듈만 가능하다.**

[04.모듈과 패키지] ex06.py

```
#import travel.thailand.ThailandPackage 클래스를 import하면 오류가 발생한다.

#travel 패키지 thailand 모듈을 import 한다.
import travel.thailand

trip_to = travel.thailand.ThailandPackage()
trip_to.detail()
```

[04.모듈과 패키지] ex06.py 실행결과

```
[태국 패지지 3박5일] 방콕, 파타야 여행(야시장 투어) 50만원
```

- **travel 패키지 thailand 모듈로부터 ThailandPage 클래스를 travel 패키지로부터 vietnam 모듈을 import 한다.**

[04.모듈과 패키지] ex07.py

```
#travel 패키지 thailand 모듈로부터 ThailandPackage를 import 한다.
from travel.thailand import ThailandPackage

trip_to=ThailandPackage()
trip_to.detail()

#travel 패키지로부터 vietnam 모듈을 import 한다.
from travel import vietnam

trip_to = vietnam.VietnamPackage()
trip_to.detail()
```

[04.모듈과 패키지] ex07.py 실행결과

```
[태국 패지지 3박5일] 방콕, 파타야 여행(야시장 투어) 50만원
[베트남 패지지 3박 5일] 다낭 효도 여행 60만원
```

- **travel 패키지로부터 모든 모듈을 import하면 오류가 발생한다.**

[04.모듈과 패키지] ex08.py ①

```
#travel 패키지로부터 모든 모듈을 import한다.
from travel import *

#오류가 발생한다.
trip_to = vietnam.VietnamPackage()
trip_to.detail()
```

- **__init__.py 파일은 파이썬 패키지를 정의하는데 사용되는 파일이다. 이 파일이 폴더에 존재하면 해당 폴더는 패키지로 인식된다.**
- **__all__ 변수는 'import *'를 했을 때 어떤 것들을 가져올 수 있는지를 배열로 정해준다.**

[06.모듈과 패키지]-[travel] __init__.py

```
__all__ = ['vietnam']
```

- **__init__.py 파일 __name__변수에 'vietnam'을 추가한 후 ex08.py 파일을 실행하면 오류가 발생하지 않는다.**

[04.모듈과 패키지] ex08.py ②

```
#travel 패키지로부터 모든 모듈을 import한다.
from travel import *

trip_to = vietnam.VietnamPackage()
trip_to.detail()
```

[04.모듈과 패키지] ex08.py 실행결과

```
[베트남 패지지 3박 5일] 다낭 효도 여행 60만원
```

- **__init__.py 파일 __all__ 변수에 'thailand' 값을 추가한다.**

[04.모듈과 패키지]-[travel] __init__.py

```
__all__ = ['vietnam', 'thailand']
```

- **travel 패키지의 vietnam 모듈과 thailand 모듈이 import된다.**

[04.모듈과 패키지] ex08.py ③

```
#travel 패키지로부터 모든 모듈을 import한다.
from travel import *

trip_to = vietnam.VietnamPackage()
trip_to.detail()

trip_to = thailand.ThailandPackage()
trip_to.detail()
```

[04.모듈과 패키지] ex08.py 실행결과

```
[베트남 패지지 3박 5일] 다낭 효도 여행 60만원
[태국 패지지 3박5일] 방콕, 파타야 여행(야시장 투어) 50만원
```

- **내부에서 모듈을 실행하면 __name__변수의 값이 '__main__'이 된다.**

[04.모듈과 패키지]-[travel] thailand.py

```
class ThailandPackage:
    def detail(self):
        print('[태국 패지지 3박5일] 방콕, 파타야 여행(야시장 투어) 50만원')

if __name__ == '__main__':
    print('thailand 모듈을 직접 실행')
    print('이 문장은 모듈을 직접 실행할 때만 실행됩니다.')
    trip_to = ThailandPackage()
    trip_to.detail()
else:
    print('thailand 외부에서 모듈 호출')
```

[04.모듈과 패키지]-[travel] thailand.py 실행결과

```
__name__값: __main__
thailand 모듈 내부에서 직접 실행
[태국 패지지 3박5일] 방콕, 파타야 여행(야시장 투어) 50만원
```

- **외부에서 thailand 모듈을 import한 후 실행하면 __name__변수의 값이 'travel.thailand'가 된다.**

[04.모듈과 패키지] ex08.py ④

```
from travel import *

trip_to = vietnam.VietnamPackage()
trip_to.detail()

trip_to = thailand.ThailandPackage()
trip_to.detail()
```

[04.모듈과 패키지] ex08.py 실행결과

```
__name__값: travel.thailand
thailand 모듈 외부에서 호출 실행
[베트남 패지지 3박 5일] 다낭 효도 여행 60만원
[태국 패지지 3박5일] 방콕, 파타야 여행(야시장 투어) 50만원
```

- **패키지, 모듈 위치**

[04.모듈과 패키지] ex09.py

```
import inspect  #모듈이 어느 위치에 있는지 확인

import random
print('1.....', inspect.getfile(random))

from travel import *  #travel 폴더를 random 폴더위치로 이동하면 다른 프로젝트에서 import travel로 사용할 수 있다.
print('2.....', inspect.getfile(thailand))
```

[04.모듈과 패키지] ex09.py 실행결과

```
1..... C:\Users\hdcho\AppData\Local\Programs\Python\Python310\lib\random.py
__name__값: travel.thailand
thailand 모듈 외부에서 호출 실행
2..... c:\data\python\기본문법\06.모듈과패키지\travel\thailand.py
```

- **패키지 관리**

  - 아래 주소로 들어가 필요한 라이브러리를 검색한다.

```
https://pypi.org/
```

  - 새로운 라이브러리를 설치한다.

```
pip install 패키지명
```

  - 설치한 라이브러리의 정보를 출력한다.

```
pip show 패키지명
```

  - 설치한 라이브러리 목록이 출력된다.

```
pip list
```

  - 설치된 패키지를 업그레이드한다.

```
pip install --upgrade 패키지명
```

  - 설치된 패키지를 삭제한다.

```
pip uninstall 패키지명
```

- **내장함수와 외장함수**

  - input 함수는 사용자 입력을 받는 내장함수이다.

[04.모듈과 패키지] ex10.py

```
language = input('무슨 언어를 좋아하세요?')
print(f'{language}은(는) 아주 좋은 언어입니다.')
```

  - dir() 함수는 특정 객체가 가지고 있는 속성(attribute)과 메서드(method)의 목록을 보여주는 내장 함수이다. 객체의 내부 구조를 탐색하고, 어떤 연산을 수행할 수 있는지 확인하는 데 유용하다.

[04.모듈과 패키지] ex11.py

```
print('1.외장함수를 import하기 전----------------------------------------')
print(dir()[-5:])

print('2random 외장함수를 import한 후-----------------------------------')
import random
print(dir()[-5:])

print('3.pickle 외장함수를 import한 후----------------------------------')
import pickle
print(dir()[-5:])

print('4.random 외장함수의 변수와 함수목록-------------------------------')
print(dir(random)[-5:])
```

[04.모듈과 패키지] ex11.py 실행결과

```
1.외장함수를 import하기 전----------------------------------------
['__file__', '__loader__', '__name__', '__package__', '__spec__']
2random 외장함수를 import한 후-------------------------------------
['__loader__', '__name__', '__package__', '__spec__', 'random']
3.pickle 외장함수를 import한 후-----------------------------------
['__name__', '__package__', '__spec__', 'pickle', 'random']
4.random 외장함수의 변수와 함수목록--------------------------------
['shuffle', 'triangular', 'uniform', 'vonmisesvariate', 'weibullvariate']
```

- **변수(list, name, number)에서 사용할 수 있는 함수 목록을 출력한다.**

[04.모듈과 패키지] ex12.py

```
list=[1, 2, 3]
print('1.list 변수에서 사용할 수 있는 함수 목록----------------------------------------')
print(dir(list)[-5:])

name='chris'
print('2.name 변수에서 사용할 수 있는 함수 목록----------------------------------------')
print(dir(name)[-5:])

number=0
print('3.number 변수에서 사용할 수 있는 함수 목록----------------------------------------')
print(dir(number)[-5:])
```

[04.모듈과 패키지] ex12.py 실행결과

```
1.list 변수에서 사용할 수 있는 함수 목록----------------------------------------
['insert', 'pop', 'remove', 'reverse', 'sort']
2.name 변수에서 사용할 수 있는 함수 목록----------------------------------------
['swapcase', 'title', 'translate', 'upper', 'zfill']
3.number 변수에서 사용할 수 있는 함수 목록--------------------------------------
['from_bytes', 'imag', 'numerator', 'real', 'to_bytes']
```

- **[구글]에서 아래 내용을 검색하면 내장함수들을 확인할 수 있다.**

list of python builtins

- **[구글]에서 아래 내용을 검색하면 외장함수들을 확인할 수 있다.**

list of python modules

- **glob 외장함수는 경로 내의 폴더와 파일 목록을 조회한다. 윈도우에서 dir 명령과 같다.**

[04.모듈과 패키지] ex13.py

```
import glob

#현재 폴더 아래 '01.데이터처리' 폴더 안의 확장자가 'py'인 파일 목록
print(glob.glob('./01.데이터처리/*.py')[:3])

#현재 폴더 아래 '02.흐름제어' 폴더 안의 확장자가 'py'인 파일 목록
print(glob.glob('./02.흐름제어/*.py')[:3])
```

[04.모듈과 패키지] ex13.py 실행 결과

```
['./01.데이터처리\\ex01.py', './01.데이터처리\\ex02.py', './01.데이터처리\\ex03.py']
['./02.흐름제어\\ex01.py', './02.흐름제어\\ex02.py', './02.흐름제어\\ex03.py']
```

- **os 모듈은 OS(Operating System)를 제어할 수 있는 유용한 모듈이다. 파일목록을 구하거나, 폴더생성, 폴더삭제 작업을 할 수 있다.**

[04.모듈과 패키지] ex14.py

```
import os

print('현재 폴더.....', os.getcwd())  #현재 디렉터리를 출력한다.

folder = 'sample_dir'
if os.path.exists(folder):
    print(folder, '이미 존재하는 폴더입니다.')
    os.rmdir(folder)
    print(folder, '폴더를 삭제하였습니다.')
else:
    os.makedirs(folder)
    print(folder, '폴더를 생성하였습니다.')

print(os.listdir())  #현재 폴더에 존재하는 폴더목록을 출력한다.
```

[04.모듈과 패키지] ex14.py 실행결과 ①

```
현재 폴더..... C:\data\python\기본문법
sample_dir 폴더를 생성하였습니다.
['01.데이터처리', '02.흐름제어', '03.파일관리', '04.클래스', '05.오류처리', '06.모듈과패키지', 'data', 'sample_dir']
```

[06.모듈과 패키지] ex14.py 실행결과 ②

```
현재 폴더..... C:\data\python\기본문법
sample_dir 이미 존재하는 폴더입니다.
sample_dir 폴더를 삭제하였습니다.
['01.데이터처리', '02.흐름제어', '03.파일관리', '04.클래스', '05.오류처리', '06.모듈과패키지', 'data']
```

- **시간 관련된 외장함수(time)과 날짜 관련된 외장함수(datetime)**

1) time.localtime() 함수는 현재 시스템 시간을 사용해서 지역 시간 정보를 담은 튜플(tuple)을 반환한다. 이 튜플(tuple)은 연도, 월, 일, 시, 분, 초, 요일, GMT(그리니치 표준시) 여부, 그리고 섬머 타임 적용 여부 등의 정보를 포함한다.
2) datetime.timedelta()는 두 날짜의 차이를 계산할 때 사용하는 함수이다. timedelta 객체에는 산술 연산자 +와 -를 사용할 수 있으므로 어떤 날짜에 원하는 기간(일, 시, 분, 초)을 더하거나 뺄 수 있다.

