1. 프로그래밍 언어

1.1. 프로그래밍 언어

프로그래밍 언어 : 컴퓨터와의 의사 소통을 위한 수단

1.2. 프로그래밍 언어의 발전

1972년 C(Bell 연구소, 절차지향적인 언어 - 함수 중심)
1983년 C++(Bell 연구소, 객체지향적인 언어 - 클래스 중심)
1991년 Python 발표
1995년 Java(Sun Microsystems), 2009년 Oracle에서 인수
2002년 C#(Microsoft)

1.3. 컴파일러(Compiler)

소스 코드 전체를 컴퓨터가 이해할 수 있는 바이트 코드로 변환하는 프로그램

1.4. 인터프리터(Interpreter)

소스 코드를 한 줄씩 읽어서 컴퓨터가 이해할 수 있는 바이트 코드로 변환하는 프로그램

2. 파이썬 개발환경 설정

2.1. 파이썬의 특징

인기있는 프로그래밍 언어

https://www.tiobe.com/tiobe-index/

2022년 11월 현재 파이썬의 순위는 1위

배우기 쉬운 언어

인터프리터 언어 : 컴파일 과정이 없으므로 빠른 실행이 가능함,

오픈소스

뛰어난 성능

강력한 데이터 분석 라이브러리 제공

3. 파이썬 설치

파이썬 2와 3, 하위 호환성이 없음

3.1. 파이썬의 주요 라이브러리

numpy - 수치 계산용 라이브러리
pandas - 데이터 분석용 라이브러리
matplotlib - 그래프 출력을 위한 라이브러리
scikit-learn - 머신러닝을 위한 라이브러리
tensorflow, keras, pytorch - 딥러닝을 위한 라이브러리

3.2. Anaconda 배포판 - 데이터 분석에 유용한 라이브러리가 많이 포함된 배포판

http://www.anaconda.com

For Windows Python 3.9 • 64-Bit Graphical Installer

c:₩anaconda3 에 설치 (c 드라이브에 anaconda3 디렉토리를 먼저 만들어야 함)

기본적인 옵션으로 설치
설치 시간이 오래 걸림

설치가 완료된 후 환경변수 path에 python 실행을 위한 디렉토리 추가

c:₩anaconda3
c:₩anaconda3₩scripts
c:₩anaconda3₩Library₩bin

3.3. ipython

cmd에서 ipython.exe 실행

```
a=5
print(a)
```

print() 함수를 사용하지 않고도 값을 확인할 수 있습니다.

```
a
```

자동으로 들여쓰기를 제공합니다.

```
for i in range(3):
    print(i)
```

```
num = 10
if num>4:
    print(True)
else:
    print(False)
```

함수 정의

```
def message():
    print("hello python")
```

함수 호출

```
message()
```

화면 지우기

```
cls
```

도움말 사용 방법 : 변수, 함수 뒤에 ? 입력

```
num?
```

함수 이름만 작성한 후 ?(소괄호는 작성하지 않음)

```
print?
```

매직명령어 : ipython에서만 사용 가능한 명령어, %로 시작하는 명령어

현재 사용 중인 변수 목록

```
%who
```

변수 제거

```
del num
```

변수 목록 확인

```
%who
```

모든 변수 제거

```
%reset

%who
```

시간 측정

range(n) : 0 ~ n-1

sum(n) : 1~n까지의 합계

10 ** 5 : 10의 5승

```
%time sum(range(10**5))
```

3.4. ipython notebook

시작 버튼을 누르고 jupyter Notebook (anaconda3) 을 찾아서 실행

Untitled를 더블클릭하여 이름 변경

하나의 셀에 여러 줄 입력이 가능함

```
a=1
b=2
c=3
```

실행버튼 클릭(또는 shift + Enter)

+ 버튼을 누르면 새로운 셀이 추가됩니다.
In 옆에 숫자가 있으면 실행된 셀, 없으면 실행되지 않은 셀

위,아래 화살표 코드를 위 아래로 이동

In 옆 숫자가 가장 큰 것이 최근에 실행된 셀

원형 화살표(Restart Kernel) : 메모리에 저장된 내용이 지워짐

Restart & Clear output : 실행결과가 모두 지워짐

Run All : 모두 순차적으로 실행

File - Download as - html, pdf 등 다양한 포맷으로 변환 가능
ipython notebook 종료 : 웹브라우저와 cmd를 모두 종료해야 함

기본 시작 디렉토리는 C:₩사용자₩아이디 로 자동 설정되어 있음

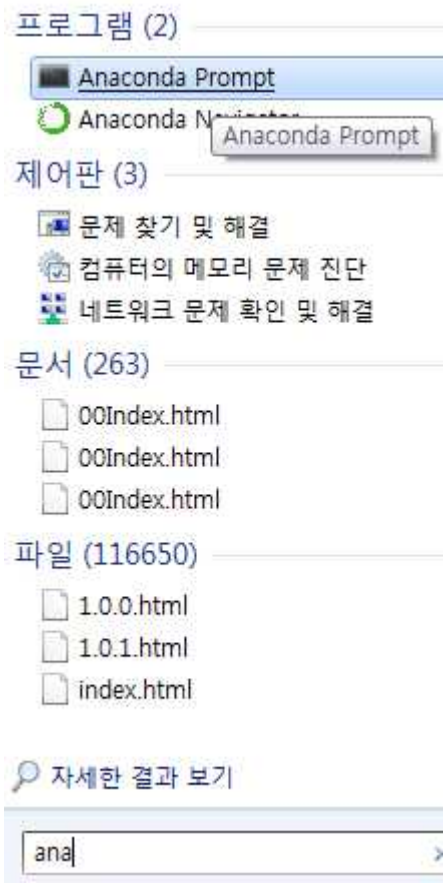
jupyter notebook의 아이콘의 속성을 열어 보면, 대상(target)에 아래와 같이 맨 끝에 %USERPROFILE% 이 붙어 있습니다.
%USERPROFILE% 은 시작 폴더를 C:₩Users₩아이디 으로 설정하는 역할을 합니다.

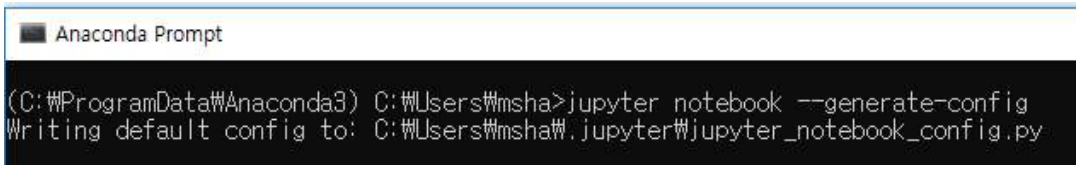
C:₩Anaconda3₩python.exe C:₩Anaconda₩cwp.py C:₩Anaconda C:₩Anaconda3₩python.exe C:₩Anaconda₩Scripts₩jupyter-notebook-script.py "%USERPROFILE%/"

"%USERPROFILE%/" 을 지웁니다.

Anaconda prompt 를 실행한 후 명령어 입력

```
jupyter notebook --generate-config
```





사용자 계정 디렉토리 아래에 다음과 같은 파일이 만들어집니다.

C:₩Users₩아이디₩.jupyter₩jupyter_notebook_config.py

jupyter_notebook_config.py 편집

#c.NotebookApp.notebook_dir = ''
#을 지우고 ipython notebook을 시작할 홈 디렉토리를 지정합니다.
백슬래시(₩) 대신 슬래시(/)를 입력해야 정상적으로 실행됩니다.

385번 라인 수정 c.NotebookApp.notebook_dir = 'c:/python'

ipython notebook을 실행하면 위에서 지정한 디렉토리가 기본 디렉토리로 표시됩니다.

3.5. visual studio code 설치

https://code.visualstudio.com

Windows x64 Installer 다운로드(system, user 모두 가능)

언어팩 설치 후 재시작

설치 완료 후 python 익스텐션 설치(첫번째 익스텐션 설치)

4. 기본문법

파이썬 표준 라이브러리 문서

https://docs.python.org/3/library/index.html

실행 방법 : cell에 코드 입력 후 Shift+Enter

예약어(Reserved Words)

파이썬에서 이미 문법적인 용도로 사용되고 있기 때문에 변수명 등의 식별자로 사용하면 안 되는 단어들

파이썬에서 이미 사용되고 있는(용도가 예약된) 단어들

예약어를 변수에 사용하면 에러는 없으나 고유 기능은 사라짐

Python 3.9에서는 36개의 예약어가 사용됨

```
#파이썬의 예약어
#keyword 모듈 로딩
import keyword
print(keyword.kwlist)
print(len(keyword.kwlist))
```

```
1+2
```

```
3/2.4
```

```
3*9
```

```
# 한줄 주석
"""
여러줄 주석(큰 따옴표 또는 작은 따옴표 3개)
"""
print(10)
```

변수 작성 규칙
첫글자는 영문 대소문자, _로 시작
나머지 글자들은 영문자, 숫자, _로 구성
대소문자 구분
길이 제한 없음
예약어는 사용할 수 없음

```
a=10
a+=1
print(a)
#a++, a--는 지원되지 않음 : a+=1   a-=1로 해야 함

#문자열은 작은따옴표 또는 큰따옴표로 감싼다.
a="파이썬"
print(a)
```

```
# 자료형이 없음(가변자료형)
a=3
b=4
# 3의 4승
a**b
```

```
# 여러개의 변수에 할당 가능
a, b = 10, 20
print(a,b)
i=j=k=10
print(i,j,k)
```

```
10 20
10 10 10
```

```
#변수의 swap
a , b = 10, 20
print(a,b)
a , b = b , a
print(a,b)
```

```
10 20
20 10
```

```
#변수의 삭제 : del
b = 2
print(b)
del b
print(b)
```

```
# 나머지 연산자
print(7%3)

print(7/4)

# 나눗셈 후 소수 이하를 버림(정수몫)
print(7//4)
```

```
1
1.75
1
```

```
#출력 형식
x=9
print("정수 1 : {0}".format(x))
print("정수 2 : {0}".format(3**4)) #제곱
print("정수 3 : {0}".format((int(8.3)/int(2.7)))) #강제형변환
print(f'정수 1 : {x}')
print(f'정수 2 : {3**4}')
print(f'정수 3 : {(int(8.3)/int(2.7))}')
```

```
#소수 이하 자리수 지정 방법
print("실수 1 : {0:.3f}".format(8.3/2.7))
y=2.5*4.8
print("실수 2 : {0:.1f}".format(y))
r=8/float(3)
print("실수 3 : {0:.2f}".format(r))
print("실수 4 : {0:.4f}".format(8.0/3))
```

실수 1 : 3.074
실수 2 : 12.0
실수 3 : 2.67
실수 4 : 2.6667

```
#자료형의 출력
print(type(10))
print(type(10.5))
print(type('hello'))
```

<class 'int'>
<class 'float'>
<class 'str'>

```
a = 10
a.bit_length() #비트수
```

```
#숫자가 커질수록 필요한 비트수는 늘어난다.
a = 100000
a.bit_length()
```

```
#정수형은 크기 제한이 없다
a = 10 ** 1000
print(a)
print(a.bit_length())
```

```
# 실수형은 미세한 오차가 있다.
# 이진법으로 처리하기 어려운 부분이 있기 때문
b = 0.35
print(b)
print(b + 0.1)
```

```
#논리형 변수
hungry=True
sleepy=False
print(type(hungry))
print(not hungry)
print(hungry and sleepy)
print(hungry or sleepy)
```

```
<class 'bool'>
False
False
True
```

비교연산자

x < y x가 y보다 작다

x > y x가 y보다 크다

x == y x와 y가 같다

x != y x와 y가 같지 않다

x >= y x가 y보다 크거나 같다

x <= y x가 y보다 작거나 같다

x or y

x and y

not x

```
a=1
b=3
if a==1 and b==3:
    print("ok")
# ₩(백슬래시) - 코드가 길어질 경우 사용
if a==1 and ₩
b==3:
    print("ok")
```

ok

ok

```
# Multi Line Text
multiline = 'Hello₩nPython'
print(multiline)
# 작은따옴표 3개 또는 큰따옴표 3개를 쓰면  여러 라인에 걸쳐 문자열 작성 가능
multiline = '''Hello
Python'''
print(multiline)
```

```
Hello
Python
Hello
Python
```

```
# 문자열 연결
a = '안녕'
b ='하세요.'
print(a + b)
```

```
# 문자열과 숫자는+ 연산을 할 수 없음.
name = '김철수'
age = 20
print('이름 : ' + name + ', 나이:' + age)
```

```
name = '김철수'
age = 20

print('이름 : ' + name + ', 나이:' + str(age))
```

이름 : 김철수, 나이:20

```
#문자열 반복
a = '파이썬'
print(a*2)
print(a*3)

print('=' * 50)
print('My Program')
print('=' * 50)
```

파이썬파이썬
파이썬파이썬파이썬
==
My Program
==

```
# 인덱싱 - 문자의 위치
# 문자열의 인덱스는 0부터 시작함
a = '안녕하세요'
# 앞에서 네번째
print(a[3])
# 뒤에서 두번째
print(a[-2])
#슬라이싱 - 일부분을 잘라냄
# 0번에서 3번까지
print(a[0:4])
# 3번부터 끝까지
print(a[3:])
# 0번부터 6번까지
print(a[:7])
# 처음부터 끝까지
print(a[:])
```

```
# in, not in
a = "Hello world"
print("world" in a)
print("world" not in a)
```

```
True
False
```

```
#문자열 formatting

# % 위치에 숫자 대입
age = 20
print('당신의 나이는 %s세입니다.' % age)
# % 위치에 문자열 대입
fruit = '사과'
print('나는 %s를 좋아합니다.' % fruit)
# 2개 이상의 값 대입
name ='김철수'
print('%s님의 나이는 %d세입니다.' % (name, age))
# %를 쓰고 싶을 경우 %%로 두번 입력합니다.
rate = 98
print('정확도 : %d%%' % rate)

pi = 3.141592
print('원주율 : %.3f' % pi)
```

```
당신의 나이는 20세입니다.
나는 사과를 좋아합니다.
김철수님의 나이는 20세입니다.
에러율 : 98%
원주율 : 3.142
```

```
  #문자열 포매팅

  #숫자 대입
  age = 15
  result = "나이 : {0}세".format(age)
  print(result)

  #문자열 대입
  name = "김철수"
  result = "이름 : {0}".format(name)
  print(result)

  #2개 이상의 값 넣기
  result = "{0}님의 나이는 {1}세입니다.".format(name, age)
  print(result)

  #변수 이름으로 입력
  result = "{name}님의 나이는 {age}세입니다.".format(age=20,
name='김철수')
  print(result)
```

나이 : 15세
이름 : 김철수
김철수님의 나이는 15세입니다.
김철수님의 나이는 20세입니다.

```
  #왼쪽 정렬
  # 전체 자리수는 10으로 설정
  result = "{0:<10}".format("hi")
  print(result)

  #오른쪽 정렬
  result = "{0:>10}".format("hi")
  print(result)

  #가운데 정렬
  result = "{0:^10}".format("hi")
  print(result)

  #공백 채우기 - 공백문자 대신 =로 채움, 공백문자는 정렬 문자 앞에 넣어야 함
  result = "{0:=^10}".format("hi")
  print(result)

  result = "{0:-<10}".format("hi")
  print(result)
```

```
hi
        hi
    hi
====hi====
hi--------
```

```
#소수점 표현
y = 3.42134234
#소수 이하 네자리까지만 표현
result = "{0:.4f}".format(y)
print(result)
```

3.4213

```
# 문자 개수 세기
a = "hobby"
print( a.count('b') )
```

2

```
# 문자열의 위치
a = "Have a nice day."
print( a.find('nice') )
# 문자열을 찾지 못하면 -1을 리턴
# 대소문자를 구분함
print( a.find('have') )
```

7
-1

```
# 문자열의 위치
a = "Life is too short"
print( a.index('too') )
# 찾는 내용이 없으면 에러가 발생합니다.
print( a.index('kind') )
```

(결과)

```
8


---------------------------------------------------------------------------
ValueError                                Traceback (most
recent call last)
<ipython-input-16-972e2a710c5c> in <module>()
      3 print( a.index('too'))
      4# 찾는 내용이 없으면 에러가 발생합니다.
----> 5print( a.index('kind'))

ValueError: substring not found
```

```
# 문자열 삽입
# abcd 문자열의 각각의 문자 사이에 ,를 삽입합니다.
a= ","
result = a.join('abcd')
print(result)
```

a,b,c,d

```
# 소문자를 대문자로
a = "hi"
result = a.upper()
print(result)
# 대문자를 소문자로
a = "HI"
result=a.lower()
print(result)
#문장의 첫번째 문자를 대문자로
a = "python program"
result=a.capitalize()
print(result)
```

```
HI
hi
Web program
```

```
a = "    hi    "
# 왼쪽 공백 지우기(lstrip - Left Strip)
result=a.lstrip()
print(result)
# 오른쪽 공백 지우기(rstrip - Right Strip )
result=a.rstrip()
print(result)
# 양쪽 공백 지우기(strip)
result=a.strip()
print(result)
```

```
# 문자열 바꾸기(replace)
a = "신입 개발자"
result=a.replace("신입", "경력")
print(result)
```

경력 개발자

```
# 문자열 나누기(split) - 기본적으로 공백을 기준으로 나눔
a = "신입 개발자"
result=a.split()
print(result)

# 구분자를 지정할 수 있음
a = "a:b:c:d"
result=a.split(':')
print(result)
```

['신입', '개발자']
['a', 'b', 'c', 'd']

5. List, Tupple, Dictionary

5.1. 리스트(List)

임의의 객체를 순차적으로 저장하는 집합적 자료형
각 값에 대해 인덱스가 부여됨
변경 가능
대괄호 [] 사용
range() 함수를 통한 인덱스 리스트 생성 가능
range(k): 0부터 k-1까지의 숫자의 리스트를 반환함

5.2. 튜플(Tupple)

리스트와 유사하지만 튜플 내의 값을 변경할 수 없음
각 값에 대해 인덱스가 부여됨
변경 불가능
소괄호 () 사용

5.3. 사전(Dictionary)

정수형 인덱스가 아닌 키를 이용하여 값을 저장 및 조회하는 자료 구조

저장된 각 자료에 대한 순서는 의미 없음

중괄호 {} 사용

컴머(,)를 기준으로 아이템 구분

사전(Dictionary)의 원소 → key:value(쌍으로 이루어짐)

사전.keys( ) → key만 추출(임의의 순서)

사전.values( ) → value만 추출(임의의 순서)

사전.items( ) → key와 value를 튜플로 추출(임의의 순서)

keys, values, items가 반환하는 자료형 → 리스트

5.4. 자료형 요약 정리

자료형	변경가능여부	인덱스사용여부
수치형 상수	변경불가능(Immutable)	
문자열 상수	변경불가능(Immutable)	사용
List	변경가능(Mutable)	사용
Tupple	변경불가능(Immutable)	사용
Dictionary	변경가능(Mutable)	

