

스마트 모빌리티 프로그래밍

## Ch 8. 파일 입출력, 데이터 분석



영남대학교 정보통신공학과  
교수 김 영 탁

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

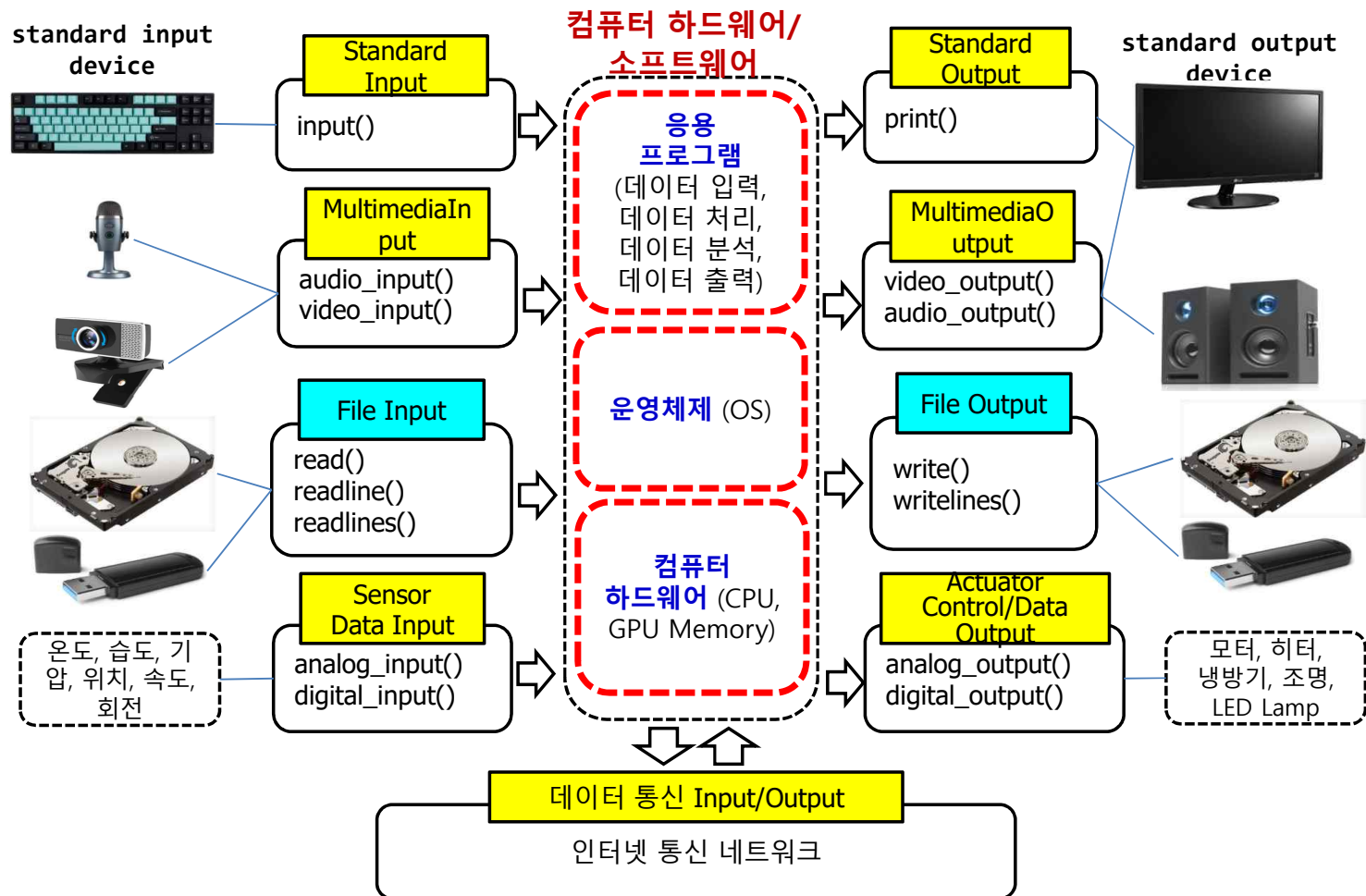
# Outline

- ◆ 파일 입력과 출력
- ◆ 파일 입출력을 위한 io모듈
- ◆ 텍스트 파일 (text file) 입력과 출력
- ◆ 파일 및 디렉토리 (directory) 관리
- ◆ 파일 입출력 기능 활용 예
- ◆ 이진파일 (binary file) 입력과 출력
- ◆ 데이터 분석
- ◆ 시계열 데이터 분석