[04.모듈과 패키지] ex15.py

```
import time, datetime

print(time.localtime())
print(time.strftime('%Y-%m-%d %H:%M:%S'))  #현재시간에 포맷을 주어 출력


print(f'오늘날짜는: {datetime.date.today()}')

today = datetime.date.today()
timedelta = datetime.timedelta(days=100)
print(f'오늘부터 100일전 날짜는: {today - timedelta}')
print(f'오늘부터 100일후 날짜는: {today + timedelta}')
```

[04.모듈과 패키지] ex15.py 실행결과

```
2025-08-14 23:08:58
오늘날짜는: 2025-08-14
오늘부터 100일전 날짜는: 2025-05-06
오늘부터 100일후 날짜는: 2025-11-22
```

05. 클래스

객체지향 프로그래밍(Object Oriented Programming)이란 많은 객체(Object)들이 모여서 상호 협력하면서 데이터를 처리하는 방식의 프로그래밍 설계 방법을 일컫는다. 객체지향 프로그래밍(OOP)에서 사용하는 객체(Object)는 클래스(Class)를 사용하여 생성된다. 즉 클래스(Class)는 객체(Object)를 생성하기 위해 정의한 설계도 또는 틀이라 정의할 수 있다.

• 클래스의 정의

클래스 이름은 대문자로 시작하는 것이 일반적이고 생성자 메서드, 변수(데이터 속성)과 함수(메서드)로 구성되어 있다.

- **학생 클래스(Student)를 정의한다. 객체를 생성하는 생성자 메서드(__init__())와 출력 메서드(info_print())로 구성되어있다.**

[05.클래스] Student.py

```
class Student:
    def __init__(self):  #객체 생성자
        self.no=''
        self.name=''
        self.dept='컴퓨터정보공학'  #no, name, dept, birthday 속성, 초깃값을 지정할 수 있다.
        self.birthday=''

    def info_print(self):  #속성 출력 메서드
        print(f'{self.no}, {self.name}, {self.dept}, {self.birthday}')
```

- **학생 클래스를 이용해 학생 객체를 생성하고 학생정보를 출력한다.**

[05.클래스] Student_ex01.py

```
#Student.py 파일에서 Student 클래스를 import한다.
from Student import Student

if __name__=='__main__':
    student = Student()
    student.no = '100'
    student.name = '홍길동'
    student.birthday = '00-10-04'
    student.info_print()

    student.no = '200'
    student.name = '강감찬'
    student.birthday = '02-12-17'
    student.info_print()

    student.no = '300'
    student.name = '이순신'
    student.birthday = '00-10-24'
    student.info_print()

    student.no = '400'
    student.name = '성춘향'
    student.dept = '전기전자공학'
    student.birthday = '00-10-24'
    student.info_print()
```

[05.클래스] Student_ex01.py 실행결과

```
100, 홍길동, 컴퓨터정보공학, 00-10-04
200, 강감찬, 컴퓨터정보공학, 02-12-17
300, 이순신, 컴퓨터정보공학, 00-10-24
400, 성춘향, 전기전자공학, 00-10-24
```

- **성적 클래스를 정의하고 객체를 생성한다.**

[05.클래스] Report.py

```
class Report:
    def __init__(self, no, kor, eng, mat):
        self.no=no
        self.kor=kor
        self.eng=eng
        self.mat=mat
        self.sum=self.kor + self.eng + self.mat
        self.avg=self.sum/3
        self.result=self.result()
        self.grade=self.grade()

    def result(self):
        if self.avg >= 70:
            return 'Pass'
        else:
            return 'Fail'

    def grade(self):
        if self.avg >= 90:
            return 'A'
        elif self.avg >= 80:
            return 'B'
        elif self.avg >= 70:
            return 'C'
        elif self.avg >= 60:
            return 'D'
        else:
            return 'F'

    def info_print(self):
        print(f'번호:{self.no}, 총점:{self.sum}, 평균:{self.avg:.2f}, 평점:{self.grade}, 결과:{self.result}')
```

[05.클래스] Report_ex01.py

```
from Report import Report

if __name__ == '__main__':
    report = Report('100', 80, 95, 97)
    report.info_print()

    report = Report('200', 67, 56, 76)
    report.info_print()

    report = Report('300', 97, 86, 96)
    report.info_print()

    report = Report('400', 87, 86, 86)
    report.info_print()
```

[05.클래스] Report_ex01.py 실행결과

```
번호:100, 총점:272, 평균:90.67, 평점:A, 결과:Pass
번호:200, 총점:199, 평균:66.33, 평점:D, 결과:Fail
번호:300, 총점:279, 평균:93.00, 평점:A, 결과:Pass
번호:400, 총점:259, 평균:86.33, 평점:B, 결과:Pass
```

- **상속**

클래스 상속은 객체 지향 프로그래밍에서 기존 클래스의 속성과 메서드를 재사용하여 새로운 클래스를 정의하는 방법이다. 새로운 클래스는 기존 클래스의 기능을 물려받으면서, 필요에 따라 새로운 기능을 추가하거나 기존 기능을 수정할 수 있다. 이를 통해 코드 재사용성을 높이고, 프로그램의 구조를 더욱 체계적으로 만들 수 있다. 상속의 주요 개념은 다음과 같다.

- **상위 클래스 (부모 클래스, Superclass): 상속을 제공하는 클래스입니다.**
- **하위 클래스 (자식 클래스, Subclass): 상위 클래스를 상속받는 클래스입니다.**

Report 클래스는 아래와 같이 Student 클래스를 상속 받는다. Student 클래스가 부모클래스, Report 클래스가 자식 클래스이므로 자식 클래스 Report는 부모 클래스 Student의 모든 속성과 메서드를 상속 받아 사용할 수 있다.

[05.클래스] Report.py 수정

```
from Student import Student

class Report(Student):
    def __init__(self, no, kor, eng, mat):
        super().__init__()  #Student.__init__(self)와 동일하다.
        self.no=no
        self.kor=kor
        self.eng=eng
        self.mat=mat
        self.sum=self.kor + self.eng + self.mat
        self.avg=self.sum/3
        self.result=self.result()
        self.grade=self.grade()

    def result(self):
        if self.avg >= 70:
            return 'Pass'
        else:
            return 'Fail'

    def grade(self):
        ...

    def info_print(self):
        super().info_print()
        print(f'번호:{self.no}, 총점:{self.sum}, 평균:{self.avg:.2f}, 평점:{self.grade}, 결과:{self.result}')
```

[05.클래스] Report_ex02.py

```
from Report import Report

if __name__=='__main__':
    report = Report('100', 80, 95, 97)
    report.name='홍길동'
    report.info_print()

    report = Report('200', 67, 56, 76)
    report.name='강감찬'
    report.info_print()
```

[05.클래스] Report_ex02.py 실행결과

```
100, 홍길동, 컴퓨터정보공학,
번호:100, 총점:272, 평균:90.67, 평점:A, 결과:Pass
200, 강감찬, 컴퓨터정보공학,
번호:200, 총점:199, 평균:66.33, 평점:D, 결과:Fail
```

- **상품 매출 프로그램**

상품코드, 상품이름, 상품단가, 판매수량을 입력하여 판매일, 판매가격을 출력하는 프로그램을 작성한다.

[05.클래스] Product.py

```
class Product:
    def __init__(self, code, name, price):
        self.code=code
        self.name=name
        self.price=price

    def info_print(self):
        print('****상품정보****')
        print(f'상품코드:{self.code}, 상품이름:{self.name}, 상품단가:{self.price:,}만원')
```

[05.클래스] Product_ex01.py

```
from Product import Product

if __name__ == '__main__':
    product = Product('100', 'LG 비스코프 냉장고', 250)
    product.info_print()
```

[05.클래스] Product_ex01.py 실행결과

```
****상품정보****
상품코드:100, 상품이름:LG 비스코프 냉장고, 상품단가:250만원
```

[05.클래스] Sale.py

```
from Product import Product
from datetime import date

class Sale(Product):
    def __init__(self, code, name, price, qnt):
        Product.__init__(self, code, name, price)
        self.qnt = qnt
        self.date=date.today()
        self.sum = self.qnt * self.price

    def info_print(self):
        super().info_print()
        print('****매출정보****')
        print(f'판매수량:{self.qnt}개, 판매일:{self.date}, 판매총액:{self.sum:,}만원')
```

[05.클래스] Sale_ex01.py

```
from Sale import Sale

if __name__=='__main__':
    sale = Sale('A100','LG 비스코프 냉장고', 250, 100)
    sale.info_print()
```

[05.클래스] Sale_ex01.py 실행결과

```
****상품정보****
상품코드:A100, 상품이름:LG 비스코프 냉장고, 상품단가:250만원
****매출정보****
판매수량:100개, 판매일:2025-08-14, 판매총액:25,000만원
```

Quiz

주어진 코드를 활용하여 부동산 프로그램을 작성하시오.

(출력 예제)
총 4대의 매물이 있습니다.
강남 아파트 매매 10억 2010년
마포 오피스텔 전세 5억 2007년
송파 빌라 월세 500/50 2000년
강서 아파트 매매 8억 2015년
강북 아파트 전세 5억 2015년

[코드]
```
class House:
    #매물 초기화
    def __init__(self, location, house_type, deal_type, price, completion_year):
        pass
    #매물 정보 표시
    def show_detail(self):
        pass
```

[05.클래스] House.py

```
class House:
    def __init__(self, location, house_type, deal_type, price, completion_year):
        self.location=location
        self.house_type=house_type
        self.deal_type=deal_type
        self.price=price
        self.completion_year=completion_year

    def show_detail(self):
        print(self.location, self.house_type, self.deal_type, self.completion_year)

houses = []
house1 = House('강남', '아파트', '매매', '10억', '2010년')
house2 = House('마포', '오피스텔', '전세', '5억', '2007년')
house3 = House('송파', '빌라', '월세', '500/50', '2000년')
house4 = House('강서', '아파트', '매매', '8억', '2015년')
house5 = House('강북', '아파트', '전세', '5억', '2015년')

houses.append(house1)
houses.append(house2)
houses.append(house3)
houses.append(house4)
houses.append(house5)

print(f'총 {len(houses)}대의 매물이 있습니다.')
for house in houses:
    house.show_detail()
```

[05.클래스] House.py 실행결과

```
총 5대의 매물이 있습니다.
강남 아파트 매매 2010년
마포 오피스텔 전세 2007년
송파 빌라 월세 2000년
강서 아파트 매매 2015년
강북 아파트 전세 2015년
```

---

06. 예외처리 방법

에러가 발생하더라도 프로그램 실행을 멈추지 말고 계속 유지하도록 명령해 주는 작업을 예외 처리라고 한다.