5.5. 실습예제

```
# 리스트
# [] 대괄호로 감싸고 쉼표로 요소를 구분합니다.
a = [1, 3, 5, 7, 9]
print(a)
#첫번째 요소
print(a[0])
#뒤에서 첫번째 요소
print(a[-1])

a = [1, 2, 3, ['a', 'b', 'c']]
#마지막 요소(1차원 배열)의 첫번째 값
print(a[-1][0])
```

```
[1, 3, 5, 7, 9]
1
9
a
```

```
#리스트의 복사
a = [1, 2, 3, 4, 5]

#리스트 복사(얕은 복사)
b = a # 원본 리스트는 한개, 리스트 참조변수만 2개
print(id(a)) #주소값
print(id(b)
b[0] = 10
print('a:',a)
print('b:',b)



#리스트 복사(깊은 복사)
a = [1, 2, 3, 4, 5]
b = a[:] # 복사본 리스트가 생성됨
b[0] = 10
print('a:',a)
print('b:',b)
```

```
a: [10, 2, 3, 4, 5]
b: [10, 2, 3, 4, 5]
a: [1, 2, 3, 4, 5]
b: [10, 2, 3, 4, 5]
```

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

#리스트의 슬라이싱
print( a[0:2] )

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

# 리스트 반복
a = [1, 2, 3]
# 리스트 a를 3회 반복
b = a * 3
print(b)
```

```
[1, 2]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

```
# 숫자와 문자열 연결
a = [1, 2, 3]
print ( a[2] + "hi" ) # 에러
```

```
# 숫자와 문자열 연결
a = [1, 2, 3]
#숫자를 문자열로 바꾼 후 붙여야 함
print (str(a[2]) + "hi" )
```

3hi

```
# 리스트의 수정
a = [1, 2, 3]
a[1] = 20
print(a)
```

```
#리스트 삭제
a = [1, 2, 3]
a[1:3] = [ ]
print(a)

a = [1, 2, 3]
del a[1]
print(a)
```

[1]
[1, 3]

```
#리스트에 요소 추가(append)
a = [1, 2, 3]
a.append(4)
print(a)
#리스트를 추가할 수 있음
a.append([5,6])
print(a)
```

```
[1, 2, 3, 4]
[1, 2, 3, 4, [5, 6]]
```

```
#리스트의 정렬
a = [1, 4, 3, 2]
a.sort()
print(a)

a = ['a', 'c', 'b']
a.sort()
print(a)
```

```
[1, 2, 3, 4]
['a', 'b', 'c']
```

```
#리스트 뒤집기
a = ['a', 'c', 'b']
a.reverse()
print(a)
```

['b', 'c', 'a']

```
#인덱스 반환
a = [1,2,3]
print(a.index(3))
print(a.index(1))
#존재하지 않는 값을 입력하면 에러 발생
print(a.index(0))
```

2
0

```
#리스트에 요소 삽입(insert)
a = [1, 2, 3]
print(a)
#인덱스 0에 4 insert
a.insert(0, 4)
print(a)
```

[1, 2, 3]
[4, 1, 2, 3]

```
#리스트 요소 제거(remove)
a = [1, 2, 3, 1, 2, 3]
#첫번째 값만 제거합니다
a.remove(3)
print(a)
```

```
[1, 2, 1, 2, 3]
```

```
#리스트 요소 끄집어내기(pop)
#pop() : 리스트의 맨 마지막 요소를 돌려 주고 그 요소는 삭제

a = [1,2,3]
b=a.pop()
print(a)
print(b)
```

```
[1, 2]
3
```

```
#리스트에 포함된 요소 x의 개수 세기(count)
a = [1,2,3,1]
b=a.count(1)
print(b)
```

```
2
```

```
#리스트 확장(extend)
#extend(x)에서 x에는 리스트만 올 수 있으며 원래의 a 리스트에 x 리스트를 더하게 됩니다.
a = [1,2,3]
a.extend([4,5])
print(a)
```

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

```
#in과 not in
a = [1,2,3,4,5]

#a에 2가 포함되어 있는가?
result = 2 in a
print(result)

#a에 2가 포함되어 있지 않은가?
result = 2 not in a
print(result)
```

```
True
False
```

```
#튜플 자료형
#리스트는 대괄호[]로 감싸고 튜플은 소괄호 ()로 감싼다.
#리스트는 수정,삭제가 되지만 튜플은 수정,삭제가 안됨

t1 = (1, 2, 'a', 'b')
t1[0] = 'c'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-a594f0c53bbd> in <module>()
      4
      5 t1 =(1,2,'a','b')
----> 6t1[0]='c'

TypeError: 'tuple' object does not support item assignment
```

```
# 튜플은 삭제할 수 없다.
t1 = (1, 2, 'a', 'b')
del t1[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-21-cbb57ff166a2> in <module>()
      1# 튜플은 삭제할 수 없다.
      2 t1 =(1,2,'a','b')
----> 3del t1[0]

TypeError: 'tuple' object doesn't support item deletion
```

```
t1 = (1, 2, 'a', 'b')
print(t1)
print(t1[0])

print(t1[1:])

t2 = (3, 4)
print( t1 + t2 )

print( t2 * 3 )
```

```
(1, 2, 'a', 'b')
1
(2, 'a', 'b')
(1, 2, 'a', 'b', 3, 4)
(3, 4, 3, 4, 3, 4)
```

```
#튜플 풀기
t1 = (1,2,3)
#튜플의 값을 각각의 변수에 저장
n1,n2,n3 = t1
print(f'{n1} {n2} {n3}')

# 변수 값 교환하기
n1,n2=n2,n1
print(f'{n1} {n2}')
```

```
1 2 3
2 1
```

```
#튜플과 리스트의 변환
my_list = [1, 2, 3]
my_tuple = ('x', 'y', 'z')
#리스트를 튜플로 복사
print(tuple(my_list))
#튜플을 리스트로 복사
print(list(my_tuple))
```

```
(1, 2, 3)
['x', 'y', 'z']
```

```
#딕셔너리 자료형
# key 와 value를 한 쌍으로 갖는 자료형(자바의 map과 비슷함)
# {key: value}
a = {'name': '김철수'}
a['age'] = 20
print(a)
```

```
{name: '김철수', age: 20}
```

```
a = {'name':'김철수', 'phone':'01099933323', 'birth': '901231'}
print(a['name'])
print(a.get('name'))
print(a['phone'])
print(a.get('phone'))
print(a['birth'])
print(a.get('birth'))
```

```
#Value 리스트 만들기(values)
a = {'name':'김철수', 'phone':'01099933323', 'birth': '901231'}
b=a.values()
print(b)
```

```
dict_values(['김철수', '01099933323', '901231'])
```

```
  # 딕셔너리의 아이템
  a = {'name':'김철수', 'phone':'01099933323', 'birth': '901231'}
  # 튜플 형식의 키-값 쌍으로 구성된 리스트가 리턴됨
  print(a.items())
```

```
dict_items([('name', '김철수'), ('phone', '01099933323'), ('birth', '901231')])
```

```
  #해당 Key가 딕셔너리 안에 있는지 조사하기(in)
  a = {'name':'김철수', 'phone':'01099933323', 'birth': '901231'}
  b='name' in a
  print(b)
  b='email' in a
  print(b)
```

```
True
False
```

```
#집합 자료형
# 중복값을 허용하지 않음, 순서가 없음
s1 = set([1,2,3])
print(s1)
```

```
{1, 2, 3}
```

```
#교집합
s1 = set([1, 2, 3, 4, 5, 6])
s2 = set([4, 5, 6, 7, 8, 9])
s3 = s1 & s2
print(s3)
s3 = s1.intersection(s2)
print(s3)
#합집합
s4 = s1 | s2
print(s4)
s4 = s1.union(s2)
print(s4)
#차집합
s5 = s1 - s2
print(s5)
s5 = s1.difference(s2)
print(s5)
```

```
{4, 5, 6}
{4, 5, 6}
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{1, 2, 3}
{1, 2, 3}
```

```
#값 1개 추가하기(add)
s1 = set([1, 2, 3])
s1.add(4)
print(s1)
s1.add(4,5) #1개만 추가할 수 있음
print(s1)

#값 여러 개 추가하기(update)
s1 = set([1, 2, 3])
s1.update([4, 5, 6])
print(s1)
s1.update(4) #리스트만 추가할 수 있음
print(s1)
```

```
{1, 2, 3, 4}
{1, 2, 3, 4, 5, 6}
```

```
#특정 값 제거하기(remove)
s1 = set([1, 2, 3])
s1.remove(2)
print(s1)
```

```
{1, 3}
```

```
  #날짜
  # import 모듈 (해당 모듈 전체를 가져옴)
  # from 모듈 import method/변수 (해당 모듈 내에 있는 특정 method나 변수)
  from datetime import date, time, datetime, timedelta

  #today() : 오늘 날짜를 구하는 함수
  today = date.today()
  print("today: {0}".format(today))

  print("연도: {0}".format(today.year))
  print("월: {0}".format(today.month))
  print("일: {0}".format(today.day))
  # 날짜 시간 구하기
  current_datetime = datetime.today()
  print("날짜시간:{0}".format(current_datetime))
```

```
today: 2021-03-29
연도: 2021
월: 3
일: 29
날짜시간:2021-03-29 12:07:40.636269
```

```
  today = date.today()

  days = timedelta(days=-1)
  print("어제: {0!s}".format(today + days))

  days = timedelta(days=-7)
  print("1주 전: {0!s}".format(today + days))

  days = timedelta(days=-30)
  print("30일 전: {0!s}".format(today + days))

  hours = timedelta(hours=-8)
  print("{0!s}일 {1!s}초".format(hours.days, hours.seconds))
```

```
어제: 2021-03-28
1주 전: 2021-03-22
30일 전: 2021-02-27
-1일 57600초
```

```
  #날짜 출력 형식
  today = datetime.today()
  print("{}".format(today.strftime('%m/%d/%Y')))
  print("{}".format(today.strftime('%Y-%m-%d    %H:%M:%S'
)))
  # Y 4자리 연도, y 2자리 연도, H 24시간제, I 12시간제
  print("{}".format(today.strftime('%y-%m-%d %I:%M:%S')))
```

```
11/07/2021
2021-11-07 08:49:42
```

```
from datetime import datetime
 #문자열을 날짜 형식으로 변환
dateStr = '2021-04-15 12:23:38'
days=timedelta(days=-1)
print( dateStr + days ) #에러 발생(스트링은 날짜 연산을 할 수 없음)
myDate = datetime.strptime(dateStr, '%Y-%m-%d %H:%M:%S')
# 날짜 형식으로 바뀐 것 확인
print(type(myDate))
print(myDate)
days=timedelta(days=-1)
print( myDate + days ) #날짜 연산이 가능함
```

```
<class 'datetime.datetime'>
2021-04-15 12:23:38
```

```
import time

#유닉스 시간, 1970년 1월 1일을 기준으로 경과한 시간을 초단위로 표시
print(time.time())

#문자열로 표현한 시간
t = time.time()
print(time.ctime(t))
```

```
import datetime

#우리나라 시간
now = datetime.datetime.now()
print("%d년 %d월 %d일" % (now.year, now.month, now.day))
print("%d:%d:%d" % (now.hour, now.minute, now.second))
```

```
import time
#실행시간 측정
start = time.time()
for a in range(1000):
    print(a,end=" ")
end = time.time()
print()
print(end - start)
```

```
print("1")
time.sleep(1) #1초 멈춤
print("2")
time.sleep(5)
print("7")
```

```
import time

for dan in range(2, 10):
    print(dan, "단")
    for i in range(1, 10):
        print(dan, "*", i, "=", dan*i)
        time.sleep(0.2)
    print()
    time.sleep(1)
```

```
import calendar
#달력객체 리턴
#프린트 함수로 출력
print(calendar.calendar(2021))
print(calendar.month(2021, 11))
```

```
#달력을 직접 호출
calendar.prcal(2021)
calendar.prmonth(2021, 11)
```

```
import calendar
#요일 함수
yoil = ['월', '화', '수', '목', '금', '토', '일']
day = calendar.weekday(2021,11,12)
print(yoil[day] + "요일입니다." )
```

6. 콘솔입출력

6.1. 실습예제

```
#콘솔 입력 처리 함수
a = input("입력하세요:")
print("결과:",a)
```

입력하세요:안녕하세요
결과: 안녕하세요

```
#input 함수는 기본적으로 문자열로 입력받음
#eval() 함수 또는 int() 함수로 처리 가능
dan = input("단을 입력하세요:")
print( type(dan))
dan = int(input("단을 입력하세요:"))
print( type(dan))
```

단을 입력하세요:2
<class 'str'>
단을 입력하세요:3
<class 'int'>

```
dan = eval(input("단을 입력하세요:"))

for i in range(1, 10):
    print(dan,'x',i,'=',dan*i)
```

```
#아래 명령어를 입력한 후 c:/python/source/ex01.py로 저장합니다.
import sys
args = sys.argv[1:]
for i in args:
    print(i.upper(), end=' ')
#명령 프롬프트를 실행한 후 아래 명령어를 실행합니다.
# cd c:/python/source
# python ex01.py i love you
#            0      1  2   3
```

```
# {} 위치에 Kim이 입력되어 출력됨
print('Hello, {0}.'.format('Kim'))
```

Hello, Kim.

```
name = input('이름을 입력하세요:')
print('Hello, {0}.'.format(name))
```

이름을 입력하세요:Park

Hello, Park.

```
name = input('이름을 입력하세요:')
job = input('직업을 입력하세요:')
print('{0}의 직업은 {1}입니다.'.format(name,job))
```

이름을 입력하세요:김철호

직업을 입력하세요:프로그래머

김철호의 직업은 프로그래머입니다.

```
name = input('이름을 입력하세요:')
job = input('직업을 입력하세요:')
#출력 순서를 바꿀 수 있습니다.
print('{1}의 직업은 {0}입니다.'.format(job,name))
```

이름을 입력하세요:박철민
직업을 입력하세요:디자이너
박철민의 직업은 디자이너입니다.

```
name = input('이름을 입력하세요:')
job = input('직업을 입력하세요:')
#인덱스 대신 변수명을 쓸 수 있습니다.
print('{n}의 직업은 {j}입니다.'.format(j=job,n=name))
```

이름을 입력하세요:김철수
직업을 입력하세요:경찰
김철수의 직업은 경찰입니다.

```
import math
print('원주율:{0:f}'.format(math.pi))
#소수 이하 2자리
print('원주율:{0:.2f}'.format(math.pi))
print(f'원주율:{math.pi:.2f}')
```

원주율:3.141593
원주율:3.14

```
  num = 10
  a = '10진수 : {0:d}, 16진수(소문자) : {0:x}, 16진수(대문자) : {0:X}, 8진수 : {0:o}, 2진수 : {0:b}'.format(num)
  print(a)
  # 샵을 붙이면 접두사가 추가됨
  a = '10진수 : {0:d}, 16진수(소문자) : {0:#x}, 16진수(대문자) : {0:#X}, 8진수 : {0:#o}, 2진수 : {0:#b}'.format(num)
  print(a)
```

10진수 : 10, 16진수(소문자) : a, 16진수(대문자) : A, 8진수 : 12, 2진수 : 1010

10진수 : 10, 16진수(소문자) : 0xa, 16진수(대문자) : 0XA, 8진수 : 0o12, 2진수 : 0b1010

```
#천단위 컴머
num = 1234567890
print( '{0:,}'.format(num))
print( f'{num:,}')
#백분율
num=0.9815
print(f'{num:.0%}')
print(f'{num:.2%}')
```

1,234,567,890

98%

98.15%

7. 제어문

7.1. 들여쓰기와 제어문

파이썬은 들여쓰기를 강제하여 코드의 가독성을 높입니다.

블록 내부에 있는 문장들은 반드시 들여쓰기가 일치해야 합니다.

```
if a > 1:
    print 'a'
      print 'b' =>에러
```

블록의 시작은 콜론(:)이며 블록의 끝은 들여쓰기가 끝나는 부분으로 처리됩니다. - python에는 { }, begin, end 등의 키워드가 존재하지 않습니다.

들여쓰기를 할 때에는 탭과 공백을 섞어 쓰지 않습니다.