## 파일 입력과 출력

# 프로그램의 입력과 출력



# 컴퓨터 입력 장치

컴퓨터 입력 장치	설 명
키보드	한글 및 영문, 숫자, 특수 기호 등을 문자 단위로 입력 컴퓨터 사용자의 수작업에 의한 입력
마우스	화면상에서 GUI (graphic user interface)를 사용하기 위한 마우스 이동 및 좌, 우측 버튼 클릭으로 이벤트 생성
터치 스크린	스크린에 펜 또는 손가락으로 터치하여 입력
스캐너 (scanner)	흑백 또는 컬러 문서의 스캔 입력에 사용
카메라	정지영상 또는 동영상의 입력에 사용
마이크로폰	오디오 및 음성 입력에 사용
디스크 저장장치	텍스트 또는 이진 파일의 저장 및 입력
인터넷 통신 모듈	인터넷 통신을 통한 원격 파일 전송/수신
사물인터넷 센서	사물인터넷 단말 장치에 연결된 센서 (온도, 습도, GPS, 가속도, 전압, 거리, 움직임 등)



## 컴퓨터 출력장치

컴퓨터 출력 장치	설 명
모니터	컴퓨터 화면에 텍스트, 사진, 동영상 등으로 출력
프린터	문서 형태로 출력
스피커	오디오 및 음성 출력
디스크 저장장치	텍스트 또는 이진 파일의 저장
인터넷 통신 모듈	인터넷 통신을 통한 원격 파일 전송
사물인터넷 액추에이터	사물인터넷 단말장치에 연결된 모터, 히터, 냉방기 (에어콘), 디스플레이 등 구동장치

# 파일

## ◆ 텍스트 파일 (text file)

- 문자열 자료형의 객체로 입력 및 출력
- 인코딩 (encoding) 방식이 설정됨: ASCII, UNICODE
- 주로 문서 편집기 (e.g., MS-Word, HWP, Eclipse, Visual Studio)를 사용하여 작성됨

## ◆ 이진 파일 (binary file)

- 바이트 객체 (byte object) (예: bytes, bytearray, memoryview, array.array) 로 입력 및 출력
- 영상 파일: BMP, JPG, PNG
- 멀티미디어 파일: AVI, MPEG, MP3/MP4
- 실행 파일 (execution file)

## 파일 저장 장치

컴퓨터 파일 저장 장치	설 명
컴퓨터 메모리	RAM (random access memory), ROM (read only memory) RAM의 경우 전원이 끊어지면 데이터가 손실됨 ROM의 경우 전원이 끊어져도 데이터가 유지됨
Flash 메모리	소비 전력이 작고, 전원이 끊어져도 저장된 데이터가 손실되지 않고 유지됨. 데이터 입출력이 자유로워 디지털 카메라, 캠코더, 휴대전화 등 다양한 휴대장치에서 사용
하드 디스크 드라이브 (HDD)	비휘발성 컴퓨터 보조 기억장치이며, HDD (hard disk driver)의 디스크 상에 자기장 (magnetic field) 물질에 데이터를 저장하며, 전원이 끊어져도 데이터가 유지됨
USB 외장 디스크	USB (universal serial bus) 접속 기능을 사용하여 컴퓨터에 연결된 외장 디스크 장치. HDD보다 데이터 입력 및 출력 속도를 높은 SSD (solid state disk) 방식을 주로 사용



# 파일 입출력

## ◆ 파일 입출력 종류

파일 입출력	설 명
텍스트 파일 입출력	화면으로 출력이 가능한 문자들로 구성된 문서 (text) 자료의 입력 및 출력. 파이썬의 텍스트 파일 인코딩 방식으로 ASCII, 유니코드 등을 사용.
이진 파일 입출력	사진, 동영상, 오디오 등의 binary file 입력 및 출력 JPEG, MP3, MP4, MPEG 등의 압축 표준을 사용