- **입력한 값을 정수로 변환하는 경우 정수가 아니면 예외처리를 한다.**

[06.예외처리] ex01.py

```
try:
    num = int(input('숫자를 입력 하세요 : '))

except Exception as err:
    print('입력 값이 숫자가 아닙니다.')
    print('오류메시지:', err)
```

- **입력한 값이 숫자가 아니면 메시지를 출력한 후 다시 입력 받고 숫자인 경우 프로그램을 종료한다.**

[06.예외처리] ex02.py

```
while True:
    try:
        num = int(input('숫자를 입력 하세요 : '))
        break
    except ValueError:
        print('숫자가 아닙니다. 다시 입력하세요.')
```

- **입력 값이 숫자가 아닌 경우 예외처리를 하고 한 자리 숫자가 아닌 경우에도 예외처리를 한다.**

[06.예외처리] ex03.py

```
class BigNumberError(Exception):
    pass

while True:
    try:
        num = int(input('숫자를 입력 하세요 : '))
        if num >= 10:
            raise BigNumberError
        break
    except ValueError as err:
        print('숫자가 아닙니다. 다시 입력 하세요.')
    except BigNumberError:
        print('한 자리 숫자가 아닙니다. 다시 입력 하세요.')
```

[06.예외처리] ex04.py

```
class BigNumberError(Exception):
    def __init__(self, msg):
        self.msg = msg
    def __str__(self):
        return self.msg

while True:
    try:
        num = int(input('숫자를 입력 하세요 : '))
        if num >= 10:
            raise BigNumberError('한 자리 숫자가 아닙니다. 다시 입력 하세요.')
        break
    except ValueError as err:
        print('숫자가 아닙니다. 다시 입력 하세요.')
    except BigNumberError as err:
        print(err)
```

- finally절을 추가하여 에러가 발생하면 '다시 입력하세요.' 메시지를 출력하고 아니면 프로그램을 종료한다.

[06.예외처리] ex05.py

```
class BigNumberError(Exception):
    def __init__(self, msg):
        self.msg = msg
    def __str__(self):
        return self.msg

while True:
    error=False
    try:
        num = int(input('숫자를 입력 하세요 : '))
        if num >= 10:
            raise BigNumberError('한 자리 숫자가 아닙니다.')
    except ValueError as err:
        print('숫자가 아닙니다.')
        error=True
    except BigNumberError as err:
        print(err)
        error=True
    finally:
        if error:
            print('다시 입력해 주세요.')
        else:
            print('프로그램을 종료합니다.')
            break
```

- 숫자를 체크하는 함수를 만들어 입력한 숫자들의 합계를 구하는 프로그램을 작성하시오.

[06.예외처리] ex06.py

```
def isNumber(num):
    try:
        num = int(num)
        return num
    except ValueError:
        print('숫자가 아닙니다. 다시 입력해 주세요.')
        return 0

sum = 0
while True:
    num = input('숫자를 입력 하세요 : ')
    if num=='':
        print('프로그램을 종료합니다.')
        break
    num = isNumber(num)
    sum += num
print(f'합계:{sum}')
```

[06.예외처리] ex06.py 실행결과

```
숫자를 입력 하세요 : 100
숫자를 입력 하세요 : 200
숫자를 입력 하세요 : one
숫자가 아닙니다. 다시 입력해 주세요.
숫자를 입력 하세요 : 300
숫자를 입력 하세요 :
프로그램을 종료합니다.
합계:600
```

Quiz

동네에 항상 대기 손님이 있는 맛있는 치킨집이 있습니다.
대기 손님의 치킨 요리 시간을 줄이고자 자동 주문 시스템을 제작하였습니다.
시스템 코드를 확인하고 적절한 예외처리 구문을 넣으시오.

조건1: 1보다 작거나 숫자가 아닌 입력 값이 들어올 때는 ValueError로 처리
출력 메시지: "잘못된 값을 입력하였습니다."

조건2: 대기 손님이 주문할 수 있는 총 치킨양은 10마리로 한정 치킨 소진 시 사용자 정의 에러[SoldOutError]를 발생시키고 프로그램 종료.
출력 메시지: "재고가 소진되어 더 이상 주문을 받지 않습니다."

[코드]

```
#남은 치킨 수
chicken = 10

#홀 안에는 현재 만석, 대기번호 1번부터 시작
waiting = 1

while(True):
    print("[남은 치킨 : {0}".format(chicken))
    order = int(input("치킨 몇 마리 주문하시겠습니까?"))

    if order > chicken:  #남은 치킨보다 주문량이 많을 때
        print("재료가 부족합니다.")
    else:
        print("[대기번호 {0} {1} 마리 주문이 완료되었습니다.".format(waiting, order))
        waiting += 1
        chicken -= order
```

[06.예외처리] ex07.py

```
class SoldOutError(Exception):
    pass

chicken = 10  #남은 치킨 수
wating = 1  #홀 안에는 현재 만석, 대기번호 1부터 시작

while True:
    try:
        print(f'[남은 치킨:{chicken}마리]')
        order=int(input('치킨 몇 마리 주문하시겠습니까? '))

        if order > chicken:
            print('재료가 부족합니다.')
        elif order <= 0:
           raise ValueError
        else:
            print(f'[대기번호:{wating}번] {order}마리 주문이 완료되었습니다.')
            wating += 1
            chicken -= order
        if chicken==0:
            raise SoldOutError

    except ValueError:
        print('잘못된 값을 입력하였습니다.')
    except SoldOutError:
        print('재고가 소진되어 더 이상 주문을 받지 않습니다.')
        break
```

---

07. 파일에 데이터 읽고 쓰기

- **표준 입출력**

[07.파일관리] ex01.py

```
#end='' 다음 문장을 출력할 경우 줄 바꿈을 하지 않는다.
#sep=',' 변수와 변수 사이에 콤마를 출력한다.
print('Python', 'Java', sep=',', end='')
print(' 무엇이 더 재미있을까요?')
print('-'* 50)

print('Python', 'Java', sep=' vs ', end='')
print(' 무엇이 더 재미있을까요?')
print('-'* 50)

print('Python', 'Java', sep=',', end=' 둘 중 ')
print('무엇이 더 재미있을까요?')
print('-'* 50)
```

[07.파일관리] ex01.py 실행결과

```
Python,Java 무엇이 더 재미있을까요?
--------------------------------------------------
Python vs Java 무엇이 더 재미있을까요?
--------------------------------------------------
Python,Java 둘 중 무엇이 더 재미있을까요?
--------------------------------------------------
```

[07.파일관리] ex02.py

```
scores = {'수학':0, '영어':50, '코딩': 100}

for subject, score in scores.items():
    print(subject.ljust(4), str(score).rjust(4), sep=':')

#zfill(4)는 4개의 공간에 num을 출력한 후 나머지는 0으로 채운다.
for num in range(1, 21):
    print('대기번호:'+ str(num).zfill(4))
```

[07.파일관리] ex02.py 실행결과

```
수학  :   0
영어  :  50
코딩  : 100
대기번호:0001
대기번호:0002
대기번호:0003
대기번호:0004
대기번호:0005
대기번호:0006
대기번호:0007
대기번호:0008
대기번호:0009
대기번호:0010
대기번호:0011
대기번호:0012
대기번호:0013
대기번호:0014
대기번호:0015
대기번호:0016
대기번호:0017
대기번호:0018
대기번호:0019
대기번호:0020
```

[07.파일관리] ex03.py

```
answer = input('아무 숫자나 입력하세요? ')
print(type(answer))

answer = int(answer)
print(type(answer))
print('입력한 값은 '+ str(answer) + '입니다.')
```

[07.파일관리] ex03.py 실행결과

```
아무 숫자나 입력하세요? 3
<class 'str'>
<class 'int'>
입력한 값은 3입니다.
```

- **다양한 출력 포맷**

[07.파일관리] ex04.py

```
#빈자리는 빈 공간으로 두고 오른쪽 정렬을 하되 총10자리 공간을 확보
num=500
print(f'1.{num:>10}')

#양수일 때는 +로 표시 음수일 때는 -로 표시
num=500
print(f'2.{num:>+10}')
num=500
print(f'3.{num:>-10}')

#왼쪽 정렬하고 부호를 출력하고 빈칸으로 '_'로 채움
num=500
print(f'4.{num:_<+10}')

#3자리마다 콤마를 찍어주기
num=1000000
print(f'5.{num:,}')

#3자리마다 콤마를 찍고 부호도 출력
num=1000000
print(f'6.{num:+,}')
num=-1000000
print(f'7.{num:+,}')

#3자리마다 콤마를 찍고 부호도 출력하고 빈자리는 '^'로 채우기
num=1000000000000000
print(f'8.{num:^<+30,}')

#소수점을 특정 자릿수까지만 표시(소수점 3째 자리에서 반올림)
num=5/3
print(f'9.{num:.2f}')
```

[07.파일관리] ex04.py 실행결과

```
1.       500
2.      +500
3.       500
4.+500______
5.1,000,000
6.+1,000,000
7.-1,000,000
8.+1,000,000,000,000,000^^^^^^^^
9.1.67
```

- **파일 입출력**

파이썬에서 파일 입출력은 open() 함수를 사용하여 파일을 열고, read() 또는 write() 함수를 사용하여 데이터를 읽고 쓰는 방식으로 수행된다. 파일 모드는 읽기(r), 쓰기(w), 추가(a) 등을 지정할 수 있다.

- **텍스트 피일 쓰기**

[07.파일관리] ex05.py

```
#1)새로운 파일을 생성한다.
score_file=open('data/score.txt', 'w', encoding='utf-8')
print('수학:80', file=score_file)
print('영어:75', file=score_file)
score_file.close()

#2)파일이 없으면 생성하고 있으면 내용을 추가한다.
score_file=open('data/score.txt', 'a', encoding='utf-8')
score_file.write('과학:90')
score_file.write('\n코딩:88')
score_file.close()
```

- **텍스트 파일 읽기**

[07.파일관리] ex06.py

```
#방법1.
score_file=open('data/score.txt', 'r', encoding='utf-8')
file=score_file.read()
print(file)
score_file.close()
print('-'*30)

#방법2.
score_file=open('data/score.txt', 'r', encoding='utf-8')
print(score_file.readline(), end='')
print(score_file.readline(), end='')
print(score_file.readline(), end='')
print(score_file.readline())
score_file.close()
print('-'*30)

#방법3.
score_file=open('data/score.txt', 'r',encoding='utf-8')
while True:
    line=score_file.readline()
    if not line:
        print()
        break;
    print(line, end='')
score_file.close()
print('-'*30)

#방법4
score_file=open('data/score.txt', 'r', encoding='utf-8')
lines=score_file.readlines()  #list형태로 모든 데이터 저장
print(type(lines))
for line in lines:
    print(line, end='')
score_file.close()
print('-'*30)
```

- pickle은 Python에서 사용하는 딕셔너리, 리스트, 클래스 등의 자료형을 변환 없이 그대로 파일로 저장하고 이를 불러오는 모듈이다.