7.2. 조건문(if)

```
if 조건식1:
  statements
elif 조건식2:
  statements
elif 조건식3:
  statements
else:
   statements
```

조건식이나 else 다음에 콜론(:) 표기 필요
들여쓰기(indentation)를 잘 지켜야 함
switch 문은 없음
실행할 코드가 없는 경우 pass

```
  if not a:
       pass
```

7.3. 반복문

```
for <타겟> in <컨테이너 객체>:
    statements
else:
    statements
```

* for 루프의 중첩

```
for x in range(2, 10):
    print("=== {0}단 ===".format(x))
    for y in range(1, 10):
        print("{0} x {1} = {2:2d}".format(x, y, x*y))
    print() #빈 라인 출력
```

* while

조건식이 만족하는 동안 while 블록내의 문장들을 반복 수행합니다.

```
count = 1
while count < 11:
    print(count)
    count = count + 1
else:
    print('else block')
```

7.4. 실습예제

```
color = input('색상을 입력하세요: ')
if color == 'blue':
    print('건너가세요.')
else:
    print('기다리세요.')
```

색상을 입력하세요: blue
건너가세요.

```
num = 100
# 조건문 다음에 콜론(:)
# 탭을 쓰거나 공백4개를 씁니다.
if num % 2 == 0:
    print("짝수입니다")
else:
    print("홀수입니다")
```

짝수입니다

```
color = input('색상을 입력하세요: ')
if color == 'blue':
    print('건너가세요.')
elif color == 'red':
    print('기다리세요.')
else:
    print('잘못된 입력입니다.')
```

색상을 입력하세요: red
기다리세요.

```
for i in [1,2,3,4,5]:
    print(i)
```

1
2
3
4
5

```
numbers = [1,2,3,4,5]
for i in numbers:
    print(i)
```

1
2
3
4
5

```
#0부터 9
a = range(10)
print(type(a))
print(a)
a = list(range(10))
print(a)
#range(start, stop)
a = range(1, 11)
print(a)
a = list(range(1,11))
print(a)
#range(start, stop, step)
a = list(range(1,11,2))
print(a)
a = 0
for i in range(1, 11):
    a += i
print(a)
```

```
<class 'range'>
range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
range(1, 11)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 3, 5, 7, 9]
55
```

```
#다중 for문
for x in range(2, 5): #2~4단
    print(f'=== {x}단 ===')
    for y in range(1, 10):
        print(f'{x} x {y} = {x*y:2d}')
    print() #빈 라인 출력
```

```
=== 2단 ===
2 x 1 =  2
2 x 2 =  4
2 x 3 =  6
2 x 4 =  8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18

=== 3단 ===
3 x 1 =  3
3 x 2 =  6
3 x 3 =  9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
```

```
#리스트의 요소가 튜플인 경우
a = [(1,2), (3,4), (5,6)]
for (first, last) in a:
    print(f'{first}+{last}={first + last}')
```

```
1+2=3
3+4=7
5+6=11
```

```
points = [90, 25, 67, 45, 80]
number = 0
for point in points:
    number = number +1
    if point >= 60:
        result='합격'
    else:
        result='불합격'

    print(f'{number}번 학생: {point}점, {result}입니다.')
```

```
1번 학생 : 90점, 합격입니다.
2번 학생 : 25점, 불합격입니다.
3번 학생 : 67점, 합격입니다.
4번 학생 : 45점, 불합격입니다.
5번 학생 : 80점, 합격입니다.
```

```
# 딕셔너리 내부에 for 반복문 사용
a={ x: x**2 for x in (2,4,6)}
print(a)
```

```
{2: 4, 4: 16, 6: 36}
```

```
coffee = 5
money = 30000
while money:
    coffee = coffee - 1
    money = money - 3000
    print(f'남은 커피: {coffee}개, 잔액:{money}')
    if not coffee:
        print("품절")
        break
```

```
# continue 예제
# 홀수만 출력
a = 0
while a < 10:
    a = a+1
    if a % 2 == 0:
        continue
    print(a)
```

```
1
3
5
7
9
```

```
#무한 루프
while True:
    s = input('내용을 입력하세요(종료하려면 x 버튼을 누르세요): ')
    print(s)
    if s == 'x':
        break
```

```
내용을 입력하세요(종료하려면 x 버튼을 누르세요): aaa
내용을 입력하세요(종료하려면 x 버튼을 누르세요): bbb
내용을 입력하세요(종료하려면 x 버튼을 누르세요): x
```

```
while 1:
    dan = int(input('단을 입력하세요(종료하려면 0을 입력하세요):'))
    if dan==0:
        break
    for n in range(1, 10):
        print(dan, 'x', n, '=', dan*n)
```

```
단을 입력하세요(종료하려면 0을 입력하세요):2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
단을 입력하세요(종료하려면 0을 입력하세요):3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
단을 입력하세요(종료하려면 0을 입력하세요):0
```

```
#합계가 10만보다 커지게 되는 n을 구하기
x = 1
total = 0
while 1:
    total = total + x
    if total > 100000:
        print(x)
        print(total)
        break
    x = x + 1
```

* 난수 관련 예제

```
import random
#실수 난수
for i in range(5):
    print(random.random())

#정수 난수
for i in range(5):
    print(random.randint(1,10)) # 1~10

#1~100 사이의 실수 난수
for i in range(5):
    print(random.uniform(1,100))
```

```
# 정수 난수 5개 생성(중복값 제거)
myset=set()
while True:
    a=random.randint(1,10)
    print(a)
    myset.add(a)
    if len(myset)==5:
        break

myset
```

```
#리스트에서 임의의 요소를 하나 골라서 리턴
food = ["짜장면", "짬뽕", "탕수육", "군만두"]
f=random.choice(food)
print(f)
```

```
#무작위로 섞음
food = ["짜장면", "짬뽕", "탕수육", "군만두"]
print(food)
random.shuffle(food)
print(food)
```

```
# 4개 중에서 2개를 무작위로 샘플링
food = ["짜장면", "짬뽕", "탕수육", "군만두"]
print(random.sample(food, 2))
```

```
a = random.randint(1, 9)
b = random.randint(1, 9)
question = "%d + %d = ? " % (a, b)
c = int(input(question))

if c == a + b:
    print("정답입니다.")
else:
    print("틀렸습니다.")
```

```
#반복문을 사용한 버전
correct = 0
while True:
    a = random.randint(1, 9)
    b = random.randint(1, 9)
    question = "%d + %d = ?(끝낼 때는 0) " % (a, b)
    c = int(input(question))

    if c == 0:
        break
    elif c == a + b:
        print("정답입니다.")
        correct = correct + 1
    else:
        print("틀렸습니다.")

print("%d 개 맞췄습니다." % correct)
```

```
#연산자 선택
correct = 0
while True:
    a = random.randint(10, 99)
    b = random.randint(10, 99)
    op = random.randint(1, 3) #연산자 선택
    if op == 1:
        ans = a + b
        mark = "+"
    elif op == 2:
        if (a < b):
            a, b = b, a
        ans = a - b
        mark = "-"
    else:
        ans = a * b
        mark = "*"
    question = "%d %s %d = ?(끝낼 때는 0) " % (a, mar
k, b)
    c = int(input(question))
    if c == 0:
        break
    elif c == ans:
        print("정답입니다.")
        correct = correct + 1
    else:
        print("틀렸습니다.")

print("%d 개 맞췄습니다." % correct)
```

```
secret = random.randint(1,100)
count = 0
while True:
    num = int(input("숫자를 입력하세요(끝내려면 0) : "))
    if num == 0:
        break
    count += 1
    if num == secret:
        print("%d번만에 맞췄습니다" % count)
        break
    elif num > secret:
        print("더 작은 수를 입력하세요.")
    else:
        print("더 큰 수를 입력하세요.")
```

8. 함수

8.1. 함수의 정의

일정한 작업을 수행하는 코드 블록

8.2. 함수의 장점

반복적인 코드를 없애 주어 코드의 길이를 짧게 만들어 준다.
프로그램의 유지보수를 쉽게 만들어 준다.

8.3. 함수의 구조

```
def 함수명(입력매개변수):
    <수행할 문장1>
    <수행할 문장2>
    return 리턴값
```

함수의 몸체(body)에는 최소한 한개 이상의 문장이 필요함
그러므로, 아무런 내용이 없는 함수를 만들 때에는 pass 라는 키워드를 적어주어야 함
튜플을 사용하여 두 개 이상의 값을 동시에 반환하는 것처럼 처리할 수 있습니다.

8.4. 함수의 종류

내장함수
사용자정의함수

8.5. 실습예제(내장함수)

```
#절대값
print( abs(-3))
```

```
3
```

```
# all 함수, 모든 요소가 참이면 True, 아니면 False
print( all([1,2,3]) )
print( all([1,2,3,0]) )
# any 함수, 하나라도 참이면 True, 모두 거짓이면 False
print( any([0,1,2]) )
print( any([0,""]) )
```

```
True
False
True
False
```

```
# dir : 객체의 멤버(변수 또는 함수)
print('리스트 관련:', dir([1,2,3]))
print('딕셔너리 관련:', dir({'name':'kim'}))
```

```
# divmod(a, b) a를 b로 나눈 몫과 나머지를 튜플 형태로 리턴
print(divmod(7, 3))
print( divmod(1.3, 0.2) )
```

```
(2, 1)
(6.0, 0.09999999999999998)
```

```
# enumerate() 순서가 있는 자료형(리스트, 튜플, 문자열)을 입력받아
# 인덱스 값을 포함하는 enumerate 객체를 리턴
for i, num in enumerate([10,20,30,40,50]):
    print(i, num)
```

```
0 10
1 20
2 30
3 40
4 50
```

```
#eval(expression) : 실행가능한 문자열을 입력받아 문자열을 실행한 결과값을 리턴
print(eval('1+2'))
print(eval('divmod(4, 3)'))
```

```
3
(1, 1)
```

```
# hex(x) : 정수값을 입력받아 16진수(hexadecimal)로 변환하여 리턴
print(hex(234))
print(hex(3))
```

```
0xea
0x3
```

```
# id(object) : 객체를 입력받아 객체의 고유 주소값(레퍼런스)을 리턴
a = 3
print(id(3))
print(id(a))
b = a
print(id(b))
```

```
1630236192
1630236192
1630236192
```

```
# input([prompt]) : 사용자가 입력한 값을 받는 함수
a = input()
print(a)
b = input("입력하세요: ")
print(b)
```

```
10
10
입력하세요: 100
100
```

```
# int(x) : 문자열 형태의 숫자나 소수점이 있는 숫자 등을 정수 형태로 리턴
a=int('3')
print(a)
b=int(3.4)
print(b)
# int(입력값, 진법) : 입력된 값을 radix 진수로 표현
# 2진수 1111을 10진수로
c=int('1111',2)
print(c)
# 16진수 FF를 10진수로
d=int('FF', 16)
print(d)
```

```
3
3
15
255
```

```
# join 리스트를 특정 구분자를 포함하여 문자열로 변환
fruits = ['사과', '배', '수박', '바나나', '포도']
# 리스트의 각 요소와 쉼표 구분자를 연결
print (','.join(fruits))
print ('/'.join(fruits))
print ('₩n'.join(fruits))
```

```
# len(s) : 요소의 전체 개수
a=len("python")
print(a)
b=len([1,2,3])
print(b)
c=len((1, 'a'))
print(c)
```

```
6
3
2
```

```
# list(s) : 리스트를 리턴
a=list("python")
print(a)
b=list((1,2,3))
print(b)
```

```
['p', 'y', 't', 'h', 'o', 'n']
[1, 2, 3]
```

```
# map(function, iterable) : 집합의 각 요소가 함수에 의해 수행된 결과를 묶어서 리턴
def calc(x):
    return x*2
a=list(map(calc, [1, 2, 3, 4]))
print(a)
def plus(x):
    return x+1
print(list(map(plus, [1, 2, 3, 4, 5])))
```

```
[2, 4, 6, 8]
[2, 3, 4, 5, 6]
```

```
# max(iterable) : 최대값 리턴
c="김"
print(c)
print(ord(c)) # ord(문자) 문자코드
b="hello python"
for i in b:
    print("{0}==>{1}".format(i, ord(i)))

a=max([1, 2, 3])
print(a)
# 문자열에서는 문자코드값이 가장 큰 문자 리턴
b=max("python")
print(b)
```

```
3
y
```

```
# min(iterable) : 최소값 리턴
a=min([1, 2, 3])
print(a)
b=min("python")
print(b)
```

```
1
h
```

```
# pow(x,y) x를 y 거듭제곱한 결과 리턴
a=pow(2, 4)
print(a)
b=pow(3, 3)
print(b)
```

```
16
27
```

```
  # range([start,] stop [,step])
  #입력받은 숫자에 해당되는 범위의 값을 반복 가능한 객체로 만들어 리턴
  # 0부터 4까지
  a=list(range(5))
  print(a)
  # 5부터 9까지
  b=list(range(5, 10))
  print(b)
  # 1부터 9까지 2씩 증가
  c=list(range(1, 10, 2))
  print(c)
  # 0부터 -9까지 1씩 감소
  d=list(range(0, -10, -1))
  print(d)
```

```
[0, 1, 2, 3, 4]
[5, 6, 7, 8, 9]
[1, 3, 5, 7, 9]
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
```

```
# sorted(iterable) : 입력값을 정렬한 후 그 결과를 리스트로 리턴(원본은 정렬되지 않음)
a=sorted([3, 1, 2])
print(a)
b=sorted(['a', 'c', 'b'])
print(b)
c=sorted("zero")
print(c)
#기본내장함수 sorted()와 리스트 자료형의 sort() 함수의 차이점
# sort() 함수는 원본을 정렬하고 결과를 리턴하지 않음
items = [3,1,2]
#리턴값이 없기 때문에 result 변수에 None이 리턴됩니다.
result=items.sort()
print(result)
# items 리스트는 정렬이 되어 있습니다.
print(items)
```

```
[1, 2, 3]
['a', 'b', 'c']
['e', 'o', 'r', 'z']
None
[1, 2, 3]
```

```
# split : 문자열을 구분자를 기준으로 리스트로 변환
a = '사과,포도,배,복숭아,자두'
fruits = a.split(',')
print(fruits)
```

```
['사과', '포도', '배', '복숭아', '자두']
```

```
# strip() 문자열 좌우의 공백, 개행문자, 탭문자를 제거
a = '₩n₩t hi '
print(a)
b=a.strip()
print(b)
c=a.lstrip() #왼쪽 제거
print(c)
d=a.rstrip() #오른쪽 제거
print(d)
```

```
  hi
hi
hi

  hi
```

```
# str(object) : 문자열 형태로 객체를 변환하여 리턴
a=3
print( type(a))
b=str(3)
print( type(b))
```

```
<class 'int'>
<class 'str'>
```

```
('a', 'b', 'c')
```

```
#tuple(iterable) : 반복 가능한 자료형을 입력받아 튜플 형태로 바꾸어 리턴
print( tuple("abc") )
print( tuple([1, 2, 3]) )
```

```
(1, 2, 3)
```

```
# type(object) : 입력값의 자료형을 리턴
print(type('abc'))
print(type([ ]))
```

```
<class 'str'>
<class 'list'>
```

```
#zip(iterable) : 동일한 개수로 이루어진 자료형을 묶어 주는 역할
# 여러 재료를 놓고 김밥을 써는 것과 비슷한 개념
# 두개의 리스트를 같은 인덱스의 값끼리 묶음
names=['김철수','박철호','최정호','이철수','홍찬수']
points=[100,80,70,60,90]
# 같은 인덱스끼리 조합
for (name,point) in list(zip(names,points)):
    print('이름:',name,',점수:',point)
```

8.6. 실습예제(사용자정의함수)

```
#입력값과 리턴값이 없는 함수
def say():
    print('Hi')
say()
```

Hi

```
#입력값은 있지만 리턴값이 없는 함수
def hello(name):
    print('Hello,', name)
hello('kim')
#return 문장이 없거나 리턴값이 없을 경우 None을 반환합니다.
a = hello('kim')
print(a)
#None은 NoneType이라는 클래스의 값입니다.
print(type(None))
```

Hello, kim
Hello, kim
None
<class 'NoneType'>

```
  #아래와 같이 입력매개변수의 값이 없을 경우에 None을 활용하여 체크할 수 있음
  def hello2(name=None):
        if name==None:
              print('이름을 입력하세요')
              return
        print('Hello,', name)

  hello2()
  hello2('kim')
```

```
이름을 입력하세요
Hello, kim
```

```
#입력값이 없는 함수
def hello():
    return '안녕'
a = hello()
print(a)
```

안녕

```
#리턴값이 없는 함수
def mysum(a, b):
    print("%d + %d = %d" % (a, b, a+b))

mysum(3,4)
```

3 + 4 = 7

```
#입력값과 리턴값이 있는 함수
def mysum(a, b):
    return a+b

a = 3
b = 4
c = mysum(a, b)
print(c)
```

7

```
#가변사이즈 매개변수
#여러 개의 입력값을 받는 함수 만들기
# *변수 => 입력값들을 모두 모아서 튜플로 만들어준다.
def mysum(*args):
    print(args)
    result = 0
    for i in args:
        result = result + i
    return result
result = mysum(1,2,3)
print(result)
result = mysum(1,2,3,4,5,6,7,8,9,10)
print(result)
```

```
(1, 2, 3)
6
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
55
```

```
  def order(size, *toppings):
        print("₩n"+size + " 피자를 주문하셨습니다.₩n토핑:", end
='')
        for topping in toppings:
              print( topping+",",end='')

  order("large", '더블크러스트')
  order("medium", '파인애플', '버섯', '토마토')
```

large 피자를 주문하셨습니다.
토핑:더블크러스트,
medium 피자를 주문하셨습니다.
토핑:파인애플,버섯,토마토,

```
  # 함수의 리턴값은 1개이지만 튜플을 이용하여 여러개처럼 처리할
수 있습니다.
  def a():
        return 1,2,3,4,5
  a,b,c,d,e=a()
  print(a,b,c,d,e)
```

1 2 3 4 5

```
#튜플을 리턴하는 함수
def sum_and_mul(a,b):
    return a+b, a*b
result = sum_and_mul(3,4)
print(result)
#하나의 튜플값을 2개의 결과값으로 받고 싶을 경우
sum, mul = sum_and_mul(3, 4)
print("합:", sum)
print("곱:", mul)
```

```
(7, 12)
합: 7
곱: 12
```

```
#아래의 예제에서 두번째 return 문장은 실행되지 않습니다.
#return은 한번만 호출 가능합니다.
def sum_and_mul(a,b):
    return a+b
    return a*b
result = sum_and_mul(2, 3)
print("리턴값", result)
```

```
리턴값 5
```

```
#default 입력매개변수
def say(name, age=20):
    print("당신의 이름은 %s 입니다." % name)
    print("당신의 나이는 %d살입니다." % age)

#두번째 값이 빈값인 경우 default 값으로 처리됩니다.
say("김철수", 27)
say("김선영")
```

```
당신의 이름은 김철수 입니다.
당신의 나이는 27살입니다.
당신의 이름은 김선영 입니다.
당신의 나이는 20살입니다.
```

```
#default 입력매개변수는 마지막에 위치해야 함
def say(name='김철수',age ):
    print("당신의 이름은 %s 입니다." % name)
    print("당신의 나이는 %d살입니다." % age)

#첫번째 값이 비어있으므로 에러 발생
say(27)
```

```
  File "<ipython-input-18-4346d881c4ef>", line 2
    def say(name='김철수',age ):
                ^
SyntaxError: non-default argument follows default argume
nt
```

```
#지역변수와 전역변수
a = 1
def test(a):
    #입력매개변수 a는 지역변수입니다
    a = a +1
#전역변수의 값은 그대로 1입니다
test(a)
print(a)
a = 1
def test():
    #전역변수 a의 값이 바뀐다.
    global a
    a = a +1
test()
print(a)
```

```
1
2
```

```
#함수로 딕셔너리 반환
def set_person(name, job, age=0):
    person = {'name': name, 'job': job}
    #딕셔너리에 새로운 변수 추가
    if age:
        person['age'] = age
    #딕셔너리를 리턴
    return person
writer = set_person('김철수', '작가', 27)
print(writer)
```

```
{'name': '김철수', 'job': '작가', 'age': 27}
```

```
#함수에 리스트 전달
def greet(n):
    for name in n:
        msg = name + "님 환영합니다."
        print(msg)

names = ['김철호', '나철호', '송민수']
greet(names)
```

```
김철호님 환영합니다.
나철호님 환영합니다.
송민수님 환영합니다.
```

9. 클래스(class)

9.1. 클래스의 개요

모듈 : .py로 끝나는 파일
클래스 : 새로운 자료형을 만드는 방법
인스턴스 : 클래스로부터 만들어낸 객체

클래스의 형식

```
class 클래스이름:
    #Properties
    nation = "한국"
    name = str()
    age = int()
    #Methods
    def display(self):
        print('국적:',self.nation, end=' ')
        print(', 이름:',self.name, end=' ')
        print(', 나이:',self.age, end=' ')
```