## ◆ 파이썬 객체의 인터넷 정보 전달용 표준

파이썬의 인터넷 정보 전달용 표준	설 명
json	JSON (JavaScript Object Notation)
pickle	파이썬 클래스와 자료형 객체를 binary 형태로 저장

# 파일 입출력을 위한 파이썬 IO 모듈

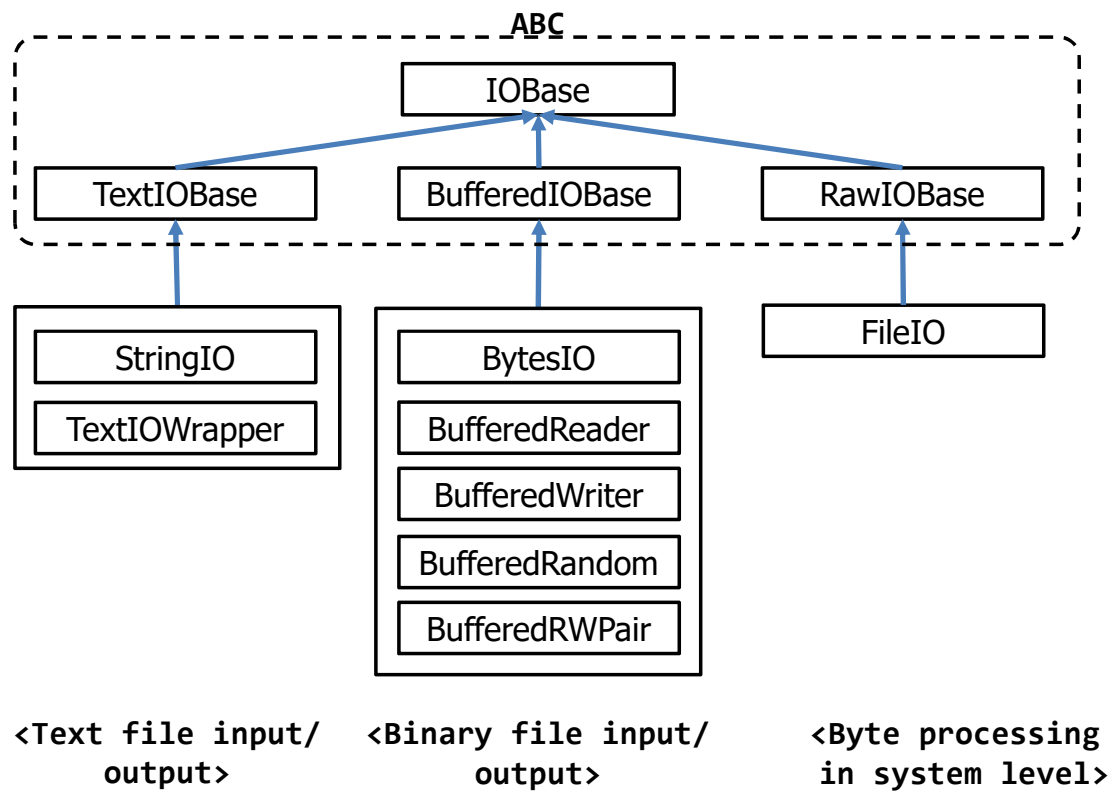
# 파일 입출력을 위한 파이썬 IO 모듈

## ◆ 파이썬 파일 입출력 관련 기본 함수

io 모듈의 텍스트 파일 입출력 함수	설 명
open()	입출력을 위하여 파일의 열기 (open)
close()	입출력 실행이 완료된 파일을 닫기 (close)
seek()	입력 및 출력 기능이 실행되는 지점으로 이동
tell()	파일의 시작위치로부터 현재 파일 객체의 바이트 위치를 알려줌
read()	파일로부터 읽기
readline()	파일을 줄 단위로 읽기
readlines()	파일에서 여러 줄 읽기
write()	파일에 쓰기
writelines()	파일에 줄 단위로 쓰기
flush()	파일 출력에서 버퍼에 쌓인 문자열을 파일로 밀어 냄