[07.파일관리] ex07.py

```
#pickle을 사용하면 binary형태로 저장되기 때문에 용량이 매우 작아진다.
import pickle

profile_file=open('data/profile.pickle', 'wb')
profile={
    '이름':'박명수',
    '나이':30,
    '취미':['축구', '골프', '코딩']
}
print(profile)

#profile에 있는 정보를 file에 저장
pickle.dump(profile, profile_file)

profile_file.close()
```

[07.파일관리] ex08.py

```
import pickle

profile_file=open('data/profile.pickle', 'rb')

#profile_file 내용을 읽어온다.
profile=pickle.load(profile_file)
print(type(profile))
print(profile)

#profile에서 키 '취미'값을 읽어온다.
hobbies=profile.get('취미')
print(type(hobbies))
print(hobbies)

profile_file.close()
```

- with를 이용하면 close할 필요가 없다.

[07.파일관리] ex09.py

```
import pickle

#profile.pickle 파일을 binary 형태로 읽어온다.
with open('data/profile.pickle', 'rb') as profile_file:
    print(pickle.load(profile_file))
```

[07.파일관리] ex10.py

```
#text 파일에 문자열을 저장한다.
with open('data/study.txt', 'w', encoding='utf-8') as study_file:
    study_file.write('파이썬을 열심히 공부하고 있어요.')

#text 파일에서 내용을 읽어온다.
with open('data/study.txt', 'r', encoding='utf-8') as study_file:
    file=study_file.read()
    print(file)
```

[07.파일관리] ex11.py

```
with open('data/juso.txt', 'a', encoding='utf-8') as juso_file:
    #키보드에서 이름, 전화, 주소를 입력 받는다.
    name = input('이름: ')
    phone = input('전화: ')
    address = input('주소: ')

    #입력받은 값을 파일에 출력하고 다음 줄로 이동한다.
    juso_file.write(f'{name},{phone},{address}\n')
```

[07.파일관리] ex11.py 실행결과

```
이름: 홍길동
전화: 010-1010-1010
주소: 인천 서구 경서동
```

[07.파일관리] ex12.py

```
with open('data/juso.txt', 'r', encoding='utf-8') as juso_file:
    while True:
        #text 파일을 한 라인씩 읽어온다.
        line=juso_file.readline()

        #line의 값이 없는 경우 반복문을 빠져나간다.
        if not line:
            break

        #line마다 리턴 값이 있으므로 리턴을 제거하기 위해 end 옵션을 준다.
        print(line, end='')
```

Quiz

당신의 회사에서는 매주 1회 작성해야 하는 보고서가 있습니다. 보고서는 항상 아래와 같은 형태로 출력되어야 합니다.

```
- X 주차 주간보고
부서:
이름:
업무요약:
```

1주차부터 50주차까지의 보고서 파일을 만드는 프로그램을 작성하시오.
조건: 파일명은 '1주차.txt,', '2주차.txt', ...와 같이 만듭니다.

[07.파일관리] ex13.py

```
import os

#폴더가 없을 경우 새로 폴더 생성
if not os.path.exists('data/report'):
    os.mkdir('data/report')

#i 변수 값을 range함수를 이용해 1부터 50까지 반복한다.
for i in range(1, 51):
    with open('data/report/'+ str(i) + '주차.txt', 'w', encoding='utf-8') as report_file:
        report_file.write('- {0} 주차 주간보고'.format(i))
        report_file.write('\n부서:')
        report_file.write('\n이름:')
        report_file.write('\n업무요약:')
```

- **파일 변환(텍스트파일 -> JSON파일 -> CSV파일)**

[data] 행정구역.txt

```
시도,시군구,기관유형,상위 보건기관명,보건기관명,주소,읍면동명,도서지역 여부,대표 전화번호
서울특별시,종로구,보건소,종로구보건소,종로구보건소,"서울특별시 종로구 자하문로19길 36 (옥인동, 종로구보건소, 청운효자동자치회관) 1~4층",청운효자동,아니오,02-2148-3514
서울특별시,중구,보건소,서울중구보건소,서울중구보건소,"서울특별시 중구 다산로39길 16 (무학동, 중구보건소) ",신당동,아니오,02-3396-6302
서울특별시,중구,일반보건지소,서울중구보건소,약수보건지소 ,"서울특별시 중구 다산로 92 (신당동, 약수동주민센터) 약수동주민센터",신당동,아니오,02-3396-6964
서울특별시,중구,일반보건지소,서울중구보건소,황학보건지소 ,"서울특별시 중구 난계로11길 52 (황학동, 황학동주민센터) 황학동주민센터",황학동,아니오,02-3396-6989
서울특별시,중구,일반보건지소,서울중구보건소,다산보건지소 ,"서울특별시 중구 동호로15길 50 (신당동, 동사무소어린이집) ",신당동,아니오,02-3396-6959
서울특별시,용산구,보건소,용산구보건소,용산구보건소,"서울특별시 용산구 녹사평대로 150 (이태원동, 용산구종합행정타운) ",이태원동,아니오,02-2199-8350
서울특별시,성동구,보건소,성동구보건소,성동구보건소,"서울특별시 성동구 마장로23길 10 (홍익동, 성동구보건소) ",왕십리도선동,아니오,02-2286-7115
서울특별시,성동구,일반보건지소,성동구보건소,성수보건지소,"서울특별시 성동구 왕십리로5길 3 (성수동1가, 성수1가제2동 공공복합청사) ",성수1가2동,아니오,02-2286-7822
서울특별시,광진구,보건소,광진구보건소,광진구보건소,"서울특별시 광진구 자양로 117 (자양동, 광진구청), 보건소 보건소",자양동,아니오,02-450-1422
```

- **'행정구역.txt' 텍스트 파일을 '행정구역.json' JSON 파일로 변환한다.**

[07.파일관리] ex14.py

```
import csv
import json

#csv 파일을 읽어 json 파일 형식으로 변환
data = []
with open('data/행정구역.txt', 'r', encoding='utf-8') as txt_file:
    reader = csv.DictReader(txt_file)
    data = list(reader)

#변환된 json data를 파일에 저장
with open('data/행정구역.json', 'w', encoding='utf8') as json_file:
    json_file.write(json.dumps(data, indent=4, ensure_ascii=False))
```

- **'행정구역.json' JSON 파일을 '행정구역.csv' CSV 파일로 변환한다.**

[07.파일관리] ex15.py

```
import json
import csv

data = []
with open('data/행정구역.json', 'r', encoding = 'utf-8') as json_file:
    #json 파일을 읽어 data에 저장한다.
    data = json.load(json_file)

with open('data/행정구역.csv', 'w', newline='', encoding='utf-8') as csv_file:
    #csv_file에 저장할 csv writer를 생성한다.
    writer = csv.writer(csv_file)

    #json data key값으로 제목 행을 추가한다.
    cols = data[0].keys()
    writer.writerow(cols)

    #json data를 한행씩 읽어 값을 추가한다.
    for line in data:
        writer.writerow([line[col] for col in cols])
```

- **주소관리 예제 프로그램**

1) 주소 목록을 출력한다. 만약 처음 실행한다면 파일이 생성되지 않았으므로 '입력한 주소 데이터가 없습니다.' 메시지를 출력한다.

[07.파일관리]-[file] menu.py

```
import file

while True:
    print(60 * '-')
    print('1.주소입력|2.주소목록|3.주소검색|4.주소삭제|0.프로그램종료')
    print(60 * '-')

    menu = input('메뉴선택>')
    if not menu.isnumeric():
        print('0~4로 입력하세요.')
        continue
    else:
        menu=int(menu)
    if menu == 0:
        print('프로그램을 종료합니다.')
        break
    elif menu == 1:
        pass
    elif menu == 2:
        file.list()
    elif menu == 3:
        pass
    elif menu == 4:
        pass
```

[07.파일관리]-[file] file.py

```
import os
#print(os.getcwd()) #현재폴더 출력
file_name = './data/address'

def list():
    if os.path.isfile(file_name):
        with open(file_name, 'r', encoding='utf-8') as file:
            lines = file.readlines()
            for line in lines:
                items=line.split(',')
                print(f'{items[0]}\t{items[1]}\t{items[2]}\t{items[3]}', end='')
            print(f'등록된 학생은 {len(lines)}명 입니다.')
    else:
        print('입력한 주소 데이터가 없습니다.')
```

[07.파일관리]-[file] menu.py 실행결과

```
------------------------------------------------------------
1.주소입력|2.주소목록|3.주소검색|4.주소삭제|0.프로그램종료
------------------------------------------------------------
메뉴선택>2
입력한 주소 데이터가 없습니다.
------------------------------------------------------------
1.주소입력|2.주소목록|3.주소검색|4.주소삭제|0.프로그램종료
------------------------------------------------------------
메뉴선택>0
프로그램을 종료합니다.
```

---

2) 주소 입력 프로그램을 작성한다. 입력하기 전에 번호 최댓값을 구하여 리턴 한다.

[07.파일관리]-[file] file.py

```
...
def max_no():
    with open(file_name, 'r', encoding='utf-8') as file:
        lines=file.readlines();
        if lines:
            line = lines[len(lines)-1]
            items=line.split(',')
            return int(items[0])
        else:
            return 0

def insert(name, phone, juso):
    with open(file_name, 'a', encoding='utf-8') as file:
        no=max_no() + 1
        file.write(f'{no},{name},{phone},{juso}\n')
        print('새로운 주소가 등록되었습니다.')
```

[07.파일관리]-[file] menu.py

```
import file

while True:
    ...
    menu = input('메뉴선택>')
    if not menu.isnumeric():
        print('0~4로 입력하세요.')
        continue
    else:
        menu=int(menu)
    if menu == 0:
        print('프로그램을 종료합니다.')
        break
    elif menu == 1:
        name = input('이름>')
        if name == '':
            continue
        phone = input('전화>')
        juso = input('주소>')
        file.insert(name, phone, juso)
    elif menu == 2:
        file.list()
    elif menu == 3:
        pass
    elif menu == 4:
        pass
```

[07.파일관리]-[file] menu.py 실행결과

```
------------------------------------------------------------
1.주소입력|2.주소목록|3.주소검색|4.주소삭제|0.프로그램종료
------------------------------------------------------------
메뉴선택>1
이름>홍길동
전화>010-1010-1010
주소>인천 서구 경성동
새로운 주소가 등록되었습니다.
```

3) 주소 검색 프로그램을 작성한다. 이름을 입력하여 검색하고 이름이 없으면 메시지를 출력한다.