* 클래스 내부의 method 선언 - def 키워드 사용
  일반 함수와 다른 점은 첫번째 인수로 self 사용
  self: 인스턴스 자신의 주소값을 가리킴
  각 인스턴스들은 self를 이용하여 자신의 이름 공간에 접근
* class method:
  인스턴스와 무관하게 클래스 이름 공간에 존재하는 method
  클래스 이름을 이용하여 호출함
  첫 인수로 클래스 인스턴스를 자동으로 받는 method

* __init__: 초기화함수
  인스턴스가 생성될 때 자동으로 호출되는 method
  self 인자가 정의되어야 함

* 상속(Inheritance)
  코드의 재사용
  자식 클래스는 상속을 해준 부모 클래스의 모든 기능을 그대로 사용
  자식 클래스는 필요한 기능만을 정의하거나 기존의 기능을 변경할 수 있음
  다중 상속 가능

* 오버라이드(Override)
  서브 클래스에서 슈퍼 클래스에 정의된 method를 재정의하여 사용하는 기능

* 다형성(Polymorphism)
  상속 관계 내의 다른 클래스들의 인스턴스들이 같은 멤버 함수 호출에 대해 각각 다르게 반응하도록 하는 기능
  적은 코딩으로 다양한 인스턴스들에게 유사한 작업을 수행시킬 수 있음
  코드의 가독성 향상

9.2. 실습예제

```
a=1
print(type(a))
b='a'
print(type(b))
```

원둘레와 원넓이 계산 예제

```
import math

class MyCircle: #새로운 클래스(사용자정의 자료형)
    a=0
    b=0
    r=0
    def set_r(self,r): #반지름을 입력하는 함수
        # self.변수 => 클래스 전체에서 사용할 수 있는 변수
        # self가 붙지 않은 변수 => 지역변수
        self.r=r

    def length(self): #둘레를 계산하는 함수
        self.a=2 * math.pi * self.r

    def area(self): #넓이를 계산하는 함수
        self.b=math.pi * self.r * self.r

    def disp(self): #출력함수
        print(f'반지름:{self.r},둘레:{self.a:.2f},넓이:{self.b:.2f}')
```

```
  a=MyCircle() # 클래스의 인스턴스를 만들고 참조변수 a에 주소 전달
  a.set_r(50) # a 변수가 가리키는 인스턴스의 set_r 함수 호출
  a.length()
  a.area()
  a.disp()
```

신상정보 클래스 예제

```
#변수들만으로 구성된 클래스
class Person:
    name=''
    age=0
    height=0
```

```
# 클래스이름() => 클래스의 인스턴스를 메모리에 생성
p1=Person()
p1.name='김철수'
p1.age=20
p1.height=187.7
print( f'이름:{p1.name}, 나이:{p1.age}, 키:{p1.height}' )
```

```
class Person:
    name=''
    age=0
    height=0
    #함수가 추가됨, 클래스의 멤버함수(method)
    #첫번째 매개변수에는 self를 써야 함
    #self : 현재 인스턴스의 시작번지값을 가리킴
    #self.변수 : 클래스의 멤버변수
    def disp(self):
        print( f'이름:{self.name}, 나이:{self.age}, 키:{self.heigh
t}')
```

```
p2=Person()
p2.name='김철수'
p2.age=-50
p2.height=187.7
p2.disp() #함수를 호출함
```

```
class Person:
    name=''
    age=0
    height=0
#변수에 직접 값을 입력할 경우 오류가 발생할 수 있음
    def set_age(self,age):
        if age>100 or age<0:
            self.age=int(input('나이를 정확히 입력하세요:'))
        else:
            self.age=age

    def disp(self):
        print( f'이름:{self.name}, 나이:{self.age}, 키:{self.heigh
t}')
```

```
p3=Person()
p3.name='김철수'
p3.set_age(-100)
p3.height=187.7
p3.disp() #함수를 호출함
```

사원정보 클래스 예제

```
class Emp:
    name=''
    salary=0
    bonus=0
    money=0
```

```
e1=Emp() #인스턴스 생성
e1.name='김철수' # 멤버변수에 값을 입력함
e1.salary=500
e1.bonus=300
e1.money=e1.salary*12 + e1.bonus
print( f'이름:{e1.name}, 월급:{e1.salary}, 보너스:{e1.bonus}, 연봉:{e1.money}' )
```

```
class Emp:
    name=''
    salary=0
    bonus=0
    money=0
    #출력함수
    def disp(self):
        print( f'이름:{self.name}, 월급:{self.salary}, 보너스:{self.bonus}, 연봉:{self.money}' )
```

```
e2=Emp() #인스턴스 생성
e2.name='김철수' # 멤버변수에 값을 입력함
e2.salary=500
e2.bonus=300
e2.money=e1.salary*12 + e1.bonus
e2.disp()
```

```
class Emp:
    name=''
    salary=0
    bonus=0
    money=0
    #초기화 함수(인스턴스를 만들 때 자동으로 호출되는 함수)
    def __init__(self,name,salary,bonus):
        #print('초기화 함수가 호출되었습니다.')
        self.name=name
        self.salary=salary
        self.bonus=bonus

    #출력함수
    def disp(self):
        print( f'이름:{self.name}, 월급:{self.salary}, 보너스:{sel
f.bonus}, 연봉:{self.money}' )
```

```
#인스턴스 생성, 괄호 안의 값들이 init 함수에 전달됨
e3=Emp('김철수',500,700)
e3.money=e3.salary*12 + e3.bonus
e3.disp()
```

```
class Emp:
    name=''
    salary=0
    bonus=0
    money=0
    #초기화 함수(인스턴스를 만들 때 자동으로 호출되는 함수)
    def __init__(self,name,salary,bonus):
        #print('초기화 함수가 호출되었습니다.')
        self.name=name
        self.salary=salary
        self.bonus=bonus
    #연봉계산함수
    def calc(self):
        self.money=self.salary*12 + self.bonus
    #출력함수
    def disp(self):
        print( f'이름:{self.name}, 월급:{self.salary}, 보너스:{self.bonus}, 연봉:{self.money}' )
```

```
e4=Emp('김철수',500,700)
e4.calc() #계산함수 호출
e4.disp()
```

점수계산 클래스 예제

```
class Point:
    name=''
    kor=0
    eng=0
    mat=0
    tot=0
    avg=0
    grade=''
```

```
  p1=Point()  #인스턴스 생성
  p1.name='김철수'
  p1.kor=80
  p1.eng=90
  p1.mat=95
  p1.tot=p1.kor+p1.eng+p1.mat
  p1.avg=p1.tot / 3
  if 90<= p1.avg <= 100:
       p1.grade='A'
  elif 80<= p1.avg < 90:
       p1.grade='B'
  elif 70<= p1.avg < 80:
       p1.grade='C'
  elif 60<= p1.avg < 70:
       p1.grade='D'
  else:
       p1.grade='F'

  print('이름₩t국어₩t영어₩t수학₩t총점₩t평균₩t등급')
  print(  f'{p1.name}₩t{p1.kor}₩t{p1.eng}₩t{p1.mat}₩t{p1.tot}
₩t{p1.avg:.1f}₩t{p1.grade}' )
```

```
class  Point:
    name=''
    kor=0
    eng=0
    mat=0
    tot=0
    avg=0
    grade=''
    #초기화함수
    def  __init__(self,name,kor,eng,mat):
        self.name=name
        self.kor=kor
        self.eng=eng
        self.mat=mat
```

```
  p2=Point('김철수',88,99,100)  #인스턴스 생성
  p2.tot=p2.kor+p2.eng+p2.mat
  p2.avg=p2.tot / 3
  if 90<= p2.avg <= 100:
      p2.grade='A'
  elif 80<= p2.avg < 90:
      p2.grade='B'
  elif 70<= p2.avg < 80:
      p2.grade='C'
  elif 60<= p2.avg < 70:
      p2.grade='D'
  else:
      p2.grade='F'
  print('이름₩t국어₩t영어₩t수학₩t총점₩t평균₩t등급')
  print(  f'{p2.name}₩t{p2.kor}₩t{p2.eng}₩t{p2.mat}₩t{p2.tot}
₩t{p2.avg:.1f}₩t{p2.grade}' )
```

```
class Point:
    def __init__(self,name,kor,eng,mat):
        self.name=name
        self.kor=kor
        self.eng=eng
        self.mat=mat
    def calc(self):
        self.tot=self.kor+self.eng+self.mat
        self.avg=self.tot / 3
        if 90<= self.avg <= 100:
            self.grade='A'
        elif 80<= self.avg < 90:
            self.grade='B'
        elif 70<= self.avg < 80:
            self.grade='C'
        elif 60<= self.avg < 70:
            self.grade='D'
        else:
            self.grade='F'
```

```
p3=Point('김철수',88,99,100)  #인스턴스 생성
p3.calc()
print('이름₩t국어₩t영어₩t수학₩t총점₩t평균₩t등급')
print(  f'{p3.name}₩t{p3.kor}₩t{p3.eng}₩t{p3.mat}₩t{p3.tot}
₩t{p3.avg:.1f}₩t{p3.grade}' )
```

```
class Point:
    def __init__(self,name,kor,eng,mat):
        self.name=name
        self.kor=kor
        self.eng=eng
        self.mat=mat
    def calc(self):
        self.tot=self.kor+self.eng+self.mat
        self.avg=self.tot / 3
        if 90<= self.avg <= 100:
            self.grade='A'
        elif 80<= self.avg < 90:
            self.grade='B'
        elif 70<= self.avg < 80:
            self.grade='C'
        elif 60<= self.avg < 70:
            self.grade='D'
        else:
            self.grade='F'
    def disp(self):
        print('이름₩t국어₩t영어₩t수학₩t총점₩t평균₩t등급')
        print( f'{self.name}₩t{self.kor}₩t{self.eng}₩t{self.mat}₩t{self.tot}₩t{self.avg:.1f}₩t{self.grade}' )
```

```
p4=Point('김철수',88,99,100)  #인스턴스 생성
p4.calc()
p4.disp()
```

클래스의 상속

```
class Book:
    def __init__(self,name,price):
        self.name=name
        self.price=price
    #출력함수
    def disp(self):
        #print('도서명₩t단가')
        print(f'{self.name}₩t{self.price}')
```

```
class BookSale(Book):
    def __init__(self,name,price,amount):
        Book.__init__(self,name,price)
        self.amount=amount
    #계산함수
    def calc(self):
        self.money=self.price*self.amount
        if 1<=self.amount<=1000:
            self.fee=self.money * 0.01
        elif 1001<=self.amount<=3000:
            self.fee=self.money * 0.03
        elif 3001<=self.amount:
            self.fee=self.money * 0.05
    #출력함수
    def disp(self):
        #print('도서명₩t단가₩t수량₩t금액₩t₩t인세')
        print(f'{self.name}₩t{self.price}₩t{self.amount}₩t{sel
f.money}₩t{self.fee}')
```

```
b1=BookSale('Java',50000,1000)
b1.calc()
b1.disp()

b2=BookSale('C',40000,2500)
b2.calc()
b2.disp()

b3=BookSale('Python',35000,3500)
b3.calc()
b3.disp()
```

10. 패키지와 모듈

10.1. 모듈(module)

* 함수 : 파일 내에서 일정한 작업을 수행하는 코드 블록

* 모듈 : 함수나 변수 또는 클래스들을 모아 놓은 파이썬 소스 코드 파일

비슷하거나 관련된 일을 하는 함수나 상수들을 모아서 하나의 파일에 저장하고 추후에 재사용하기 위해 사용

모듈 이름은 py 확장자를 제외한 파일이름

* 패키지 - 파이썬 모듈들을 모아놓은 디렉토리

모듈 = 파일, 패키지 = 디렉토리

10.2. 모듈을 사용하는 목적

코드의 재사용 : 자주 사용되는 함수를 매번 작성하지 않고 import 하여 사용할 수 있음

10.3. 모듈의 종류

표준 모듈

파이썬 언어 패키지 안에 기본적으로 포함된 모듈

대표적인 표준 모듈 예 : math, string

사용자 정의 모듈 : 개발자가 직접 정의한 모듈

10.4. 모듈의 선언

* 모듈 전체를 참조

import 모듈

import 모듈명 as 모듈별칭

해당 모듈을 짧은 이름으로 사용하고자 할 때 사용

기존 모듈 이름이 너무 길거나 현재 사용 중인 다름 이름들과 충돌이 일어날 때 유용

* 모듈 내에서 필요한 부분만 참조

from 모듈 import 변수 or 함수

모듈 이름을 붙이지 않고 바로 해당 모듈의 함수를 쓸 수 있습니다.

from 모듈명 import *

해당 모듈에 속한 '__'로 시작되는 함수,변수들을 제외한 모든 멤버들을 참조

from 모듈명 import 이름 as 별칭

해당 모듈 내에 정의된 이름을 짧은 이름으로 사용하고자 할 때 사용

* 모듈 내부의 함수 호출 방법

모듈명.함수()

모듈을 참조할 수 있는 디렉토리 확인

기본적으로 현재 디렉토리에 있는 파일이나 path에 설정된 파이썬 라이브러리가 저장된 디렉토리에 있는 모듈만 불러올 수 있습니다.

import sys 후 sys.path 로 확인 가능

sys.path.append(), sys.path.remove()로 추가, 삭제 가능

10.5. 패키지(package)

여러개의 파이썬 모듈 파일들을 모아 놓은 디렉토리

모듈 : 파일, 패키지 : 디렉토리

__init__.py 파일

해당 디렉토리가 패키지임을 알려주는 역할을 하는 파일

(파이썬 3.3 버전부터는 __init__.py 파일이 없어도 패키지로 인식 가능함

3.3 이전의 버전에서는 디렉토리에 __init__.py 파일이 없으면 패키지로 인식되지 않음. 하위 버전 호환성을 위해 __init__.py 파일을 생성하는 것이 좋음)

10.6. 실습예제

* 파이참 설치

PyCharm - Python 전용 개발툴(IntelliJ 기반)

https://www.jetbrains.com/ko-kr/pycharm/

* 프로젝트 생성 : myproject1

디렉토리 : c:/python/myproject1

* 패키지 추가 : test, common

* common 패키지에 mod1.py

```
def gugu(a):
    for i in range(1,10):
        print( f'{a} x {i} = {a*i}' )
```

* common 패키지에 mod2.py

```
def mysum(a):
    result=0
    for i in range(1,a+1):
        result += i
    return result
```

* test 패키지 ex01.py

```
# import
import common.mod1 as m1
import common.mod2 as m2

m1.gugu(5)
b=m2.mysum(10)
print(b)
```

* test 패키지 ex02.py

```
#실행 : Alt+Shift+F10
# __로 시작하는 멤버를 제외하고 import
# from ~ import
from common.mod1 import *
gugu(9)

from common import mod2
a=mod2.mysum(100)
print(a)

from common.mod2 import mysum
b=mysum(200)
print(b)
```

* test 패키지 ex03.py

```
#sys 라이브러리를 import합니다.
import sys
# 파이썬 라이브러리가 설치되어 있는 디렉토리 확인
print(sys.path)
# 파이썬 라이브러리 디렉토리를 추가합니다.
sys.path.append("c:/python/source")
print(sys.path)
```

* common/mod1.py 에 추가

```
#common/mod1.py 파일에 아래 코드를 추가합니다.
def gugu(a):
    for i in range(1,10):
        print('{} x {} = {}'.format(a,i,a*i))

if __name__ == "__main__":
    gugu(5)
```

if __name__ == "__main__" 을 사용하면 python mod1.py 와 같이 명령 프롬프트상에서 실행시킬 경우에는 main 부분이 자동으로 실행됩니다.

__name__ 변수는 파이썬이 사용하는 특별한 변수명

명령 프롬프트에서 실행할 경우 __name__ 변수에 __main__ 이라는 값이 할당됨

* common/mod2.py 에 추가

```
def mysum(a):
    result = 0
    for i in range(1, a + 1):
        result += i
    return result
if __name__ == "__main__":
    print(mysum(100))
```

11. 파일입출력

11.1. 파일 오픈

open(filename, mode)

Mode	설명
r	읽기 전용
w	쓰기 전용
a	파일 끝에 추가
rb	이진 파일 읽기 전용
wb	이진 파일 쓰기 전용
ab	이진 파일 끝에 추가

11.2. 파일 닫기

모든 작업이 끝나면 close()를 호출하여 작업 프로세스의 자원 점유 해제

close()을 마지막에 호출하지 않으면 해당 file 객체가 다른 값으로 치환되거나 프로그램이 종료될 때 자동으로 close()가 호출됨

하지만 명시적으로 close()를 호출하는 것을 권장함

11.3. 파일 접근 방법

순차 접근(기본 방식) : 파일을 앞에서부터 순차적으로 읽고 쓰는 방식

임의 접근 : 파일 내 임의 위치에서 읽고 쓰는 방식

임의 접근을 위한 file 객체 포인터 (pointer) 관련 메소드

seek(n): 파일의 가장 첫번째 위치에서 n번째 바이트로 포인터 이동

tell(): 파일 내 현재 포인터 위치를 반환

11.4. 실습

```
#현재 실행중인 디렉토리에 파일 생성
# open(filename, mode) : 파일 오픈
# mode : w 쓰기모드, r 읽기모드, a 추가 모드, b 바이너리 모드
# f = open("binary_file", "rb")  바이너리 읽기 모드
f = open("file1.txt", 'w')
f.close()
```

```
#지정된 디렉토리에 파일 생성
f = open("c:/python/source/file2.txt", 'w')
f.close()
```

```
#파일에 내용을 출력
f = open("c:/python/source/file3.txt", 'w')
for i in range(1, 11):
    data = f'{i}번째 줄입니다.₩n'
    f.write(data)
f.close()
```

```
#파일 내용 읽기
f = open("c:/python/source/file3.txt")
while True:
    line = f.readline()
    if not line: break
    print(line,end='')
f.close()
```

```
1번째 줄입니다.
2번째 줄입니다.
...
9번째 줄입니다.
10번째 줄입니다.
```

```
# csv 형식으로 저장하는 방법
#실행할 때마다 텍스트 파일에 내용이 추가됩니다.
nums = list(range(10))
cnt = len(nums)
output_file = "c:/python/source/result.csv"
f = open(output_file, 'a')
for idx in range(cnt):
    if idx < (cnt-1):
        f.write(str(nums[idx])+',')
    else:
        f.write(str(nums[idx])+'₩n')
f.close()
print("저장되었습니다.")
```

```
f = open("c:/python/source/file3.txt")
#파일의 모든 라인을 읽어서 각각의 라인을 요소로 갖는 리스트 리턴
lines = f.readlines()
print(lines)
for line in lines:
    print(line, end='')
f.close()
```

['1번째 줄입니다.₩n', '2번째 줄입니다.₩n', '3번째 줄입니다.₩n', '4번째 줄입니다.₩n', ... '15번째 줄입니다.₩n', '16번째 줄입니다.₩n', '17번째 줄입니다.₩n', '18번째 줄입니다.₩n', '19번째 줄입니다.₩n']

1번째 줄입니다.
2번째 줄입니다.
...
18번째 줄입니다.
19번째 줄입니다.

```
f = open("c:/python/source/file3.txt")
#파일 내용 전체를 읽어들임
data = f.read()
print(data)
f.close()
```

```
#파일 내용 읽기(간단한 방법)
f = open("c:/python/source/file3.txt", 'r')
for line in f:
    print(line, end='')
f.close()
```

```
1번째 줄입니다.
2번째 줄입니다.
...
14번째 줄입니다.
15번째 줄입니다.
16번째 줄입니다.
17번째 줄입니다.
18번째 줄입니다.
19번째 줄입니다.
```

```
# with  블록을 벗어나는 순간 파일이 자동으로 닫힌다.
with open("c:/python/source/file4.txt", "w") as f:
    f.write("hello python")
```

```
f = open("c:/python/source/file3.txt", 'r')
for line in f:
    print(line, end='')
#다시 첫번째 라인으로 이동
#f.seek(offset, from_what)
#from_what : 0 파일의 시작 위치(기본값), 1 현재 위치, 2 파일의 끝 위치(1의 경우 텍스트파일에서는 0만 허용되고 이진파일에서는 0 이외의 숫자도 가능함)
f.seek(0,0)
line=f.readline()
print(line)
#파일 포인터의 위치 확인
print(f.tell())
f.close()
```

```
    1번째 줄입니다.
    2번째 줄입니다.
    3번째 줄입니다.
    4번째 줄입니다.
    5번째 줄입니다.
...
    15번째 줄입니다.
    16번째 줄입니다.
    17번째 줄입니다.
    18번째 줄입니다.
    19번째 줄입니다.
    1번째 줄입니다.

    17
```

```
#파일의 내용을 리스트로 저장
f = open("c:/python/source/file3.txt", 'r')
f_list = list(f)
print(f_list)
f.close()
```

['1번째 줄입니다.₩n', '2번째 줄입니다.₩n', '3번째 줄입니다.₩n', '4번째 줄입니다.₩n', '5번째 줄입니다.₩n', '6번째 줄입니다.₩n', '7번째 줄입니다.₩n', '8번째 줄입니다.₩n', '9번째 줄입니다.₩n', '10번째 줄입니다.₩n', '11번째 줄입니다.₩n', '12번째 줄입니다.₩n', '13번째 줄입니다.₩n', '14번째 줄입니다.₩n', '15번째 줄입니다.₩n', '16번째 줄입니다.₩n', '17번째 줄입니다.₩n', '18번째 줄입니다.₩n', '19번째 줄입니다.₩n']

```
filename = 'c:/data/text/pi_million_digits.txt'
with open(filename) as f1:
    lines = f1.readlines()

for line in lines:
    print(line,end='')
```

```
a = "i love you, you love me"
# 문자열.count("검색어") 검색어를 찾은 횟수
count=a.count("you")
print(count)
```

2

```
# pi를 소수점 아래 백만자리까지 저장한 텍스트 파일을 로딩
filename='c:/data/text/pi_million_digits.txt'
with open(filename) as f:
    lines = f.readlines()

print(len(lines))  #리스트의 요소 개수
print(lines[0]) #리스트의 첫번째 요소 출력

pi_string = ''
for line in lines:
    pi_string += line.strip()
num = input('숫자를 입력하세요: ')
if num in pi_string:
    total=pi_string.count(num)
    print(num+'숫자를 pie 문자열에서',total,'회 찾았습니다.')
else:
    print(num+'숫자를 찾지 못했습니다.')
```

```
  #파일압축
  from zipfile import *

  def compressZip(zipname, filename):
        print('[%s] -> [%s] 압축...' %(filename, zipname))
        with ZipFile(zipname, 'w') as ziph: #쓰기 모드로 오픈
             ziph.write(filename,compress_type=ZIP_DEFLATED)
#파일압축

        print('압축이 끝났습니다.')

  filename = 'mydata.txt'
  zipname = filename + '.zip'
  compressZip(zipname, filename)
```

```
  #디렉토리 압축
  from zipfile import *
  import os

  def compressAll(zipname, folder):
      print('[%s] -> [%s] 압축...' %( folder, zipname))
      with ZipFile(zipname, 'w') as ziph:
          # 디렉토리이름,[하위디렉토리리스트],[파일이름리스트]
          # os.walk(디렉토리) : 디렉토리 탐색
          # 하위디렉토리들을 반복적으로 탐색
          for dirname, subdirs, files in os.walk(folder):
              print(dirname,subdirs,files)
              for file in files:
                  ziph.write(os.path.join(dirname, file),compre
ss_type=ZIP_DEFLATED)

  folder = 'c:/temp'
  zipname = folder + '.zip'
  compressAll(zipname, folder)
```

```
#압축 풀기
from zipfile import *
import os
#현재 작업중인 디렉토리 변경
os.chdir('c:/python/source')

def extractZip(zipname):
    with ZipFile(zipname, 'r') as ziph:
        ziph.extractall()
        print('[%s]가 성공적으로 추출되었습니다.' %zipname)

extractZip('c:/temp.zip')
```

```
#웹서버 로그
#총페이지뷰 수 계산
pageviews = 0

with open('c:/data/text/access_log', 'r') as f:
    logs = f.readlines()
    for log in logs:
        log = log.split() #공백으로 분리
        status = log[8] #9번째
        if status == '200':
            pageviews += 1

print( f'총 페이지뷰: {pageviews}' )
```

12. 예외처리

12.1. 에러와 예외

에러 (Syntax Error)

문법적 에러

파이썬은 상대적으로 문법이 간단하기 때문에 구문 자체의 에러 발생 비율이 낮음

예외 (Exception)

구문 에러는 없으나 프로그램 실행 중 더 이상 진행할 수 없는 상황

예외가 발생하면 프로그램은 바로 종료됨

12.2. 예외 처리 방법

```
try:
  (예외 발생 가능한) 문장들
except Exception:
  예외가 발생했을 때 실행되는 문장들
else:
  예외가 발생하지 않았을 때 실행되는 문장들
finally:
  예외 발생 유무와 관계없이 항상 실행되는 문장들
```

12.3. 파이썬 내장 예외의 종류

https://docs.python.org/3/library/exceptions.html

12.4. 실습

```
#예외 - 코드 실행 중에 에러가 발생하는 경우
4 / 0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-31-1c883440cb28> in <module>()
      1 #예외 - 코드 실행 중에 에러가 발생하는 경우
----> 2 4 / 0
ZeroDivisionError: division by zero
```

```
try:
    4 / 0
except ZeroDivisionError as e:
    print(e)
```

```
division by zero
```

```
print("종료하려면 q를 입력하세요.")
while True:
    num1 = input("₩n분자: ")
    if num1 == 'q':
        break
    num2 = input("분모: ")
    try:
        result = int(num1) / int(num2)
    except ZeroDivisionError:
        print("분모에 0이 올 수 없습니다.")
    else:
        print(result)

print("프로그램을 종료합니다.")
```

```
종료하려면 q를 입력하세요.
분자: 3
분모: 2
1.5
분자: q
프로그램을 종료합니다.
```

```
def exception_test():
    print('start')
    try:
        print( 2 + '2')
    except TypeError as e:
        #자세한 에러 메시지 출력
        print(f'TypeError : {e}')
    print('end')
exception_test()
```

```
import traceback
def exception_test():
    print('start')
    try:
        print( 2 + '2')
    except :
        #트레이스백 메시지를 출력
        traceback.print_exc()
    print('end')

exception_test()
```

```
#텍스트파일을 작성한 후 실습
try:
    f = open('c:/python/temp/test.txt', 'r')
except FileNotFoundError as e:
    print(e)
else: #예외가 발생하지 않을 경우 실행되는 코드
    while True:
        line = f.readline()
        if not line: break
        print(line,end='')
    f.close()
```

```
111
222
333
444
555
```

```
  #예외 처리가 되지 않았으므로 파일 경로가 잘못될 경우 예외가 발생합니다.
  filename = 'c:/data/text/alice.txt'
  with open(filename, encoding='utf-8') as file:
        contents = file.read()
        words = contents.split()
        num = len(words)
        print(f'파일이름: {filename}, 단어개수: {num}' )
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-36-fff9540eab42> in <module>()
      2 filename = 'c:/python/soure/alice.txt'
      3
----> 4 with open(filename, encoding='utf-8') as file:
      5     contents = file.read()
      6     words = contents.split()

FileNotFoundError: [Errno 2] No such file or directory: 'c:/python/soure/alice.txt'
```

```
#예외 처리한 코드
filename = 'c:/data/text/alice.txt'
try:
    with open(filename, encoding='utf-8') as file:
        contents = file.read()
except FileNotFoundError as e:
    print(e)
    msg = "파일을 찾을 수 없습니다."
    print(msg)
else:
    #공백을 기준으로 나눔
    words = contents.split()
    num = len(words)
    print(f'파일이름: {filename}, 단어수: {num}' )
```

[Errno 2] No such file or directory: 'c:/python/soure/alice.txt'

파일을 찾을 수 없습니다.

```
# 다중 예외처리
try:
    a = [1,2]
    4/0 #산술연산 오류
    print(a[2]) #인덱스 오류

except ZeroDivisionError:
    print("0으로 나눌 수 없습니다.")
except IndexError:
    print("인덱싱 할 수 없습니다.")
except Exception:
    print("기타 예외입니다.")

print("종료")
```

0으로 나눌 수 없습니다.

```
try:
    f = open('c:/python/temp/test.txt', 'r')
except FileNotFoundError as e:
    print(e)
else: #예외가 발생하지 않을 경우 실행되는 코드
    while True:
        line = f.readline()
        if not line: break
        print(line,end='')
    f.close()
finally:
    print('종료되었습니다.')
```

```
#여러 파일 다루기
def count_words(filename):
    try:
        with open("c:/data/text/"+filename) as f:
            contents = f.read()
    except:
        pass
    else:
        words = contents.split()
        num_words = len(words)
        print( f'파일명: {filename} , 단어수 : {num_words}' )

filenames = ['alice.txt', 'moby_dick.txt', 'little_women.txt']
for filename in filenames:
    count_words(filename)
```

```
파일명: alice.txt , 단어수 : 29461
파일명: moby_dick.txt , 단어수 : 215136
파일명: little_women.txt , 단어수 : 189079
```

13. 정규표현식

13.1. 유효성 검증과 정규표현식

사용자가 입력한 값이 형식에 맞는지 검증하는 것을 유효성 검증이라고 합니다.

유효성 검증을 위한 표현식을 정규표현식(Regular Expression)이라고 합니다.

파이썬에서는 re 모듈을 사용하여 정규표현식을 간단히 사용할 수 있습니다.

13.2. 실습예제

```
try:
    n=int(input('숫자를 입력하세요:')
    result=0
    for i in range(1,n+1):
        result += i
    print('합계:', result)
except:
    print('잘못된 값을 입력했습니다. 다시 입력하세요.')
```

위의 방법으로 사용자가 입력한 값이 숫자가 아닌 경우 처리할 수 있습니다. 좀더 자세한 방법으로 입력값을 확인하려면 어떻게 해야 할까요?

```
#re 모듈 - Regular Expression을 지원하는 모듈
import re
s = 'My id number is kim0902'
# findall('정규식패턴', 문자열)
a=re.findall('a',s)
print(a)
b=re.findall('kim',s)
print(b)
c=re.findall('m',s)
print(c)
```

```
  s = 'My id number is kim0502'
  # 소문자를 모두 찾아서 리스트로 반환
  a=re.findall('[a-z]', s)
  print(a)
  #  + 반복 옵션, 소문자를 하나씩 끊어서 찾지 않고 연속해서 찾음(단어 단위)
  b=re.findall('[a-z]+', s)
  print(b)
  # 대문자를 모두 찾아서 리스트로 반환
  c=re.findall('[A-Z]', s)
  print(c)
  # 숫자를 모두 찾아서 리스트로 반환
  d=re.findall('[0-9]', s)
  print(d)
  #  + 반복 옵션, 숫자를 하나씩 끊어서 찾지 않고 연속해서 찾음(단어 단위)
  e=re.findall('[0-9]+', s)
  print(e)
```

```
['y', 'i', 'd', 'n', 'u', 'm', 'b', 'e', 'r', 'i', 's', 'k', 'i', 'm']
['y', 'id', 'number', 'is', 'kim']
['M']
['0', '5', '0', '2']
['0502']
```

```
s = 'My id number is 김철수_kim_0502$%'
#소문자, 대문자, 숫자(문자 단위)
a=re.findall('[a-zA-Z0-9]', s)
print(a)
#소문자, 대문자, 숫자(단어 단위)
b=re.findall('[a-zA-Z0-9]+', s)
print(b)
#^ not , 소문자,대문자,숫자가 아닌 문자들(공백문자, 특수문자)
c=re.findall('[^a-zA-Z0-9]', s)
print(c)
#한글, 영문자, 숫자, _
d=re.findall('[₩w]', s)
print(d)
#한글, 영문자, 숫자, _ (단어단위)
e=re.findall('[₩w]+', s)
print(e)
# 한글, 영문자, 숫자,_ 가 아닌 경우
f=re.findall('[₩W]+', s)
print(f)
```

```
['M', 'y', 'i', 'd', 'n', 'u', 'm', 'b', 'e', 'r', 'i', 's', 'k', 'i', 'm', '0', '9', '0', '2']
['My', 'id', 'number', 'is', 'kim', '0902']
[' ', ' ', ' ', ' ', '김', '철', '수', '_', '_', '$', '%']
['M', 'y', 'i', 'd', 'n', 'u', 'm', 'b', 'e', 'r', 'i', 's', '김', '철', '수', '_', 'k', 'i', 'm', '_', '0', '9', '0', '2']
['My', 'id', 'number', 'is', '김철수_kim_0902']
[' ', ' ', ' ', ' ', '$%']
```

```
import re
# 비밀번호 정합성 체크를 위한 함수
def pwd_check(pwd):
    # 비밀번호 길이 확인 (6~12)
    if len(pwd) < 6 or len(pwd) > 12:
        print(pwd, ': 길이가 적당하지 않습니다.')
        return
    # 숫자 혹은 알파벳 유무 확인
    # 소문자, 대문자, 숫자만 선택, 특수문자는 걸러지게 됨
    # findall() 리스트로 리턴, [0] 첫번째 요소
    if re.findall('[a-zA-Z0-9]+', pwd)[0]  != pwd:
        print(pwd, ': 숫자와 영문자만 쓸 수 있습니다.')
        return
    # 알파벳 대소문자 확인
    # 소문자의 길이가 0이거나 대문자의 길이가 0이면
    if len(re.findall('[a-z]', pwd))==0 or len(re.findall('[A-Z]',
pwd))==0:
        print(pwd, ': 대문자와 소문자가 모두 필요합니다.')
        return
    print(pwd, ': 올바른 비밀 번호입니다.')

pwd_check('12abc')          # False, 길이 오류
pwd_check('123abc')     # False, 대문자 부재
pwd_check('123abc%')        # False, 특수문자 포함
pwd_check('123Abc')     # True
```

12abc : 길이가 적당하지 않습니다.
123abc : 대문자와 소문자가 모두 필요합니다.
123abc% : 숫자와 영문자만 쓸 수 있습니다.
123Abc : 올바른 비밀 번호입니다.

```
import re
# 이메일 주소 정합성 체크를 위한 함수
def email_check(email):
    exp=re.findall('^[a-z0-9]{2,}@[a-z0-9]{2,}₩.[a-z]{2,}$',ema
il)
    if len(exp) == 0:
        print(email, ': 이메일 규칙에 맞지 않습니다.')
        return False

    print(email, ': 올바른 이메일 주소입니다.')
    return True

email_check('kim@naver')
email_check('kim_naver.com')
email_check('kim')
email_check('kim@naver.com')

a=input('이메일 주소를 입력하세요.')
if email_check(a):
    print('올바른 이메일 주소입니다.')
else:
    print('이메일 규칙에 맞지 않습니다.')
```

kim@naver ==> 잘못된 이메일 형식입니다.
kim_naver.com ==> 잘못된 이메일 형식입니다.
kim ==> 잘못된 이메일 형식입니다.
kim@naver.com ==> 올바른 이메일 주소입니다!