[07.파일관리]-[file] file.py

```
...
def read(name):
    if os.path.isfile(file_name):
        with open(file_name, 'r', encoding='utf-8') as file:
            lines=file.readlines()
        isFind = False
        for line in lines:
            items=line.split(',')
            if name == items[1]:
                isFind = True
                print(f'{items[0]}\t{items[1]}\t{items[2]}\t{items[3]}')
        if isFind == False:
            print(f'{name} 존재하지 않습니다.')
    else:
        print('입력한 주소 데이터가 없습니다.')
```

[07.파일관리]-[file] menu.py

```
import file

while True:
    print(60 * '-')
    print('1.주소입력|2.주소목록|3.주소검색|4.주소삭제|0.프로그램종료')
    print(60 * '-')

    menu = input('메뉴선택>')
    if not menu.isnumeric():
        print('0~4로 입력하세요.')
        continue
    else:
        menu=int(menu)
    if menu == 0:
        print('프로그램을 종료합니다.')
        break
    ...
    elif menu == 3:
        name = input('검색할 이름>')
        if name == '':
            continue
        file.read(name)
    elif menu == 4:
        pass
```

[07.파일관리]-[file] menu.py 실행결과

```
------------------------------------------------------------
1.주소입력|2.주소목록|3.주소검색|4.주소삭제|0.프로그램종료
------------------------------------------------------------
메뉴선택>3
검색할 이름>심청이
심청이 존재하지 않습니다.
------------------------------------------------------------
1.주소입력|2.주소목록|3.주소검색|4.주소삭제|0.프로그램종료
------------------------------------------------------------
메뉴선택>3
검색할 이름>홍길동
1       홍길동  010-1010-1010   인천 서구 경성동
```

4) 주소 삭제 프로그램을 작성한다.

[07.파일관리]-[file] file.py

```
...
def delete(no):
    if os.path.isfile(file_name):
        with open(file_name, 'r', encoding='utf-8') as file:
            lines = file.readlines()
            isFind = False
            new_lines=''
            for line in lines:
                items = line.split(',')
                if no == items[0]:
                    isFind = True
                else:
                    new_lines += line
            if isFind == False:
                print(f'{no}번이 존재하지 않습니다.')
            else:
                with open(file_name, 'w', encoding='utf-8') as file:
                    file.write(new_lines)
                print(f'{no}번 정보가 삭제되었습니다.')
    else:
        print('입력한 주소 데이터가 없습니다.')
```

[07.파일관리]-[file] menu.py

```
import file

while True:
    print(60 * '-')
    print('1.주소입력|2.주소목록|3.주소검색|4.주소삭제|0.프로그램종료')
    print(60 * '-')

    menu = input('메뉴선택>')
    ...
    if menu == 0:
        print('프로그램을 종료합니다.')
        break
    elif menu == 4:
        no = input('삭제할 번호>')
        if no == '':
            continue
        file.delete(no)
```

[07.파일관리]-[file] menu.py 실행결과

```
------------------------------------------------------------
1.주소입력|2.주소목록|3.주소검색|4.주소삭제|0.프로그램종료
------------------------------------------------------------
메뉴선택>4
삭제할 번호>10
10번이 존재하지 않습니다.
------------------------------------------------------------
1.주소입력|2.주소목록|3.주소검색|4.주소삭제|0.프로그램종료
------------------------------------------------------------
메뉴선택>4
삭제할 번호>1
1번 정보가 삭제되었습니다.
```

---

08. 데이터베이스 (SQLite3)

1) SQLite3 데이터베이스 설치하기

아래 주소로 접속해서 'Precompiled Binaries for Windows' 항목의 'sqlite-tools-win-x64~.zip' 파일을 내려 받는다. 내려 받은 파일을 압축 해제하면 SQLite3.exe 파일이 생성된다. 이 실행 파일을 더블클릭하면 셸이 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 셸에 다음과 같이 입력한다. 마지막에 세미콜론(;)을 잊지 말고 입력해야 한다.

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

- SQLite 셸에 '.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 열기 (Connection)
2	커서 열기 (Cursor)
3	커서를 이용하여 데이터 추가/조회/수정/삭제 (execute, commit)
4	커서 닫기(close)
5	커넥션 닫기(close)

1) 데이터베이스 및 테이블을 생성한다. 테이블이 존재하면 삭제하고 새로운 테이블을 생성한다.

[08.SQLite] 01.create.py

```
import sqlite3

con = sqlite3.connect('./data/phone.db')
cursor = con.cursor()

sql = "drop table if exists phonebook"
cursor.execute(sql)

sql  = "create table if not exists phonebook("
sql += "id integer primary key autoincrement,"
sql += "name varchar(32),"
sql += "phone varchar(20),"
sql += "email varchar(64))"
cursor.execute(sql)

cursor.close()
con.close()
print('새로운 테이블이 생성되었습니다.')
```

2) 샘플 데이터 3개를 입력한다.

[08.SQLite] 02.insert.py

```
import sqlite3

con = sqlite3.connect('./data/phone.db')
cursor = con.cursor()

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

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

sql = "insert into phonebook(name, phone, email) values('이순신', '026-542-7576', 'lee@test.com')"
cursor.execute(sql)

id = cursor.lastrowid

con.commit()
cursor.close()
con.close()
print(f'{id}개의 데이터가 입력되었습니다.')
```

3) 샘플 데이터 목록을 출력한다.

[08.SQLite] 03.list.py

```
import sqlite3

con = sqlite3.connect('./data/phone.db')
cursor = con.cursor()

sql = "select * from phonebook"
cursor.execute(sql)

rows = cursor.fetchall()
for row in rows:
    print(f'ID:{row[0]}, NAME:{row[1]}, PHONE:{row[2]}, EMAIL:{row[3]}')

cursor.close()
con.close()
print(f'{len(rows)}개의 데이터가 존재합니다.')
```

[08.SQLite] 03.list.py 실행결과

```
ID:1, NAME:홍길동, PHONE:021-322-1542, EMAIL:hong@test.com
ID:2, NAME:심청이, PHONE:021-445-2424, EMAIL:shim@test.com
ID:3, NAME:이순신, PHONE:026-542-7576, EMAIL:lee@test.com
3개의 데이터가 존재합니다.
```

[08.SQLite] 03.list.py

```
sql = "select * from phonebook order by name desc"
```

[08.SQLite] 03.list.py 실행결과

```
ID:1, NAME:홍길동, PHONE:021-322-1542, EMAIL:hong@test.com
ID:3, NAME:이순신, PHONE:026-542-7576, EMAIL:lee@test.com
ID:2, NAME:심청이, PHONE:021-445-2424, EMAIL:shim@test.com
3개의 데이터가 존재합니다.
```

4) 샘플 데이터 중 이름에 '이'를 포함하면 출력한다.

[08.SQLite] 04.read.py

```
import sqlite3

con = sqlite3.connect('./data/phone.db')
cursor=con.cursor()

sql = "select * from phonebook where name like ?"
cursor.execute(sql, ("%이%",))

rows = cursor.fetchall()
for row in rows:
    print(f'ID:{row[0]}, NAME:{row[1]}, PHONE:{row[2]}, EMAIL:{row[3]}')

cursor.close()
con.close()
print(f'{len(rows)}개의 데이터가 존재합니다.')
```

[08.SQLite] 04.read.py 실행결과

```
ID:2, NAME:심청이, PHONE:021-445-2424, EMAIL:shim@test.com
ID:3, NAME:이순신, PHONE:026-542-7576, EMAIL:lee@test.com
2개의 데이터가 존재합니다.
```

5) 3번 샘플 데이터의 정보를 수정한다.

[08.SQLite] 05.update.py

```
import sqlite3

con = sqlite3.connect('data/phone.db')
cursor = con.cursor()

sql = "update phonebook set phone=?, name=?, email=? where id=?"
cursor.execute(sql, ("010-1010-1010", "김길동", "kim@test.com", "2"))

con.commit()
cursor.close()
con.close()
print('정보가 수정되었습니다.')
```

[08.SQLite] 05.update.py 실행결과

```
정보가 수정되었습니다.
```

- **03.list.py 실행하면 2번 정보가 수정된 것을 확인할 수 있다.**

[08.SQLite] 03.list.py 실행결과

```
ID:1, NAME:홍길동, PHONE:021-322-1542, EMAIL:hong@test.com
ID:3, NAME:이순신, PHONE:026-542-7576, EMAIL:lee@test.com
ID:2, NAME:김길동, PHONE:010-1010-1010, EMAIL:kim@test.com
3개의 데이터가 존재합니다.
```

6) 샘플 데이터 중 1번 데이터를 삭제한다.

[08.SQLite] 06.delete.py

```
import sqlite3

con = sqlite3.connect('./data/phone.db')
cursor = con.cursor()

sql = "delete from phonebook where id=?"
cursor.execute(sql, (1,))

con.commit()
cursor.close()
con.close()
print('하나의 데이터가 삭제되었습니다.')
```

[08.SQLite] 06.delete.py 실행결과

```
하나의 데이터가 삭제되었습니다.
```

- **03.list.py 실행하면 1번 정보가 수정된 것을 확인할 수 있다.**

[08.SQLite] 03.list.py 실행결과

```
ID:3, NAME:이순신, PHONE:026-542-7576, EMAIL:lee@test.com
ID:2, NAME:김길동, PHONE:010-1010-1010, EMAIL:kim@test.com
2개의 데이터가 존재합니다.
```

- **PhoneBook 관리 프로그램**

1) 메뉴 프로그램과 전화 정보를 저장할 Phone 클래스를 정의한다.

[08.SQLite]-[project] menu.py

```
def menu():
    while True:
        print(60 * '-')
        print('|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|')
        print(60 * '-')
        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:
                pass
            elif menu == 2:
                pass
            elif menu == 3:
                pass
            elif menu == 4:
                pass

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

[08.SQLite]-[project] phone.py

```
class Phone:
    def __init__(self):
        self.id=0
        self.name=''
        self.phone=''
        self.email=''
    def detail(self):
        return print(f'ID:{self.id}, NAME:{self.name}, PHONE:{self.phone}, EMAIL:{self.email}')
```

[08.SQLite]-[project] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>3
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>4
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>5
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>0
프로그램을 종료합니다.
```

---

2) 전화번호 목록 프로그램을 작성한다.