14. 데이터베이스 프로그래밍

14.1. 데이터베이스(SQLite)

공식사이트 : http://sqlite.org
2000년 8월 발표, C언어로 개발

* 특징
파일 기반의 DBMS, 저메모리, 빠른 처리 속도
오픈소스
별도의 DB 서버가 없어도 쉽고 편리하게 사용할 수 있는 Embedded SQL DB 엔진
안드로이드, 아이폰 등의 스마트폰에 내장된 DB
표준 SQL 지원

* SQLite에서 지원하지 않는 기능(https://www.sqlite.org/omitted.html)
RIGHT and FULL OUTER JOIN은 지원되지 않음: left outer join은 가능함
ALTER TABLE 사용 가능하지만 DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT은 지원되지 않음
trigger 사용 가능
Writing to VIEWs : 읽기전용뷰만 가능
GRANT and REVOKE : 지원되지 않음

* Sqlite Database 실습(console)

```
cd C:₩anaconda3₩Library₩bin

sqlite3 c:/data/db/product.db

sqlite> create table if not exists product (id integer primary key autoincrement ,product_name varchar(50) not null,price int not null,amount int not null );
sqlite> .tables
sqlite> .schema product
sqlite> insert into product (product_name, price, amount) values ('냉장고', 500000, 5);
sqlite> select * from product;
sqlite> .quit
```

항공운항 csv 파일을 sqlite로 불러오는 코드

데이터 다운로드 주소
https://doi.org/10.7910/DVN/HG7NV7

```
cd C:₩anaconda3₩Library₩bin

sqlite3 c:/data/db/ontime.db

sqlite> create table ontime (
   Year int,
   Month int,
   DayofMonth int,
   DayOfWeek int,
   DepTime  int,
   CRSDepTime int,
   ArrTime int,
   CRSArrTime int,
   UniqueCarrier varchar(5),
   FlightNum int,
   TailNum varchar(8),
   ActualElapsedTime int,
   CRSElapsedTime int,
   AirTime int,
   ArrDelay int,
   DepDelay int,
   Origin varchar(3),
```

```
    Dest varchar(3),
    Distance int,
    TaxiIn int,
    TaxiOut int,
    Cancelled int,
    CancellationCode varchar(1),
    Diverted varchar(1),
    CarrierDelay int,
    WeatherDelay int,
    NASDelay int,
    SecurityDelay int,
    LateAircraftDelay int
  );

  sqlite> .schema ontime
  sqlite> .separator ,
  # 시간이 많이 걸리는 명령어
  sqlite> .import c:/data/ontime/2007.csv ontime
  # 첫 행 제거
  sqlite> delete from ontime where typeof(year) == "text";


  #인덱스 추가(시간이 오래 걸림)
  sqlite> create index year on ontime(year);
  sqlite> create index date on ontime(year, month, dayofmo
nth);
  sqlite> create index origin on ontime(origin);
  sqlite> create index dest on ontime(dest);
```

```
sqlite> select count(*) from ontime;



#항공사별 출발 지연 시간 평균
sqlite> select uniquecarrier, avg(depdelay)
from ontime
group by uniquecarrier;
#항공사별 평균 도착 지연 시간
sqlite> select uniquecarrier, avg(arrdelay)
from ontime
group by uniquecarrier;


#항공사별 출발 지연 횟수
sqlite> select uniquecarrier, count(*)
from ontime
where depdelay > 0
group by uniquecarrier;
#항공사별 도착 지연 횟수
sqlite> select uniquecarrier, count(*)
from ontime
where arrdelay > 0
group by uniquecarrier;
```

```
#항공사별 운항 횟수
sqlite> select uniquecarrier, count(*)
from ontime
group by uniquecarrier;


#항공사별 운항거리 합계
sqlite> select uniquecarrier, sum(distance)
from ontime
group by uniquecarrier;

#2007년 도착지연횟수 월별 집계
select month, count(*) from ontime
where year=2007 and depdelay > 0
group by month;

#요일별 평균 지연 시간
select dayofweek, avg(arrdelay)
from ontime
group by dayofweek;

#시간대별( 10월, 월요일에 출발할 때 가장 좋은 시간대는? )
select CRSDepTime, avg(arrdelay) arrdelay
from ontime
where month=10  and dayofweek=1
group by CRSDepTime
order by arrdelay
limit 1;
```

* GUI 툴 활용
http://www.sqliteexpert.com/
Personal 64bit 버전 다운로드 및 설치

* 실습예제

sqlite3 모듈은 파이썬 표준 라이브러리이므로 별도의 설치 작업 없이 곧바로 파이썬 코드에서 사용할 수 있음

```
import sqlite3  # SQLite3 라이브러리 로딩

#SQLite3 모듈의 버전
print(sqlite3.version)
#SQLite의 버전
print(sqlite3.sqlite_version)
```

```
import sqlite3  # SQLite3 라이브러리 로딩
# 테이블 생성
def create():
    conn = sqlite3.connect('books.db')  # 데이터베이스 커넥션 생성
    cursor = conn.cursor()  # 커서 생성
    # my_books 테이블 생성
    #제목, 출판일자, 출판사, 페이지수, 추천여부
    cursor.execute('''create table if not exists  books (
                        title text,
                        publish_date text,
                        publisher text,
                        pages integer,
                        recommend integer
                      )''')
    conn.close()          # 커넥션 닫기

create()
```

```
import sqlite3
# 데이터 입력 함수
def insert():
    conn = sqlite3.connect('books.db')# 데이터베이스 커넥션 생성
    cursor = conn.cursor()  # 커서 생성
    # 데이터 입력
    cursor.execute("insert into books values ('Java', '2022-02-28','길벗', 500, 10)")
    # 데이터 입력 SQL
    sql = 'insert into books values (?, ?, ?, ?, ?)'
    # 튜플을 이용한 데이터 입력
    cursor.execute(sql, ('Python', '2022-03-04','한빛', 584, 20))
    # 책의 정보를 담고 있는 튜플들의 리스트
    items = [
        ('빅데이터', '2021-07-02','삼성', 296, 11),
        ('안드로이드', '2021-02-10','영진', 526, 20),
        ('Spring', '2021-12-12','에이콘', 248, 15)
    ]
    # 여러 데이터 입력
    cursor.executemany(sql, items)
    conn.commit()
    conn.close()          # 커넥션 닫기

insert()
```

```
import sqlite3  # SQLite3

def list():
    conn = sqlite3.connect('books.db')
    cursor = conn.cursor()
    cursor.execute('select * from books')
    books = cursor.fetchall()
    for book in books:
        print(book)
    conn.close()
list()

def update():
    conn = sqlite3.connect('books.db')
    cursor = conn.cursor()
    # 데이터 수정 SQL
    sql = 'update books set recommend=? where title=?'
    # 수정 SQL 실행
    cursor.execute(sql, (100, 'Java'))
    conn.commit()
    conn.close()

update()
list()
```

```
def delete():
    conn = sqlite3.connect('books.db')
    cursor = conn.cursor()
    # 데이터 삭제 SQL
    sql = "delete from books where publisher='삼성' "
    # 수정 SQL 실행
    cursor.execute(sql)
    conn.commit()
    conn.close()

list()
delete()
print('='*50)
list()
```

```
import sqlite3
# 메모리에 데이터베이스 생성(in-memory database)
#conn = sqlite3.connect(':memory:')
# 파일에 생성
conn = sqlite3.connect('c:/data/db/sales.db')
sql = """
create table if not exists  sales(
    customer varchar(20),
    product varchar(50),
    price float,
    date date
)"""
conn.execute(sql)
data = [('김철수', '배', 10000, '2021-01-02'),
('김민수', '사과', 20000, '2021-01-15'),
('김현수', '포도', 15000, '2021-02-03'),
('한민호', '자두', 30000, '2021-02-20')]
sql = "insert into sales values(?, ?, ?, ?)"
conn.executemany(sql, data)
conn.commit()
# select query 실행
cursor = conn.execute("select * from sales")
rows = cursor.fetchall()
# 레코드 개수 계산
count = 0
for row in rows:
    print(row)
    count += 1
print( f'rows: {count}' )
```

```
# db 업데이트
import sqlite3
#import sys
conn = sqlite3.connect('c:/data/db/sales.db')
conn.execute("update sales set price=?, date=? where cust
omer=?", (80000,'2021-12-31','김철수'))
conn.commit()
cursor = conn.execute("select * from sales")
rows = cursor.fetchall()
for row in rows:
    output = []
    for idx in range(len(row)):
        output.append(str(row[idx]))
    print(output)
```

```
['김철수', '배', '80000.0', '2017-12-31']
['김민수', '사과', '20000.0', '2018-01-15']
['김현수', '포도', '15000.0', '2018-02-03']
['한민호', '자두', '30000.0', '2018-02-20']
```

```
#csv 파일로부터 읽은 데이터를 테이블에 insert
import csv
import sqlite3
input_file = 'c:/data/db/input.csv'
# sqlite database file 생성
conn = sqlite3.connect('c:/data/db/suppliers.db')
cursor = conn.cursor()
sql = """
create table if not exists suppliers (
    supplier_name varchar(20),
    invoice_number varchar(20),
    part_number varchar(20),
    cost float,
    purchase_date date
)"""
cursor.execute(sql)
sql="delete from suppliers"
cursor.execute(sql)
# csv 파일에서 데이터를 읽어서 테이블에 insert
file_reader = csv.reader(open(input_file, 'r'), delimiter=',')
#첫 라인을 읽음(제목행)
header = next(file_reader, None)
#print('header',header)
#header 이후의 2번째 행부터 끝까지 읽어들이며 insert
for row in file_reader:
    data = []
    # idx에는 0~4가 입력됨
    for idx in range(len(header)):
```

```
            data.append(row[idx])
        cursor.execute("insert into suppliers values (?, ?, ?, ?,
?)", data)
    conn.commit()
    rs = cursor.execute("select * from suppliers")
    rows = rs.fetchall()
    for row in rows:
        output = []
        for idx in range(len(row)):
            output.append(row[idx])
        print(output)

    conn.close()
```

```
['A', '001-1001', '2341', '500000.0', '2014-01-20']
['A', '001-1001', '2341', '500000.0', '2014-01-20']
['A', '001-1001', '5467', '750000.0', '2014-01-20']
['A', '001-1001', '5467', '750000.0', '2014-01-20']
['B', '50-9501', '7009', '250000.0', '2018-01-30']
['B', '50-9501', '7009', '250000.0', '2018-01-30']
['B', '50-9505', '6650', '125000.0', '2017-02-03']
['B', '50-9505', '6650', '125000.0', '2017-02-03']
['C', '920-4803', '3321', '615000.0', '2017-02-03']
['C', '920-4804', '3321', '615000.0', '2017-02-10']
['C', '920-4805', '3321', '615000.0', '2017-02-17']
['C', '920-4806', '3321', '615000.0', '2017-02-24']
```

항공운항 테이블 작업

```
import sqlite3

conn = sqlite3.connect('c:/data/db/ontime.db')
cursor = conn.cursor()
cursor.execute('select * from ontime limit 5')
rows = cursor.fetchall()
for row in rows:
    print(row)

conn.close()
```

```
#항공사별 출발 지연 시간 평균(실행시간이 오래 걸림)
conn = sqlite3.connect('c:/data/db/ontime.db')
cursor = conn.cursor()
cursor.execute('''select uniquecarrier, avg(depdelay)
from ontime
group by uniquecarrier''')
rows = cursor.fetchall()
for row in rows:
    print(row)

conn.close()
```

```
#항공사별 도착 지연 시간 평균(실행시간이 오래 걸림)
conn = sqlite3.connect('c:/data/db/ontime.db')
cursor = conn.cursor()
cursor.execute('''select uniquecarrier, avg(arrdelay)
from ontime
group by uniquecarrier''')
rows = cursor.fetchall()
for row in rows:
    print(row)

conn.close()
```

```
#2007년 출발지연횟수 월별 집계
conn = sqlite3.connect('c:/data/db/ontime.db')
cursor = conn.cursor()
cursor.execute('''select month, count(*) from ontime
where year=2007 and depdelay > 0
group by month''')
rows = cursor.fetchall()
for row in rows:
    print(row)

conn.close()
```

```
x=list(range(1,13))
y=[]
for row in rows:
    y.append(row[1])
print(x)
print(y)
```

```
import matplotlib.pyplot as plt

plt.plot(x,y)
plt.bar(x,y)
plt.show()
```

```
#요일별 평균 지연 시간
conn = sqlite3.connect('c:/data/db/ontime.db')
cursor = conn.cursor()
cursor.execute('''select dayofweek, avg(arrdelay)
from ontime
group by dayofweek''')
rows = cursor.fetchall()
for row in rows:
    print(row)

conn.close()
```

```
x=['월','화','수','목','금','토','일']
y=[]
for row in rows:
    y.append(row[1])
print(x)
print(y)
```

```
  import matplotlib.pyplot as plt

  from matplotlib import rc,font_manager
  font_name=font_manager.FontProperties(fname='c:/windows
/fonts/gulim.ttc').get_name()
  rc('font',family=font_name)

  plt.plot(x,y)
  plt.bar(x,y)
  plt.show()
```

```
  #시간대별( 10월, 월요일에 출발할 때 가장 지연이 안되는 시간
대는? )
  conn = sqlite3.connect('c:/data/db/ontime.db')
  cursor = conn.cursor()
  cursor.execute('''select CRSDepTime, avg(arrdelay) arrdelay
  from ontime
  where month=10  and dayofweek=1
  group by CRSDepTime
  order by avg(arrdelay)
  limit 1''')
  row = cursor.fetchone()
  print(row)
  conn.close()
```

14.2. 데이터베이스(MySQL)

http://mysql.com
MySQL Community Edition 다운로드
https://dev.mysql.com/downloads/windows/installer/8.0.html
https://dev.mysql.com/downloads/file/?id=514518

설치 완료 후 mysql 서비스 확인

HeidiSQL 설치
http://heidisql.com

Heidi SQL에서 사용자 계정 생성
도구 - 사용자관리자 메뉴에서 사용자 추가
아이디 : web
비번 : 1234
호스트 : localhost
전체권한에 체크

mysql 테이블 생성

```
create database my_suppliers;

use my_suppliers;

create table if not exists suppliers (
supplier_name varchar(20),
invoice_number varchar(20),
part_number varchar(20),
cost float,
purchase_date date
);

describe suppliers;
```

```
  # csv 파일을 읽어서  mysql 테이블에 insert
  # mysqlclient 패키지를 미리 설치해야 함
  # pip install mysqlclient
  import csv
  import MySQLdb
  # csv 파일의 경로
  input_file ='c:/data/db/input.csv'
  # mysql server에 접속
  conn = MySQLdb.connect(host='localhost', port=3306, ₩
  db='my_suppliers', user='web', passwd='1234', charset='utf
8' )
  cursor = conn.cursor()
  # 레코드 삭제
  sql = "delete from suppliers"
  cursor.execute(sql)
  conn.commit()
  #csv 파일을 로딩(필드 구분자는 쉼표)
  file_reader = csv.reader(open(input_file, 'r'), delimiter=',')
  # 제목행을 읽음
  header = next(file_reader)
  for row in file_reader:
       data = [] #빈 리스트
       if len(row) > 0:
            for idx in range(len(header)):
                data.append(row[idx])
            print(data)
            cursor.execute('INSERT INTO Suppliers VALUES (%s,
%s, %s, %s, %s)', data)
```

```
conn.commit()

cursor.execute("SELECT * FROM Suppliers")
rows = cursor.fetchall()
for row in rows:
    output = []
    for idx in range(len(row)):
        output.append(str(row[idx]))
    print(output)
```

```
  # 테이블 검색, 결과물을 csv 파일로 출력
  import csv
  import MySQLdb
  # mysql의 실행결과를 저장할 csv 파일의 경로
  output_file = 'c:/data/db/mysql_output.csv'
  conn = MySQLdb.connect(host='localhost', port=3306, db='
my_suppliers', user='web', passwd='1234', charset='utf8' )
  cursor = conn.cursor()
  # file writer 생성 및 제목행 출력
  filewriter = csv.writer(open(output_file, 'w',  newline=''),
delimiter=',')
  header = ['Supplier Name','Invoice Number','Part Number','
Cost','Purchase Date']
  filewriter.writerow(header)
  # sql 실행, 각각의 행을 파일에 출력
  cursor.execute('SELECT * FROM Suppliers WHERE Cost > 5
00000')
  rows = cursor.fetchall()
  for row in rows:
       print(row)
       filewriter.writerow(row)
```

```
  # 레코드 갱신
  import MySQLdb
  conn = MySQLdb.connect(host='localhost', port=3306, db='
my_suppliers', ₩
  user='web', passwd='1234')
  cursor = conn.cursor()
  # 제조사이름이 A인 레코드의 정보 수정
  sql = 'UPDATE Suppliers SET Cost=%s, Purchase_Date=%s
WHERE Supplier_Name=%s'
  cursor.execute(sql, (777,'2014-01-20','A'))
  conn.commit()
  cursor.execute("SELECT * FROM Suppliers")
  rows = cursor.fetchall()
  for row in rows:
      #레코드(튜플 자료형)를 리스트에 추가
      output = []
      for idx in range(len(row)):
          output.append(str(row[idx]))
      print(output)
```

항공운항테이블 실습

```
create database ontime;
use ontime;
create table ontime (
   Year int,
   Month int,
   DayofMonth int,
   DayOfWeek int,
   DepTime  int,
   CRSDepTime int,
   ArrTime int,
   CRSArrTime int,
   UniqueCarrier varchar(5),
   FlightNum int,
   TailNum varchar(8),
   ActualElapsedTime int,
   CRSElapsedTime int,
   AirTime int,
   ArrDelay int,
   DepDelay int,
   Origin varchar(3),
   Dest varchar(3),
   Distance int,
   TaxiIn int,
   TaxiOut int,
   Cancelled int,
   CancellationCode varchar(1),
```

```
    Diverted varchar(1),
    CarrierDelay int,
    WeatherDelay int,
    NASDelay int,
    SecurityDelay int,
    LateAircraftDelay int
);
```

cmd에서 mysql에 로그인

```
mysql -u root -p --local-infile=1
```

mysql 프롬프트에서 아래의 명령을 실행

```
set global local_infile=1;

use ontime;

#시간이 오래 걸림
LOAD DATA LOCAL INFILE 'c:/data/ontime/2007.csv' INTO TABLE ontime FIELDS TERMINATED BY ',' LINES TERMINATED BY '₩n';

delete from ontime limit 1;
```

```
#인덱스 만들기(시간이 오래 걸림)
create index year on ontime(year);
create index month on ontime(year,month);
create index date on ontime(year, month, dayofmonth);
create index dayofweek on ontime(dayofweek);
create index distance on ontime(distance);
create index uniquecarrier on ontime(uniquecarrier);
create index origin on ontime(origin);
create index dest on ontime(dest);
```

항공사 테이블 만들기

```
create table carrier( code varchar(100), description varchar (500) );
```

테이블로 import

csv 파일에 필드별로 큰따옴표로 묶여져 있는 것을 제거하기 위해 enclosed by '"' 를 추가해야 함

```
LOAD DATA LOCAL INFILE 'c:/data/ontime/carriers.csv' INTO TABLE carrier FIELDS TERMINATED BY ',' enclosed by '"' LINES TERMINATED BY '₩r₩n'  ignore 1 lines;
```

인덱스 추가

```
create index carrier_code on carrier(code);
```

평균출발지연시간

```
select uniquecarrier, avg(depdelay)
from ontime
group by uniquecarrier;
```

항공사 이름이 나오도록 join

```
select a.year,a.uniquecarrier,c.description, count(*)
from ontime a, carrier c
where a.uniquecarrier=c.code
and a.arrdelay > 0
group by a.year, a.uniquecarrier, c.description
order by count(*);
```

항공운항테이블(python에서 실습)

```
  import MySQLdb
  conn = MySQLdb.connect(host='localhost', port=3306, db='
ontime',
  user='web', passwd='1234')

  cursor = conn.cursor()
  cursor.execute('select * from ontime limit 5')
  rows = cursor.fetchall()
  for row in rows:
       print(row)

  cursor.close()
  conn.close()
```

```
  #항공사별 출발 지연 시간 평균(실행시간이 오래 걸림)
  import MySQLdb
  conn = MySQLdb.connect(host='localhost', port=3306, db='
ontime',
  user='web', passwd='1234')
  cursor = conn.cursor()
  cursor.execute('''select uniquecarrier, avg(depdelay)
  from ontime
  group by uniquecarrier''')
  rows = cursor.fetchall()
  for row in rows:
       print('{}₩t{}'.format(row[0],row[1]))

  cursor.close()
  conn.close()
```

```
  #항공사별 도착 지연 시간 평균(실행시간이 오래 걸림)
  import MySQLdb
  conn = MySQLdb.connect(host='localhost', port=3306, db='
ontime',
  user='web', passwd='1234')
  cursor = conn.cursor()
  cursor.execute('''select uniquecarrier, avg(arrdelay)
  from ontime
  group by uniquecarrier''')
  rows = cursor.fetchall()
  for row in rows:
       print(row)

  conn.close()
```

```
#2008년 도착지연횟수 월별 집계
import MySQLdb
conn = MySQLdb.connect(host='localhost', port=3306, db='
ontime',
user='web', passwd='1234')
cursor = conn.cursor()
cursor.execute('''select month, count(*) from ontime
where year=2008 and arrdelay > 0
group by month''')
rows = cursor.fetchall()
for row in rows:
    print(row)

cursor.close()
conn.close()
```

```
x=list(range(1,13))
y=[]
for row in rows:
    y.append(row[1])
print(x)
print(y)
```

```
import matplotlib.pyplot as plt


plt.plot(x,y)
plt.bar(x,y)
plt.show()
```

```
#요일별 평균 지연 시간
import MySQLdb
conn = MySQLdb.connect(host='localhost', port=3306, db='
ontime',
user='web', passwd='1234')
cursor = conn.cursor()
cursor.execute('''select dayofweek, avg(arrdelay)
from ontime
group by dayofweek''')
rows = cursor.fetchall()
for row in rows:
    print(row)

cursor.close()
conn.close()
```

```
x=['월','화','수','목','금','토','일']
y=[]
for row in rows:
    y.append(row[1])
print(x)
print(y)
```

```
  import matplotlib.pyplot as plt

  from matplotlib import rc,font_manager
  font_name=font_manager.FontProperties(fname='c:/windows
/fonts/gulim.ttc').get_name()
  rc('font',family=font_name)

  plt.plot(x,y)
  plt.bar(x,y)
  plt.show()
```

```
  #시간대별( 10월, 월요일에 출발할 때 가장 좋은 시간대는? )
  import MySQLdb
  conn = MySQLdb.connect(host='localhost', port=3306, db='
ontime',
  user='web', passwd='1234')
  cursor = conn.cursor()
  cursor.execute('''select CRSDepTime, avg(arrdelay) arrdelay
  from ontime
  where month=10  and dayofweek=1
  group by CRSDepTime
  order by avg(arrdelay)
  limit 1''')
  rows = cursor.fetchall()
  for row in rows:
       print(row)

  cursor.close()
  conn.close()
```

14.3. 데이터베이스(oracle)

cmd

```
sqlplus system/1234
```

sqlplus

```
create tablespace myts
datafile 'pydb.dbf' size 10m
autoextend on
next 10m
maxsize unlimited;

--oracle 12c 이상인 경우
alter session set "_ORACLE_SCRIPT"=true;

create user python identified by 1234
default tablespace myts;

grant connect, resource, dba to python;
```

실습용 데이터 설치

테이블 생성 및 데이터 입력

```
drop table product;
drop table product cascade constraints;
create table product (
product_id number,
product_name varchar2(50),
price number default 0,
description clob,
picture_url varchar2(500),
primary key(product_id)
);
insert into product values (1,'레몬',1500,
'레몬에 포함된 구연산은 피로회복에 좋습니다. 비타민 C도 풍부합니다.','lemon.jpg');
insert into product values (2,'오렌지',2000,
'비타민 C가 풍부합니다. 생과일 주스로 마시면 좋습니다.','orange.jpg');
insert into product values (3,'키위',3000,
'비타민 C가 매우 풍부합니다. 다이어트나 미용에 좋습니다.','kiwi.jpg');
insert into product values (4,'포도',5000,
'폴리페놀을 다량 함유하고 있어 항산화 작용을 합니다.','grape.jpg');
insert into product values (5,'딸기',8000,
'비타민 C나 플라보노이드를 다량 함유하고 있습니다.','strawberry.jpg');
```

```
  insert into product values (6,'귤',7000,
  '시네피린을 함유하고 있어 감기 예방에 좋다고 합니다.','tangerine.jpg');
  commit;
```

파이썬 코드

```
  # pip install cx_Oracle
  import cx_Oracle
  conn = cx_Oracle.connect("python/1234@localhost:1521/xe")
  cursor = conn.cursor()

  sql="delete from product"
  cursor.execute(sql)

  items = [
  (1,'레몬',1500,₩
  '레몬에 포함된 구연산은 피로회복에 좋습니다. 비타민 C도 풍부합니다.','lemon.jpg'),
  (2,'오렌지',2000,₩
  '비타민 C가 풍부합니다. 생과일 주스로 마시면 좋습니다.','orange.jpg'),
  (3,'키위',3000,₩
  '비타민 C가 매우 풍부합니다. 다이어트나 미용에 좋습니다.','kiwi.jpg'),
  (4,'포도',5000,'폴리페놀을 다량 함유하고 있어 항산화 작용을 합니다.','grape.jpg'),
  (5,'딸기',8000,₩
  '비타민 C나 플라보노이드를 다량 함유하고 있습니다.','strawberry.jpg'),
  (6,'귤',7000,₩
  '시네피린을 함유하고 있어 감기 예방에 좋다고 합니다.','tangerine.jpg')
  ]
```

```
for row in items:
    sql = "insert into product values (:1,:2,:3,:4,:5)"
    cursor.execute(sql, row)
# 레코드 개수
sql = "select count(*) from product"
cursor.execute(sql)
count=cursor.fetchone()
print("상품개수:",count[0])




# 전체 레코드 조회
sql = "select * from product"
cursor.execute(sql)
for row in cursor:
    # CLOB 필드를 읽는 방법
    description=row[3].read()
    print(row)
    print(description)

#일부 레코드 삭제
sql = "delete from product where product_id=6"
cursor.execute(sql)
#모든 레코드 삭제
sql = "delete from product"
cursor.execute(sql)
```

```
#모든 레코드를 한꺼번에 insert
sql="insert into product values (:1,:2,:3,:4,:5)"
cursor.executemany(sql, items)

#모든 레코드를 한꺼번에 조회하여 리스트로 저장
sql = "select * from product"
cursor.execute(sql)
rs=cursor.fetchall()
print("₩n레코드셋:",rs)
for row in rs:
    print(row)

conn.commit()
cursor.close()
conn.close()
```

```
import cx_Oracle
conn=cx_Oracle.connect('python/1234@localhost:1521/xe')
cursor=conn.cursor()
sql='''
select s.majorno,mname,count(*)
from stud s, major m
where s.majorno=m.majorno
group by s.majorno,mname
order by s.majorno
'''
cursor.execute(sql)
rows=cursor.fetchall()
for row in rows:
    print(row)

cursor.close()
conn.close()
```

```
x=list(range(1,7)) #1~6
names=[]
y=[]
for row in rows:
    names.append(row[1])
    y.append(row[2])

print(x)
print(names) #학과명
print(y) #학생수
```

```
import matplotlib.pyplot as plt
#그래프 출력 옵션
from matplotlib import rc,font_manager
font_name=font_manager.FontProperties(
        fname='c:/windows/fonts/gulim.ttc').get_name()
rc('font',family=font_name)
plt.rcParams['figure.figsize']=(16,9)
plt.title('학과별 학생수',fontsize=20) #제목
plt.pie(y,labels=names,autopct='%.1f%%') #파이 차트
#plt.legend(names,loc='upper right') #범례
#bbox_to_anchor: figure의 width,height를 1.0을 기준으로 설정
#legend의 좌측상단좌표를 figure의 1.0, 1.2에 위치시킴
#figure의 바깥쪽에 legend가 표시됨
plt.legend(names,bbox_to_anchor=[1, 1.2]) #범례
plt.show()
```

```
import pandas as pd
df=pd.read_csv('c:/data/iris/iris.csv')
df.head()
==============================
from sqlalchemy import create_engine
import cx_Oracle
# oracle server에 접속하기 위한 엔진 생성
# oracle_cx_oracle://아이디:비번@db
engine=create_engine('oracle+cx_oracle://python:1234@xe')
conn=engine.connect()
# 데이터프레임의 내용을 테이블로 export
df.to_sql(name='iris', con=engine, if_exists='replace',index=F
alse)
```

항공운항 테이블 실습

대용량 데이터의 경우 sqldeveloper보다는 sqlplus에서 실행하는 것이 좀더 빠른 속도로 처리될 수 있음.

테이블 만들기

null , NA , 빈값 등이 있으면 에러가 발생하므로 대부분의 자료형을 varchar로 설정하였음.

```
create table ontime (
    Year int,
    Month int,
    DayofMonth int,
    DayOfWeek int,
    DepTime  varchar(50),
    CRSDepTime int,
    ArrTime varchar(50),
    CRSArrTime int,
    UniqueCarrier varchar(5),
    FlightNum varchar(50),
    TailNum varchar(8),
    ActualElapsedTime varchar(50),
    CRSElapsedTime varchar(50),
    AirTime varchar(50),
    ArrDelay varchar(50),
    DepDelay varchar(50),
    Origin varchar(3),
    Dest varchar(3),
    Distance varchar(50),
```

```
    TaxiIn varchar(50),
    TaxiOut varchar(50),
    Cancelled varchar(50),
    CancellationCode varchar(1),
    Diverted varchar(1),
    CarrierDelay varchar(50),
    WeatherDelay varchar(50),
    NASDelay varchar(50),
    SecurityDelay varchar(50),
    LateAircraftDelay varchar(50)
);
```

SQL Loader 툴을 이용하여 csv 파일을 테이블로 import 하기 위하여 ctl 파일을 만들어야 함

trailing nullcols - null 값도 입력 가능하도록 처리하는 옵션

c:/data/ontime/ontime.ctl

```
options(skip =1)
load data
infile 'c:/data/ontime/2008.csv'
replace
into table python.ontime
fields terminated by ','
OPTIONALLY ENCLOSED BY '"'
trailing nullcols
(
  Year,
  Month,
  DayofMonth,
  DayOfWeek,
  DepTime,
  CRSDepTime,
  ArrTime,
  CRSArrTime,
  UniqueCarrier,
  FlightNum,
  TailNum,
  ActualElapsedTime,
  CRSElapsedTime,
  AirTime,
  ArrDelay,
```

```
    DepDelay,
    Origin,
    Dest,
    Distance,
    TaxiIn,
    TaxiOut,
    Cancelled,
    CancellationCode,
    Diverted,
    CarrierDelay,
    WeatherDelay,
    NASDelay,
    SecurityDelay,
    LateAircraftDelay
  )
```

cmd에서 실행(시간이 오래 걸리는 작업)

```
  sqlldr python/1234 control='c:/data/ontime/ontime.ctl' log='
c:/data/ontime/fail.log'
```

```
#인덱스 만들기(시간이 오래 걸림)
create index year on ontime(year);
create index month on ontime(year,month);
create index day on ontime(year, month, dayofmonth);
create index dayofweek on ontime(dayofweek);
create index distance on ontime(distance);
create index uniquecarrier on ontime(uniquecarrier);
create index origin on ontime(origin);
create index dest on ontime(dest);
```

항공사 테이블 만들기

```
create table carrier( code varchar2(100), description varchar2(500) );
```

테이블로 import

csv 파일에 필드별로 큰따옴표로 묶여져 있는 것을 제거하기 위해 enclosed by '"' 를 추가해야 함

c:/data/ontime/carrier.ctl

```
options(skip =1)
load data
infile 'c:/data/ontime/carriers.csv'
replace
into table python.carrier
fields terminated by ','
OPTIONALLY ENCLOSED BY '"'
trailing nullcols
(
  code,
  description
)
```

cmd에서 실행

```
sqlldr python/1234 control='c:/data/ontime/carrier.ctl' log='c:/data/ontime/carrier_fail.log'
```

인덱스 추가

```
create index carrier_code on carrier(code);
```

항공사별 출발지연시간 평균(시간이 오래 걸림)

```
update ontime set depdelay=0 where depdelay='NA';
commit;

select uniquecarrier, avg(depdelay)
from ontime
group by uniquecarrier;
```

항공사 이름이 나오도록 join

```
update ontime set arrdelay=0 where arrdelay='NA';
commit;

select a.year,a.uniquecarrier,c.description, count(*)
from ontime a, carrier c
where a.uniquecarrier=c.code
and a.arrdelay > 0
group by a.year, a.uniquecarrier, c.description
order by a.year, a.uniquecarrier,c.description;
```

항공운항테이블(python에서 실습)

```
import cx_Oracle
conn = cx_Oracle.connect("python/1234@localhost:1521/xe
")

cursor = conn.cursor()
cursor.execute('select * from ontime where rownum < 5')

rows = cursor.fetchall()
for row in rows:
    print(row)

cursor.close()
conn.close()
```

```
  #항공사별 출발 지연 시간 평균(실행시간이 오래 걸림)
  import cx_Oracle
  conn = cx_Oracle.connect("python/1234@localhost:1521/xe
")

  cursor = conn.cursor()
  cursor.execute('''select uniquecarrier, avg(depdelay)
  from ontime
  group by uniquecarrier''')
  rows = cursor.fetchall()
  for row in rows:
       print('{}₩t{}'.format(row[0],row[1]))

  cursor.close()
  conn.close()
```

```
#항공사별 도착 지연 시간 평균(실행시간이 오래 걸림)
import cx_Oracle
conn = cx_Oracle.connect("python/1234@localhost:1521/xe
")

cursor = conn.cursor()
cursor.execute('''select uniquecarrier, avg(arrdelay)
from ontime
group by uniquecarrier''')
rows = cursor.fetchall()
for row in rows:
    print(row)

conn.close()
```

```
#2008년 도착지연횟수 월별 집계
import cx_Oracle
conn = cx_Oracle.connect("python/1234@localhost:1521/xe
")

cursor = conn.cursor()
cursor.execute('''select month, count(*) from ontime
where year=2008 and arrdelay > 0
group by month''')
rows = cursor.fetchall()
for row in rows:
    print(row)

cursor.close()
conn.close()
```

```
x=list(range(1,13))
y=[]
for row in rows:
    y.append(row[1])
print(x)
print(y)
```

```
import matplotlib.pyplot as plt

plt.plot(x,y)
plt.bar(x,y)
plt.show()
```

```
#요일별 평균 지연 시간
import cx_Oracle
conn = cx_Oracle.connect("python/1234@localhost:1521/xe
")

cursor = conn.cursor()
cursor.execute('''select dayofweek, avg(arrdelay)
from ontime
group by dayofweek''')
rows = cursor.fetchall()
for row in rows:
    print(row)

cursor.close()
conn.close()
```

```
x=['월','화','수','목','금','토','일']
y=[]
for row in rows:
    y.append(row[1])
print(x)
print(y)
```

```
  import matplotlib.pyplot as plt
  from matplotlib import rc,font_manager
  font_name=font_manager.FontProperties(fname='c:/windows
/fonts/gulim.ttc').get_name()
  rc('font',family=font_name)

  plt.plot(x,y)
  plt.bar(x,y)
  plt.show()
```

```
  #시간대별( 10월, 월요일에 출발할 때 가장 좋은 시간대는? )
  import cx_Oracle
  conn = cx_Oracle.connect("python/1234@localhost:1521/xe
")

  cursor = conn.cursor()
  cursor.execute('''
  select *
  from (
       select rownum as rn, A.*
       from (
        select CRSDepTime, avg(arrdelay) arrdelay
        from ontime
        where month=10  and dayofweek=1
        group by CRSDepTime
        order by arrdelay
       ) A
  ) where rn between 1 and 5
  ''')
  rows = cursor.fetchall()
  for row in rows:
       print(row)

  cursor.close()
  conn.close()
```

14.4. 데이터베이스(SQL Server)

SQL Server 2019 express edition 다운로드 및 설치

https://www.microsoft.com/ko-kr/sql-server/sql-server-downloads

설치유형 - 미디어 다운로드 선택

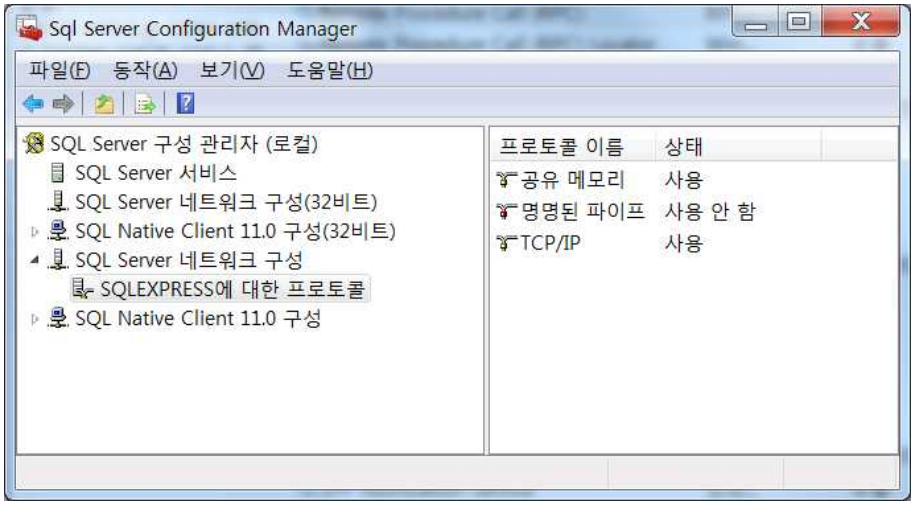
새 SQL Server 독립 실행형 설치 또는 기존 설치에 추가

기본 설치 옵션으로 진행

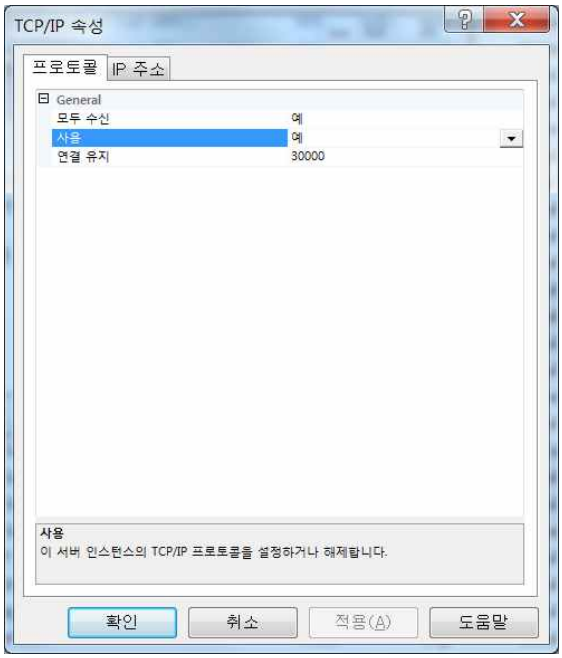
인증 모드는 혼합 모드로 설치(암호 입력 : 1234)

tcp/ip 원격 접속 설정

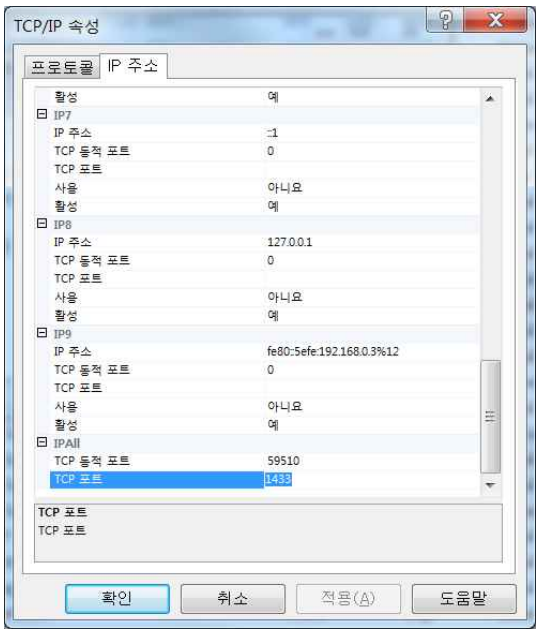
SQL Server 구성 관리자 > SQL Server 네트워크 구성에서 TCP/IP 클릭