[08.SQLite]-[project] database.py

```
import sqlite3

def connect():
    connection = sqlite3.connect('./data/phone.db')
    return connection
def list():
    try:
        con = connect()
        cursor=con.cursor()
        sql="select * from phonebook"
        cursor.execute(sql)
        rows = cursor.fetchall()
        return rows
    except Exception as err:
        print('목록오류', err)
    finally:
        con.close()
```

[08.SQLite]-[project] menu.py

```
import database as db
from phone import Phone

def menu():
    while True:
        print(60 * '-')
        print('|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|')
        print(60 * '-')
        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:
                pass
            elif menu == 2:
                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.detail()
```

[08.SQLite]-[project] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>2
ID:2, NAME:김길동, PHONE:010-1010-1010, EMAIL:kim@test.com
ID:3, NAME:이순신, PHONE:026-542-7576, EMAIL:lee@test.com
```

3) 새로운 전화번호 정보를 입력하는 프로그램을 작성한다.

[08.SQLite]-[project] database.py

```
import sqlite3

def insert(phone):
    try:
        con = connect()
        cursor = con.cursor()
        sql = 'insert into phonebook(name, phone, email) values(?,?,?)'
        cursor.execute(sql, (phone.name, phone.phone, phone.email))
        con.commit()
        print('데이터가 등록되었습니다.')
    except Exception as err:
        print('등록오류', err)
    finally:
        con.close()
```

[08.SQLite]-[project] menu.py

```
import database as db
from phone import Phone

def menu():
    while True:
        print(60 * '-')
        print('|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|')
        print(60 * '-')
        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:
                phone = Phone()
                phone.name = input('이름>')
                if phone.name == '':
                    continue
                phone.phone = input('전화>')
                phone.email = input('이메일>')
                db.insert(phone)
                ...
if __name__ == '__main__':
    menu()
```

[08.SQLite]-[project] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>1
이름>강감찬
전화>010-9897-1234
이메일>kang@test.com
데이터가 등록되었습니다.
```

---

4) 번호를 입력하여 검색하는 프로그램을 작정한다.

[08.SQLite]-[project] database.py

```
import sqlite3
...

def read(id):
    try:
        con = connect()
        cursor = con.cursor()
        sql = 'select id, name, phone, email from phonebook where id=?'
        cursor.execute(sql, (id,))
        row = cursor.fetchone()
        return row
    except Exception as err:
        print('읽기오류', err)
    finally:
        con.close()
```

[08.SQLite]-[project] menu.py

```
def menu():
    while True:
        ...
        if not menu.isnumeric():
            print('숫자로 입력하세요.')
            continue
        else:
            menu = int(menu)
            if menu == 0:
                print('프로그램을 종료합니다.')
                break
                ...
            elif menu == 3:
                id = input('검색할 번호>')
                if id == '': continue
                row = db.read(id)
                if not row:
                    print(f'{id}번 데이터가 존재하지 않습니다.')
                else:
                    phone = Phone()
                    phone.id = row[0]
                    phone.name = row[1]
                    phone.phone = row[2]
                    phone.email = row[3]
                    phone.detail()
```

[08.SQLite]-[project] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>3
검색할 번호>2
ID:2, NAME:김길동, PHONE:010-1010-1010, EMAIL:kim@test.com
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>3
검색할 번호>1
1번 데이터가 존재하지 않습니다.
```

5) 번호를 입력 받아 삭제하는 프로그램을 작성한다.

[08.SQLite]-[project] database.py

```
...
def delete(id):
    try:
        con = connect()
        cursor = con.cursor()
        sql = 'delete from phonebook where id=?'
        cursor.execute(sql, (id,))
        con.commit()
    except Exception as err:
        print('삭제오류', err)
    finally:
        con.close()
```

[08.SQLite]-[project] menu.py

```
import database as db
from phone import Phone

def menu():
    while True:
        ...
        if not menu.isnumeric():
            print('숫자로 입력하세요.')
            continue
        else:
            menu = int(menu)
            if menu == 0:
                print('프로그램을 종료합니다.')
                break
            elif menu == 4:
                id = input('삭제번호>')
                if id == '': continue
                row = db.read(id)
                if not row:
                    print(f'{id}번 데이터가 존재하지 않습니다.')
                else:
                    db.delete(id)
                    print(f'{id}번 데이터가 삭제되었습니다.')
```

[08.SQLite]-[project] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>4
삭제번호>1
1번 데이터가 존재하지 않습니다.
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>4
삭제번호>3
3번 데이터가 삭제되었습니다.
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>2
ID:4, NAME:심청이, PHONE:010-3456-7890, EMAIL:shim@test.com
ID:5, NAME:강감찬, PHONE:010-9897-1234, EMAIL:kang@test.com
```

6) 번호를 입력 받아 전화번호 정보를 수정하는 프로그램을 작성한다.

[08.SQLite]-[project] database.py

```
def update(phone):
    try:
        con = connect()
        cursor = con.cursor()
        sql = 'update phonebook set name=?, phone=?, email=? where id=?'
        cursor.execute(sql, (phone.name, phone.phone, phone.email, phone.id))
        con.commit()
    except Exception as err:
        print('수정오류', err)
    finally:
        con.close()
```

[08.SQLite]-[project] menu.py

```
def menu():
    while True:
        ...
        if not menu.isnumeric():
            ...
        else:
            menu = int(menu)
            if menu == 0:
                ...
            elif menu == 5:
                id = input('수정번호>')
                if id == '':continue
                row = db.read(id)
                if not row:
                    print(f'{id}번 데이터가 존재하지 않습니다.')
                else:
                    phone = Phone()
                    phone.id = row[0]
                    phone.name = row[1]
                    phone.phone = row[2]
                    phone.email = row[3]

                    name = input(f'이름:{phone.name}>')
                    if name != '':phone.name = name
                    tel = input(f'전화:{phone.phone}>')
                    if tel != '':phone.phone = tel
                    email = input(f'이메일:{phone.email}>')
                    if email != '':phone.email = email
                    phone.detail()

                    answer = input('정말로 수정하실래요(y)?')
                    if answer == 'y'or answer == 'Y':
                        db.update(phone)
                        print('수정이 완료되었습니다.')
```

[08.SQLite]-[project] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>5
수정번호>4
이름:김청이>심청이
전화:010-3456-7890>
이메일:shim@test.com>
ID:4, NAME:심청이, PHONE:010-3456-7890, EMAIL:shim@test.com
정말로 수정하실래요(y)?y
수정이 완료되었습니다.
```

---

09. 데이터베이스 (MySQL)

• 데이터베이스 생성

▪ Python에서 MySQL 데이터베이스를 사용하기 위해 아래 패키지를 설치한다.

```
pip install pymysql
```

▪ MySQL 서버를 설치하고 [서비스] 프로그램에서 MySQL 서버가 작동중인지 확인하다.

▪ 데이터베이스(webdb)와 테이블(phonebook)을 생성하고 샘플 데이터를 입력한다.

```
create database webdb;
create user web identified by 'pass';
grant all privileges on webdb.* to web@'%';  /* '%'는 외부에서 접근가능 */

create table phonebook(
    id int auto_increment primary key,
    name char(32),
    phone char(32),
    email char(64)
);

insert into phonebook(name, phone, email) values('홍길동', '021-322-1542', 'hong@test.com');
insert into phonebook(name, phone, email) values('심청이', '021-445-2424', 'shim@test.com');
insert into phonebook(name, phone, email) values('이순신', '026-542-7576', 'lee@test.com');
```

• PhoneBook 관리 프로그램

▪ 데이터베이스 설정파일을 작성한다.

[09.MySQL] config.json

```
{
    "host": "localhost",
    "user": "web",
    "password": "pass",
    "db":"webdb"
}
```

▪ 데이터베이스 설정파일을 이용하여 데이터베이스에 접속한다.

[09.MySQL] database.py

```
import pymysql
import json

file = open('config.json', 'r', encoding='utf-8')
config = json.loads(file.read())
file.close()

connection = pymysql.connect(
    host=config['host'],
    user=config['user'],
    password=config['password'],
    db=config['db'],
    charset='utf8',
    cursorclass=pymysql.cursors.DictCursor)
```

- 'cryptography' 에러가 발생할 경우 아래 패키지를 설치한다.

```
pip install cryptography
```

- 데이터를 저장할 클래스 파일을 정의한다.

[09.MySQL] phone.py

```
class Phone:
    def __init__(self):
        self.id=0
        self.name=''
        self.phoe=''
        self.email=''

    def detail(self):
        print(f"ID:{self.id}, NAME:{self.name}, PHONE:{self.phone}, EMAIL:{self.email}")
```

---

1) 메뉴 프로그램을 작성한다.

[09.MySQL] menu.py

```
import database as db
from phone import Phone

def menu():
    while True:
        print(60 * '-')
        print('|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|')
        print(60 * '-')
        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:
            pass
        elif menu == 2:
            pass
        elif menu == 3:
            pass
        elif menu == 4:
            pass
        elif menu == 5:
            pass

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

[09.MySQL] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>0
프로그램을 종료합니다.
```

---

2) 전화번호 목록을 출력하는 프로그램을 작성한다.

[09.MySQL] database.py

```
import pymysql
import json

def list():
    try:
        with connection.cursor() as cursor:
            sql ='select * from phonebook'
            cursor.execute(sql)
            rows = cursor.fetchall()
            return rows
    except Exception as err:
        print('목록오류', err)
        connection.close()
```

[09.MySQL] menu.py

```
import database as db
from phone import Phone

def menu():
    while True:
        ...
        if not menu.isnumeric():
            print('숫자로 입력하세요.')
            continue
        else:
            menu = int(menu)
        if menu == 0:
            print('프로그램을 종료합니다.')
            break
        elif menu == 1:
            pass
        elif menu == 2:
            rows = db.list()
            for row in rows:
                phone = Phone()
                phone.id=row['id']
                phone.name=row['name']
                phone.phone=row['phone']
                phone.email=row['email']
                phone.detail()
        elif menu == 3:
            pass
        elif menu == 4:
            pass
        elif menu == 5:
            pass
```

[09.MySQL] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>2
ID:1, NAME:홍길동, PHONE:021-322-1542, EMAIL:hong@test.com
ID:2, NAME:심청이, PHONE:021-445-2424, EMAIL:shim@test.com
ID:3, NAME:이순신, PHONE:026-542-7576, EMAIL:lee@test.com
```

3) 전화번호 정보 입력 프로그램을 작성한다.

[09.MySQL] database.py

```
...
def insert(phone):
    try:
        with connection.cursor() as cursor:
            sql = 'insert into phonebook(name,phone,email) values(%s,%s,%s)'
            cursor.execute(sql, (phone.name, phone.phone, phone.email))
            connection.commit()
    except Exception as err:
        print('입력오류', err)
        connection.close()
```

[09.MySQL] menu.py

```
import database as db
from phone import Phone

def menu():
    while True:
        print(60 * '-')
        print('|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|')
        print(60 * '-')
        menu = input('메뉴선택>')

        if not menu.isnumeric():
            print('숫자로 입력하세요.')
            continue
        else:
            menu = int(menu)
        if menu == 0:
            print('프로그램을 종료합니다.')
            break
        elif menu == 2:
            rows = db.list()
            for row in rows:
                phone = Phone()
                phone.id=row['id']
                phone.name=row['name']
                phone.phone=row['phone']
                phone.email=row['email']
                phone.detail()
        elif menu == 3:
            pass
        elif menu == 4:
            pass
        elif menu == 5:
            pass
```

[09.MySQL] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>1
이름>황희원
전화>010-1234-9090
이메일>hwang@test.com
새로운 데이터가 저장되었습니다.
```

4) 검색할 번호를 입력 받아 전화번호 정보를 출력한다.

[09.MySQL] database.py

```
def read(id):
    try:
        with connection.cursor() as cursor:
            sql = 'select * from phonebook where id=%s'
            cursor.execute(sql, id)
            row = cursor.fetchone()
            return row
    except Exception as err:
        print('읽기오류', err)
        connection.close()
```

[09.MySQL] menu.py

```
import database as db
from phone import Phone

def menu():
    while True:
        print(60 * '-')
        print('|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|')
        print(60 * '-')
        menu = input('메뉴선택>')

        if not menu.isnumeric():
            print('숫자로 입력하세요.')
            continue
        else:
            menu = int(menu)
        if menu == 0:
            print('프로그램을 종료합니다.')
            break
        elif menu == 3:
            id = input('조회번호>')
            if id == '': continue
            row = db.read(id)
            if not row:
                print(f'{id}번 데이터가 존재하지 않습니다.')
            else:
                phone = Phone()
                phone.id=row['id']
                phone.name=row['name']
                phone.phone=row['phone']
                phone.email=row['email']
                phone.detail()
```

[09.MySQL] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>3
검색번호>1
ID:1, NAME:홍길동, PHONE:021-322-1542, EMAIL:hong@test.com
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>3
검색번호>10
10번 데이터가 존재하지 않습니다.
```

5) 번호를 입력 받아 전화번호 정보를 삭제한다.