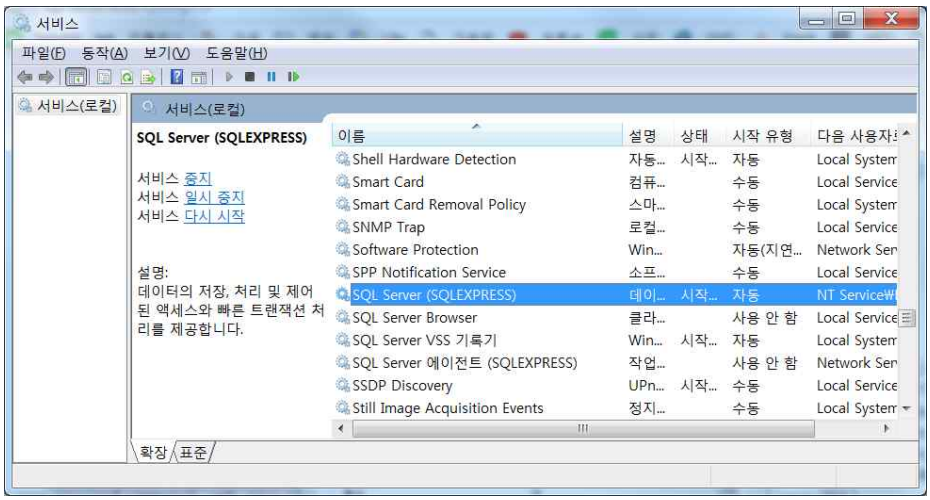
프로토콜 탭에서 사용 - 예로 변경



IP 주소 탭 - 맨 아래로 내려가서 TCP 포트에 1433 입력



서비스 재시작 -



SQL Server에 데이터베이스 및 테이블 생성

HeidiSQL로 접속

아이디 : sa
암호 : 1234

```
create database web;
```

상품테이블 실습

```
use web;
-- varchar 자료형은 한글이 깨질 수 있으므로 nvarchar 자료형(유니코드 가변문자열)을 사용함
create table product (
product_id int,
product_name nvarchar(50),
price int default 0,
description nvarchar(4000),
picture_url nvarchar(500),
primary key(product_id)
);

insert into product values (1,'레몬',1500,
'레몬에 포함된 구연산은 피로회복에 좋습니다. 비타민 C도 풍부합니다.','lemon.jpg');

insert into product values (2,'오렌지',2000,
'비타민 C가 풍부합니다. 생과일 주스로 마시면 좋습니다.','orange.jpg');

insert into product values (3,'키위',3000,
'비타민 C가 매우 풍부합니다. 다이어트나 미용에 좋습니다.','kiwi.jpg');

insert into product values (4,'포도',5000,
'폴리페놀을 다량 함유하고 있어 항산화 작용을 합니다.','grape.jpg');
```

```
  insert into product values (5,'딸기',8000,
  '비타민 C나 플라보노이드를 다량 함유하고 있습니다.','strawberry.jpg');


  insert into product values (6,'귤',7000,
  '시네피린을 함유하고 있어 감기 예방에 좋다고 합니다.','tangerine.jpg');

  select * from product;
```

```
  # pymssql 패키지 import
  # pip install pymssql 에러가 날 경우 conda install pymssql
  import pymssql

  # MSSQL 접속
  conn = pymssql.connect(host=r'(local)', database='web', us
er='sa', password='1234', charset='utf8')

  # Connection 으로부터 Cursor 생성
  cursor = conn.cursor()

  # SQL문 실행
  cursor.execute('SELECT * FROM product')

  rows = cursor.fetchall()
  for row in rows:
      print(row)

  # 연결 끊기
  conn.close()
```

```
  import pymssql

  conn = pymssql.connect(host=r'(local)', database='web', ch
arset='utf8')
  cursor = conn.cursor()

  sql="delete from product"
  cursor.execute(sql)

  items = [
  (1,'레몬',1500,₩
  '레몬에 포함된 구연산은 피로회복에 좋습니다. 비타민 C도
풍부합니다.','lemon.jpg'),
  (2,'오렌지',2000,₩
  '비타민 C가 풍부합니다. 생과일 주스로 마시면 좋습니다.','orang
e.jpg'),
  (3,'키위',3000,₩
  '비타민 C가 매우 풍부합니다. 다이어트나 미용에 좋습니다.','kiw
i.jpg'),
  (4,'포도',5000,'폴리페놀을 다량 함유하고 있어 항산화 작용을
합니다.','grape.jpg'),
  (5,'딸기',8000,₩
  '비타민 C나 플라보노이드를 다량 함유하고 있습니다.','strawberr
y.jpg'),
  (6,'귤',7000,₩
  '시네피린을 함유하고 있어 감기 예방에 좋다고 합니다.','tangeri
ne.jpg')
  ]
```

```
  for row in items:
        print(row)
        sql="INSERT INTO product VALUES (%s, %s, %s, %s,
%s)"
        cursor.execute(sql, row)

  # 레코드 개수
  sql = "select count(*) from product"
  cursor.execute(sql)
  cnt=cursor.fetchone()
  print("상품개수:",cnt[0])

  # 전체 레코드 조회
  sql = "select * from product"
  cursor.execute(sql)

  for row in cursor:
        print(row)

  #일부 레코드 삭제
  sql = "delete from product where product_id=6"
  cursor.execute(sql)

  #모든 레코드 삭제
  sql = "delete from product"
  cursor.execute(sql)
```

```
#모든 레코드를 한꺼번에 insert
sql="insert into product values (%s, %s, %s, %s, %s)"
cursor.executemany(sql, items)


#모든 레코드를 한꺼번에 조회하여 리스트로 저장
sql = "select * from product"
cursor.execute(sql)
rs=cursor.fetchall()
print("₩n레코드셋:",rs)
for row in rs:
    print(row)

cursor.close()
conn.close()
```

* 항공운항데이터 실습

csv 파일을 테이블로 가져오는 bulk insert 명령어도 있으나 속도가 매우 느리므로 SQL Server Management Studio에서 데이터 가져오기/내보내기 마법사를 사용하는 방법을 추천함.

데이터베이스 이름 선택 > 우클릭 > 태스크 > 데이터 가져오기

데이터 원본 - 플랫파일원본 > 열, 고급, 미리 보기 확인 후 진행

데이터 import 작업이 성공하면 테이블 이름을 ontime으로 변경함

```
--테이블 목록보기
select * from information_schema.tables;

--테이블의 필드 상세내용 보기 : sp_columns 테이블명
select * from information_schema.columns
where table_name = 'ontime';

select column_name, data_type, character_maximum_lengt
h, column_default
from information_schema.columns
where table_name = 'ontime';
```

NA 값을 0으로 수정

```
update ontime set depdelay=0 where depdelay='NA';
update ontime set arrdelay=0 where arrdelay='NA';
```

```
#인덱스 만들기
create index year on ontime(year);
create index month on ontime(year,month);
create index day on ontime(year, month, dayofmonth);
create index dayofweek on ontime(dayofweek);
create index distance on ontime(distance);
create index uniquecarrier on ontime(uniquecarrier);
create index origin on ontime(origin);
create index dest on ontime(dest);
```

항공사 테이블 import

c:/data/ontime/carriers.csv
SQL Server의 마법사를 이용하여 처리
기본 필드 사이즈가 50으로 되어 있으므로 에러가 발생함
사이즈를 500으로 늘린 후 다시 실행

인덱스 추가

```
create index carrier_code on carriers(code);
```

항공사별 출발지연시간 평균

```
select uniquecarrier, avg(depdelay)
from ontime
group by uniquecarrier;
```

자료형이 varchar이므로 에러가 발생함.

자료형을 변경하고 하는 것이 원칙이지만 실행 시간이 매우 오래 걸리므로 임시 형변환을 하는 방법을 추천함.

```
#이 방법은 추천하지 않음
alter table ontime alter column depdelay int;
alter table ontime alter column arrdelay int;
```

```
#이 방법을 추천함
select uniquecarrier, avg(cast(depdelay as int))
from ontime
group by uniquecarrier;
```

항공사 이름이 나오도록 join
replace - 큰따옴표를 제거하여 조인이 되도록 함

```
select year,uniquecarrier,description, count(*)
from ontime a, carriers c
where a.uniquecarrier=replace(c.code, '"', '')
and arrdelay > 0
group by year, uniquecarrier, description
order by year, uniquecarrier,description;
```

항공운항테이블(python에서 실습)

```
  # pymssql 패키지 import
  import pymssql

  # MSSQL 접속
  conn = pymssql.connect(host=r'(local)', database='web', us
er='sa', password='1234', charset='utf8')

  cursor = conn.cursor()
  cursor.execute('select top 5 * from ontime')
  rows = cursor.fetchall()
  for row in rows:
      print(row)

  cursor.close()
  conn.close()
```

```
  #항공사별 출발 지연 시간 평균(실행시간이 오래 걸림)
  # pymssql 패키지 import
  import pymssql

  # MSSQL 접속
  conn = pymssql.connect(host=r'(local)', database='web', us
er='sa', password='1234', charset='utf8')

  cursor = conn.cursor()
  cursor.execute('''select uniquecarrier, avg(cast(depdelay as i
nt)) depdelay
  from ontime
  group by uniquecarrier
  order by depdelay
  ''')
  rows = cursor.fetchall()
  for row in rows:
       print(f'{row[0]}₩t{row[1]}')

  cursor.close()
  conn.close()
```

```
  #항공사별 도착 지연 시간 평균(실행시간이 오래 걸림)
  # pymssql 패키지 import
  import pymssql

  # MSSQL 접속
  conn = pymssql.connect(host=r'(local)', database='web', us
er='sa', password='1234', charset='utf8')

  cursor = conn.cursor()
  cursor.execute('''select uniquecarrier, avg(cast(arrdelay as in
t))
  from ontime
  group by uniquecarrier''')
  rows = cursor.fetchall()
  for row in rows:
      print(row)

  conn.close()
```

```
#2008년 도착지연횟수 월별 집계
# pymssql 패키지 import
import pymssql

# MSSQL 접속
conn = pymssql.connect(host=r'(local)', database='web', us
er='sa', password='1234', charset='utf8')

cursor = conn.cursor()
cursor.execute('''select month, count(*) from ontime
where year=2008 and arrdelay > 0
group by month
order by cast(month as int)
''')
rows = cursor.fetchall()
for row in rows:
    print(row)

cursor.close()
conn.close()
```

```
x=list(range(1,13))
y=[]
for row in rows:
    y.append(row[1])
print(x)
print(y)
```

```
import matplotlib.pyplot as plt

plt.plot(x,y)
plt.bar(x,y)
plt.show()
```

```
#요일별 지연시간합계
# pymssql 패키지 import
import pymssql

# MSSQL 접속
conn = pymssql.connect(host=r'(local)', database='web', user='sa', password='1234', charset='utf8')

cursor = conn.cursor()
cursor.execute('''select dayofweek, sum(cast(arrdelay as int))
from ontime
group by dayofweek
order by dayofweek
''')
rows = cursor.fetchall()
for row in rows:
    print(row)

cursor.close()
conn.close()
```

```
x=['월','화','수','목','금','토','일']
y=[]
for row in rows:
    y.append(row[1])
print(x)
print(y)
```

```
  import matplotlib.pyplot as plt
  from matplotlib import rc,font_manager
  font_name=font_manager.FontProperties(fname='c:/windows
/fonts/gulim.ttc').get_name()
  rc('font',family=font_name)

  plt.plot(x,y)
  plt.bar(x,y)
  plt.show()
```

```
  #시간대별( 10월, 월요일에 출발할 때 가장 좋은 시간대는? )
  # pymssql 패키지 import
  import pymssql

  # MSSQL 접속
  conn = pymssql.connect(host=r'(local)', database='web', us
er='sa', password='1234', charset='utf8')

  cursor = conn.cursor()
  cursor.execute('''
        select top 5 CRSDepTime, avg(cast(arrdelay as int)) ar
rdelay
        from ontime
        where month=10  and dayofweek=1
        group by CRSDepTime
        order by arrdelay
  ''')
  rows = cursor.fetchall()
  for row in rows:
      print(row)

  cursor.close()
  conn.close()
```

14.5. 데이터베이스(MongoDB)

* NoSQL(Not Only SQL) : SQL만을 사용하지 않는 데이터베이스 관리 시스템

기존의 RDBMS의 한계를 극복하기 위해 만들어진 새로운 형태의 데이터저장소
RDBMS처럼 고정된 스키마가 존재하지 않으며 join을 사용할 수 없음
비정형 데이터를 저장하기 위해 최적화된 저장 방법을 제공함

현실 세계의 모든 데이터가 일정한 틀과 형태를 가지고 있지는 않습니다. ex) 대화, 채팅, 음악 등
테이블을 만드는 것은 공통적인 속성들을 선택하는 작업

종류 - MongoDB, HBase 등

MongoDB

NoSQL로 분류되는 Cross Flatform Document 지향 데이터베이스 시스템

전통적인 테이블-관계 기반의 RDBMS처럼 스키마가 고정된 구조가 아닌 JSON 형태의 동적 스키마형 문서를 사용함

몽고DB 사용 예 - 이베이, 뉴욕 타임즈 등

Collection 안에 Document 데이터 저장
Document는 일정한 틀을 가지지 않음
Document 내부의 Field 자료형 형식이 달라도 입력 가능하며 각 Document들은 모두 일관된 Field를 가지지 않아도 됨

Table => Collection
Record => Document

http://mongodb.com

C++로 작성된 오픈소스 문서지향(Document Oriented) Cross-platform 데이터베이스

1) MongoDB Community Server 다운로드

Software > Community Server > MongoDB Community Server 다운로드

https://www.mongodb.com/try/download/community

6.0.3 Windows zip 버전 다운로드

압축을 해제한 후 c root 디렉토리로 이동시키고 디렉토리 이름을 c:/mongodb 로 변경

2) MongoDB Shell 다운로드
https://www.mongodb.com/try/download/shell

1.6.0 Windows zip 버전 다운로드
압축을 해제한 후 c root 디렉토리로 이동시키고 디렉토리 이름을 c:/mongosh 로 변경

실행

mongod.exe 서버 실행 파일
mongosh.exe 쉘 프로그램

먼저 디렉토리를 생성해야 함
c:₩data₩db 디렉토리 생성 (몽고db에서 기본적으로 참조하는 디렉토리)
(변경하려면 mongod --dbpath "c:₩mongodb₩data₩db")
cd C:₩mongodb₩bin

서버 실행
cd c:/mongodb/bin
mongod.exe

새로운 cmd 창을 열어서 몽고디비 쉘 실행
cd c:/mongosh/bin
mongosh.exe

* mongo db 인증 설정

관리자 계정 만들기

```
use admin
db.createUser(
 {
 user: "admin",
 pwd: "1234",
 roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
 })
```

admin 데이터베이스가 생성되고 admin 관리자 계정이 생성된다.

사용권한 참조

https://docs.mongodb.com/manual/core/authorization/

Ctrl+C을 눌러서 mongod.exe 와 mongosh.exe 프로그램을 종료시킴

mongoDB 서버 인증모드로 재시작

mongod --auth

관리자 계정으로 접속(큰따옴표 사용)

mongosh --port 27017 -u "admin" -p "1234" --authentication Database "admin"

또는
mongosh 로 접속한 후
use admin
db.auth("admin", "1234")
=> 1이 출력되면 인증 성공

* 일반사용자 계정 만들기

```
use web
db.createUser(
{
 user: "web",
 pwd: "1234",
 roles: [ { role: "readWrite", db: "web" },
 { role: "read", db: "reporting" } ]
})
```

데이터베이스마다 다른 권한을 부여할 수 있음
web 데이터베이스는 읽기+쓰기 가능
reporting 데이터베이스는 읽기만 가능

mongoDB 서버 인증모드로 재시작

mongod --auth

일반사용자 계정으로 접속

mongosh --port 27017 -u "web" -p "1234" --authenticationDatabase "web"

또는

mongosh 로 접속한 후
use web
db.auth("web", "1234")
=> 1이 출력되면 인증 성공

실습예제

```
#pip install pymongo
import pymongo
from pymongo import MongoClient
```

```
#인증모드로 접속
conn=MongoClient('mongodb://web:1234@localhost/web')
conn
```

```
db=conn.web
db
```

```
#컬렉션 생성
students=db.students
students
```

```
#컬렉션에 저장된 모든 문서들을 제거
try:
    students.drop()
except:
    print('자료가 없습니다.')
```

```
#도큐먼트 생성
row1={'no':1, 'name':'김철수','kor':90,'eng':80,'mat':77}
row2={'no':2, 'name':'박민성','kor':95,'eng':83,'mat':76}
row3={'no':3, 'name':'송영희','kor':92,'eng':87,'mat':75}

#레코드 1개 추가
students.insert_one(row1)
students.insert_one(row2)
students.insert_one(row3)

#전체 문서 조회
rows=students.find()
for row in rows:
    print(row)
```

```
#문서 개별 삭제
students.delete_one({'no':2})
rows=students.find()
for row in rows:
    print(row)
```

```
#문서 수정
students.update_one({'no':1}, {'$set':{'kor':100,'eng':100}})
```

```
rows=students.find()
for row in rows:
    print(row)
```

```
  #일괄 수정
  rows=students.find()
  for row in rows:
        tot=row['kor']+row['eng']+row['mat']
        avg=tot/3
        if avg>=90:
              grade='A'
        elif 80<=avg<90:
              grade='B'
        elif 70<=avg<80:
              grade='C'
        elif 60<=avg<70:
              grade='D'
        else:
              grade='F'
        students.update_one({'no':row['no']}, {'$set':{'tot':tot,'av
g':avg,'grade':grade}})

  rows=students.find()
  for row in rows:
        print(row)
```

```
  rows=students.find()
  print('학번₩t이름₩t국어₩t영어₩t수학₩t총점₩t평균₩t등급')
  print('='*60)
  for row in rows:
      print(f"{row['no']}₩t{row['name']}₩t{row['kor']}₩t{row['eng']}₩t{row['mat']}₩t{row['tot']}₩t{row['avg']:.1f}₩t{row['grade']}")
```

```
  print('조건 검색')
  myrow=students.find({'no':3})
  for field in myrow:
      print(field)
```