[09.MySQL] database.py

```
def delete(id):
    try:
        with connection.cursor() as cursor:
            sql = 'delete from phonebook where id=%s'
            cursor.execute(sql, id)
            connection.commit()
            print(f'{id}번 데이터가 삭제되었습니다.')
    except Exception as err:
        print('삭제오류', err)
        connection.close()
```

[09.MySQL] menu.py

```
import database as db
from phone import Phone

def menu():
    while True:
        print(60 * '-')
        print('|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|')
        print(60 * '-')
        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:
            ...
        elif menu == 2:
            ...
        elif menu == 3:
            ...
        elif menu == 4:
            id = input('삭제번호>')
            if id == '': continue
            row = db.read(id)
            if not row:
                print(f'{id}번 데이터가 존재하지 않습니다.')
            else:
                db.delete(id)
```

[09.MySQL] menu.py 실행결과

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>4
삭제번호>10
10번 데이터가 존재하지 않습니다.
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>4
삭제번호>1
1번 데이터가 삭제되었습니다.
```

---

6) 번호를 입력받아 전화번호 정보를 수정한다.

[09.MySQL] database.py

```
def update(phone):
    try:
        with connection.cursor() as cursor:
            sql = 'update phonebook set name=%s,phone=%s,email=%s where id=%s'
            cursor.execute(sql, (phone.name, phone.phone, phone.email, phone.id))
            connection.commit()
            print('수정이 완료되었습니다.')
    except Exception as err:
        print('수정오류', err)
        connection.close()
```

[09.MySQL] menu.py

```
def menu():
    while True:
        ...
        if not menu.isnumeric():
            ...
        else:
            menu = int(menu)
        if menu == 0:
            ...
        elif menu == 5:
            id = input('수정번호>')
            if id == '': continue
            row = db.read(id)
            if not row:
                print(f'{id}번 데이터가 존재하지 않습니다.')
                continue
            phone = Phone()
            phone.id=row['id']
            phone.name=row['name']
            phone.phone=row['phone']
            phone.email=row['email']

            name = input(f'이름:{phone.name}>')
            if name != '': phone.name = name
            tel = input(f'전화:{phone.phone}>')
            if tel != '': phone.phone = tel
            email = input(f'{phone.email}>')
            if email != '': phone.email = email

            answer = input('정말로 수정하실래요(y)?')
            if answer == 'y' or answer == 'Y':
                db.update(phone)
```

[09.MySQL] menu.py

```
------------------------------------------------------------
|1.입력|2.목록|3.검색|4.삭제|5.수정|0.프로그램종료|
------------------------------------------------------------
메뉴선택>5
수정번호>2
이름:심청이>김청이
전화:021-445-2424>
shim@test.com>
정말로 수정하실래요(y)?y
수정이 완료되었습니다.
```

- **데이터베이스 검색 결과를 JSON 파일로 변환한다.**

[09.MySQL] db_to_json.py

```
import database as db
import json

def main():
    fileName = './data/phone.json'
    with open(fileName, 'w', encoding='utf-8') as file:
        json_data = json.dumps(db.list(), indent=4, sort_keys=True, ensure_ascii=False)  #ensure_ascii=False 한글로 저장
        file.write(json_data)
        print('json 데이터 저장')

    with open(fileName, 'r', encoding='utf-8') as file:
        rows = json.loads(file.read())
        for row in rows:
            print(row)

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

[09.MySQL] db_to_json.py 실행결과

```
json 데이터 저장
{'email': 'shim@test.com', 'id': 2, 'name': '김청이', 'phone': '021-445-2424'}
{'email': 'lee@test.com', 'id': 3, 'name': '이순신', 'phone': '026-542-7576'}
{'email': 'hwang@test.com', 'id': 4, 'name': '황희원', 'phone': '010-1234-9090'}
```

- **데이터베이스 검색 결과를 Pickle 파일로 변환한다.**

[09.MySQL] db_to_pickle.py

```
import database as db
import pickle

def main():
    fileName = './data/phone.pickle'
    file=open(fileName, 'wb')
    pickle.dump(db.list(), file)
    file.close()
    print('pickle 데이터 저장')

    file=open(fileName, 'rb')
    rows=pickle.load(file)
    file.close()
    for row in rows:
        print(row)

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

[09.MySQL] db_to_pickle.py 실행결과

```
pickle 데이터 저장
{'id': 2, 'name': '김청이', 'phone': '021-445-2424', 'email': 'shim@test.com'}
{'id': 3, 'name': '이순신', 'phone': '026-542-7576', 'email': 'lee@test.com'}
{'id': 4, 'name': '황희원', 'phone': '010-1234-9090', 'email': 'hwang@test.com'}
```

10. 스타크래프트 클래스 프로젝트

클래스는 객체지향 프로그래밍에서 객체를 생성하기 위한 일종의 설계도와 같다. 보통 붕어빵 비유를 많이 드는데 붕어빵을 만드는 붕어빵 틀은 클래스라고 하고 만들어진 붕어빵은 객체라고 볼 수 있다. 클래스는 객체의 상태와 행위가 정의되어 있는 일종의 설계도이기 때문에 비슷한 구조를 갖지만 상태는 서로 다른 많은 객체를 만들 수 있다.

- **일반적인 변수를 사용한 프로그램**

[10.스타크래프트] ex01.py

```
#마린: 공격유닛, 군인, 총을 쏠 수 있음
name='마린'  #유닛이름
hp=40  #유닛의 체력
damage=5  #유닛의 공격력
print(f'{name}유닛이 생성되었습니다.')
print(f'체력:{hp} 공격력:{damage}\n')

#탱크: 공격유닛, 탱크, 포를 쏠 수 있으며 일반모드/시즈모드
tank_name='탱크'
tank_hp=150
tank_damage=35
print(f'{tank_name}유닛이 생성되었습니다.')
print(f'체력:{tank_hp} 공격력:{tank_damage}\n')

def attack(name, location, damage):
    print(f'{name}:{location}방향으로 적군을 공격합니다. [공격력:{damage}]')

attack(name, '1시', damage)
attack(tank_name, '1시', tank_damage)
```

- **클래스를 사용한 프로그램**

[10.스타크래프트] ex02.py

```
class Unit:
    def __init__(self, name, hp, damage): #__init__ 생성자함수 name, hp, damage 멤버변수
        self.name=name
        self.hp=hp
        self.damage=damage
        print(f'{self.name}유닛이 생성되었습니다.')
        print(f'체력:{self.hp} 공격력:{self.damage}')

marine1 = Unit('마린', 40, 5)
marine2 = Unit('마린', 40, 5)
tank = Unit('탱크', 150, 35)
#marine3=Unit("마린") 생성자 변수 개수와 같게 정의하지 않으면 오류가 발생한다.

#레이스: 공중유닛, 비행기, 클로킹기능(상대방에게 보이지 않음)
wraith1 = Unit('레이스', 80, 5)
#클래스 외부에서도 멤버변수에 접근 가능하다.
print(f'유닛이름:{wraith1.name} 공격력:{wraith1.damage}')

#마인드컨트롤: 상대방 유닛을 내 것으로 만드는 것 (빼앗음)
wraith2 = Unit('빼앗은 레이스', 80, 5)

#클래스 외부에서 변수를 확장해서 사용가능하다.
wraith2.clocking = True
if wraith2.clocking == True:
    print(f'{wraith2.name}는 현재 클로킹 상태입니다.')
# if wraith1.clocking == True: wraith1에는 clocking 변수가 없으므로 오류가 발생한다.
```

- **메서드 : 해당 객체의 행위(동작)를 나타낸다. python에서는 모든 함수가 메서드이다.**

[10.스타크래프트] ex03.py ①

```
#ex02.py에서 복사해 온다.
class Unit:
    def __init__(self, name, hp, damage):
        self.name=name
        self.hp=hp
        self.damage=damage
        print(f'{self.name}유닛이 생성되었습니다.')
        print(f'체력:{self.hp} 공격력:{self.damage}')

class AttackUnit:
    def __init__(self, name, hp, damage):
        self.name=name
        self.hp=hp
        self.damage=damage

    def attack(self, location):
        print(f'{self.name}:{location}방향으로 적군을 공격합니다. [공격력:{self.damage}]')

    def damaged(self, damage):
        print(f'{self.name}:{damage}데미지를 입었습니다.')
        self.hp -= damage
        print(f'{self.name}:현재 체력은 {self.hp}입니다.')
        if self.hp <= 0:
            print(f'{self.name}:파괴되었습니다.')

#파이어뱃 : 공격유닛, 화염방사기
firebat1 = AttackUnit('파이어뱃', 50, 16)
firebat1.attack('5시')

#공격을 2번 받는다고 가정
firebat1.damaged(25)
firebat1.damaged(25)
```

- **상속 : 클래스에서 상속이란, 물려주는 클래스(Parent Class, Super class)의 내용(속성과 메서드)을 물려받는 클래스(Child class, sub class)가 가지게 되는 것입니다.**

[10.스타크래프트] ex03.py ②

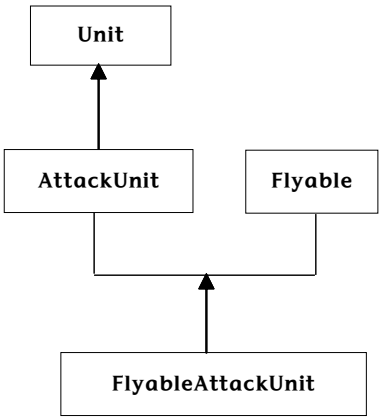
```
#일반유닛 클래스
class Unit:
    def __init__(self, name, hp):  #메딕:의무병, 공격력이 없는 유닛이므로 damage를 생략한다.
        self.name=name
        self.hp=hp

class AttackUnit(Unit):
    def __init__(self, name, hp, damage):
        Unit.__init__(self, name, hp)
        self.damage=damage

    def attack(self, location):
        print(f'{self.name}:{location}방향으로 적군을 공격합니다. [공격력:{self.damage}]')

    def damaged(self, damage):
       ...
    #파이어뱃 : 공격유닛, 화염방사기
    ...
```

- **다중상속**

[10.스타크래프트] ex03.py ③

```
#일반유닛 클래스
class Unit:
    def __init__(self, name, hp):  #메딕:의무병, 공격력이 없는 유닛이므로 damage를 생략한다.
        self.name=name
        self.hp=hp

class AttackUnit(Unit):
    def __init__(self, name, hp, damage):
        Unit.__init__(self, name, hp)
        self.damage=damage

    def attack(self, location):
        print(f'{self.name}:{location}방향으로 적군을 공격합니다. [공격력:{self.damage}]')

    def damaged(self, damage):
        print(f'{self.name}:{damage}데미지를 입었습니다.')
        self.hp -= damage
        print(f'{self.name}:현재 체력은 {self.hp}입니다.')

        if self.hp <= 0:
            print(f'{self.name}:파괴되었습니다.')

#드랍쉽: 공중유닛, 수송기(마린, 파이어뱃, 탱크등을 수송), 공격기능 없음
#날 수 있는 기능을 가진 클래스
class Flyable:
    def __init__(self, flying_speed):
        self.flying_speed=flying_speed

    def fly(self, name, location):
        print(f'{name}:{location}방향으로 날아갑니다. [속도:{self.flying_speed}]')

#공중 공격 유닛 클래스
class FlyableAttactUnit(AttackUnit, Flyable):
    def __init__(self, name, hp, damage, flying_speed):
        AttackUnit.__init__(self, name, hp, damage)
        Flyable.__init__(self, flying_speed)

#발키리 : 공중 공격 유닛, 한번에 14발 미사일 발사
valkyrie = FlyableAttactUnit('발카리', 200, 6, 5)
valkyrie.fly(valkyrie.name, '3시')
```

- **다중상속 다이어그램**



- **연산자 오버라이딩**

[10.스타크래프트] ex03.py ④

```
#일반유닛 클래스
class Unit:
    def __init__(self, name, hp, speed):  #메딕:의무병, 공격력이 없는 유닛이므로 damage를 생략한다.
        self.name=name
        self.hp=hp
        self.speed=speed

    def move(self, location):
        print('[지상 유닛 이동]')
        print(f'{self.name}:{location}방향으로 이동합니다. [속도:{self.speed}]')

class AttackUnit(Unit):
    def __init__(self, name, hp, speed, damage):
        Unit.__init__(self, name, hp, speed)
        self.damage=damage

    def attack(self, location):
        print(f'{self.name}:{location}방향으로 적군을 공격합니다. [공격력:{self.damage}]')

    def damaged(self, damage):
        print(f'{self.name}:{self.damage}데미지를 입었습니다.')
        self.hp -= damage
        print(f'{self.name}:현재 체력은 {self.hp}입니다.')

        if self.hp <= 0:
            print(f'{self.name}:파괴되었습니다.')

#드랍쉽: 공중유닛, 수송기(마린, 파이어뱃, 탱크등을 수송), 공격기능 없음
#날 수 있는 기능을 가진 클래스
class Flyable:
    def __init__(self, flying_speed):
        self.flying_speed=flying_speed

    def fly(self, name, location):
        print(f'{name}:{location}방향으로 날아갑니다. [속도:{self.flying_speed}]')

#공중 공격 유닛 클래스
class FlyableAttactUnit(AttackUnit, Flayable):
    def __init__(self, name, hp, damage, flying_speed):
        AttackUnit.__init__(self, name, hp, 0, damage)  #지상 speed 0
        Flyable.__init__(self, flying_speed)

    def move(self, location):  #연산자 오버로딩
        print('[공중 유닛 이동]')
        self.fly(self.name, location)

#벌쳐 : 지상 유닛, 기동성 좋음
vulture = AttackUnit('벌쳐', 80, 10, 20)

#배틀크루저 : 공중유닛, 체력도 굉장히 좋음
battlecuiser= FlyableAttactUnit('배틀크루저', 500, 25, 3)
vulture.move('11시')

#battlecuiser.fly(battlecuiser.name, '9시')
battlecuiser.move('9시')
```

- **pass**

[10.스타크래프트] ex03.py ⑤

```
...
#건물
class BuildingUnit(Unit):
    def __init__(self, name, hp, location):
        pass  #아무것도 하지 않고 일단 넘어간다.

#서플라이 디폿: 건물, 1개의 건물 = 8유닛 생성
supply_depot = BuildingUnit('서플라이 디폿', 500, '7시')

def game_start():
    print('[알림] 새로운 게임을 시작합니다.')

defgame_over():
    pass

game_start()
game_over()
```

- **super**

[10.스타크래프트] ex03.py ⑥

```
#일반유닛 클래스
class Unit:
    def __init__(self, name, hp, speed):  #메딕:의무병, 공격력이 없는 유닛이므로 damage를 생략한다.
        self.name=name
        self.hp=hp
        self.speed=speed
...
class BuildingUnit(Unit):
    def __init__(self, name, hp, location):
        #Unit.__init__(self, name, hp, 0)
        super().__init__(name, hp, 0)  #self를 생략한다.
        self.location = location
```

- **다중상속 시 super() 사용의 문제점**

[10.스타크래프트] ex04.py

```
class Unit:
    def__init__(self):
        print('Unit 생성자')

class Flyable:
    def__init__(self):
        print('Flyable 생성자')

class FlyableUnit(Unit, Flyable):
    def__init__(self):
        #super().__init__() 다중상속을 받는 경우 처음 상속받은 생성자만 호출이 된다.
        Unit.__init__(self)
        Flyable.__init__(self)

#드랍쉽
dropship = FlyableUnit()
```

- **스타크래프트 전반전**

ex03.py 파일내용을 복사한 후 건물 class를 삭제하여 아래와 같은 내용을 추가한다.

[10.스타크래프트] ex05.py ①

```
class Unit:
    def __init__(self, name, hp, speed):
        self.name=name
        self.hp=hp
        self.speed=speed

        #1)새로 추가한다.
        print(f'{self.name}유닛이 생성되었습니다.')

    def move(self, location):
        print(f'{self.name}:{location}방향으로 이동합니다. [속도:{self.speed}]')

    #2)일반 유닛도 데미지를 입으므로 AttackUnit에서 이동 해옴
    def damaged(self, damage):
        print(f'{self.name}:{damage}데미지를 입었습니다.')

        self.hp -= damage
        print(f'{self.name}:현재 체력은 {self.hp}입니다.')

        if self.hp <= 0:
            print(f'{self.name}:파괴되었습니다.')

class AttackUnit(Unit):
    def __init__(self, name, hp, speed, damage):
        Unit.__init__(self, name, hp, speed)
        self.damage=damage

    def attack(self, location):
        print(f'{self.name}:{location}방향으로 적군을 공격합니다. [공격력:{self.damage}]')

class Flyable:
    def __init__(self, flying_speed):
        self.flying_spped=flying_speed

    def fly(self, name, location):
        print(f'{name}:{location}방향으로 날아갑니다. [속도:{self.flying_spped}]')

class FlyableAttackUnit(AttackUnit, Flyable):
    def __init__(self, name, hp, damage, flying_speed):
        AttackUnit.__init__(self, name, hp, 0, damage)
        Flyable.__init__(self, flying_speed)

    def move(self, location):
        self.fly(self.name, location)

def game_start():
    print('[알림] 새로운 게임을 시작합니다.')

#3)일반 유닛도 데미지를 입으므로 AttackUnit에서 이동 해옴
def game_over():
    print('Player : gg')
    print('[Player]님이 게임에서 퇴장하셨습니다.')
```

[10.스타크래프트] ex05.py ②

```
#4)마린 유닛 생성
class Marine(AttackUnit):
    def __init__(self):
        AttackUnit.__init__(self, '마린', 40, 1, 5)  #체력:40, 스피드:1, 공격력:5

    #스팀팩: 일정 시간 동안 이동 및 공격 속도를 증가, 체력10 감소
    def stimpack(self):
        #체력이 10보다 크거나 같은 경우
        if self.hp >= 10:
            self.hp -= 10
            print(f'{self.name}:스팀팩을 사용합니다. (HP 10감소)')
        #체력이 10보다 작은 경우
        else:
            print(f'{self.name}:체력이 부족하여 스팀팩을 사용하지 않습니다.')

#5)탱크 유닛 생성
class Tank(AttackUnit):
    #시즈모드: 탱크를 지상에 고정시켜 더 높은 파워로 공격가능, 이동불가 (모든 탱크에 적용)
    #시즈모드 개발여부
    seize_developed = False

    def __init__(self):
        #체력:150, 스피드:1, 공격력:35
        AttackUnit.__init__(self, '탱크', 150, 1, 35)
        self.seize_mode = False

    def set_seize_mode(self):
        if Tank.seize_developed == False:
            return

        #현재 시즈모드가 아닐때 -> 시즈모드 ON
        if self.seize_mode == False:
            print(f'{self.name}:시즈모드로 전환합니다.')
            self.damage *= 2
            self.seize_mode = True
        else:
            #현재 시즈모드일때 -> 시즈모드 OFF
            print(f'{self.name}:시즈모드를 해제합니다.')
            self.damage /= 2
            self.seize_mode = False

#6)레이스 유닛 생성
class Wraith(FlyableAttackUnit):
    def __init__(self):
        FlyableAttackUnit.__init__(self, '레이스', 80, 20, 5)
        #클로킹 모드 (해제 상태)
        self.clocked = False

    def clocking(self):
        #클로킹 모드 -> 모드해제
        if self.clocked == True:
            print(f'{self.name}:클로킹 모드 해제합니다.')
            self.clocked = False
        #클로킹 모드 : 모드설정
        else:
            print(f'{self.name}:클로킹 모드 설정합니다.')
            self.clocked = True
```

- **스타크래프트 후반전**

[10.스타크래프트] ex05.py ③

```
#실제 게임 진행
game_start()

#마린 3기 생성
m1 = Marine()
m2 = Marine()
m3 = Marine()

#탱크 2기 생성
t1 = Tank()
t2 = Tank()

#레이스 1기 생성
w1 = Wraith()

#유닛 일괄 관리
attack_units = []
attack_units.append(m1)
attack_units.append(m2)
attack_units.append(m3)
attack_units.append(t1)
attack_units.append(t2)
attack_units.append(w1)

#전군 이동
for unit inattack_units:
    unit.move('1시')

#탱크 시즈모드 개발
Tank.seize_developed = True
print('[알림] 탱크 시즈모드 개발이 완료되었습니다.')

#공격 모두 준비(마린:스팀팩, 탱크:시즈모드, 레이스:클로킹)
for unit in attack_units:
    if isinstance(unit, Marine):
        unit.stimpack()
    elif isinstance(unit, Tank):
        unit.set_seize_mode()
    elif isinstance(unit, Wraith):
        unit.clocking()

#전군 공격
for unit in attack_units:
    unit.attack('1시')

#random 패키지 import
from random import*

#전군피해
for unit in attack_units:
    #공격은 Random으로 받음(5~20)
    damaged=randint(5, 21)
    unit.damaged(damaged)

#게임 종료
game_over()
```