# 파이썬 IO 모듈 클래스

## ◆ IO 모듈 클래스 체계



# 파일 객체와 open()

## ◆ 파일 객체 (file object)

- 파일 입력 및 출력에서 사용되는 객체
- 다양한 파일 저장 장치와 통신 장치로부터 파일 입력 및 출력

## ◆ 파일 객체 open()

- open() 함수를 사용한 파일 객체 개방

```
open(file, mode='r', buffering=-1, encoding=None,  
      errors=None, newline=None, closefd=True, opener=None)
```

- 파일 입출력 모드와 버퍼링 방식에 따라 적절한 파일 객체를 반환

# 파일 open() 모드 및 관련 파일 객체

## ◆ 파일 open()에서의 모드 지정 및 관련 파일 객체

파일 종류	모드 설정	파일 오픈 모드	사용되는 파일 객체
텍스트 파일	'r'	읽기 (read)	TextIOWrapper
	'r+'	읽기, 쓰기 및 update	
	'w'	쓰기 (write)	TextIOWrapper
	'w+'	쓰기, 읽기 및 update	
	'a'	추가 (append)	
	'a+'	추가, 읽기 및 update	
이진 파일	'rb'	이진 파일 읽기 (read)	BufferedReader
	'rb+'	이진 파일 읽기 및 update	BufferedRandom
	'wb'	이진 파일 쓰기 (write)	BufferedWriter
	'wb+'	이진 파일 쓰기 update	BufferedRandom
	'ab'	이진 파일 추가 (append)	
	'ab+'	이진 파일 추가 및 update	BufferedRandom

# 파일 open() 함수

## ◆ open() (1)

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,  
      newline=None, closefd=True, opener=None)
```

- file : open될 파일의 서술자 (file descriptor)
- mode :
  - 기본 모드 (text file, read mode)
  - r: read mode
  - w: write mode; 만약 파일이 존재하지 않으면 새로운 파일이 생성됨; 만약 파일이 존재하면 그 파일을 삭제하고 새로운 파일을 생성함
  - x: 파일이 존재하지 않는 경우에만 새로운 파일을 생성함; 만약 파일이 존재하면 에러가 발생됨
  - a: 추가 (append) 모드
  - b: 이진 파일 모드 (binary mode)
  - t: 텍스트 모드 (text mode)
  - +: r+, w+, x+, a+
  - wb+: 이진파일 읽기 및 쓰기 모드, 이미 존재하는 파일은 삭제
  - rb+: 이진 파일 읽기; 파일이 존재하여야 함

# open()

## ◆ open() (2)

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,  
      newline=None, closefd=True, opener=None)
```

- buffering (버퍼링)
  - 버퍼링 정책을 설정
  - 0 : 버퍼링을 사용하지 않도록 설정 (unbuffered), 이진 파일 입출력에서만 사용
  - 1 : 한 줄 단위 버퍼링 (line buffering), 텍스트 파일 입출력에서 사용
  - > 1 : 버퍼 크기를 설정
  - -1 : 기본 버퍼링 기능을 사용; io.DEFAULT\_BUFFER\_SIZE 상수가 기본 버퍼 크기를 설정



# open()

## ◆ open() (3)

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,  
      newline=None, closefd=True, opener=None)
```

- encoding (인코딩)
  - 텍스트 파일 입출력에서 파일 인코딩과 디코딩을 설정
  - 기본 인코딩 방식은 프로그램이 실행되는 컴퓨터 장치에 설정되어 있음
  - `locale.getpreferredencoding()` 함수를 사용한 인코딩 방식 확인할 수 있음
  - 인코딩 설정은 코덱 (CODEC) 모듈을 참조함
- errors
  - 인코딩에서 발생하는 에러를 처리하는 방법을 설정 (예: 'strict', 'ignore')
- newline (줄바꿈)
  - 텍스트 모드에서 줄바꿈을 처리하는 방법을 설정
  - 읽기 모드에서 `newline=None`인 경우 : '\n', '\r', '\r\n' → '\n'
  - 쓰기 모드에서 `newline=None`인 경우 : '\n' 문자는 `os.linesep`에서 설정된 문자로 변경되어 저장 (Windows : '\r\n'; Unix : '\n', Macintosh : '\r')

# open()

## ◆ open() (4)

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,  
      newline=None, closefd=True, opener=None)
```

- closefd
  - True : 파일 입력 및 출력이 완료된 후 파일 닫기 실행
- opener
  - 파일 열기가 실행되기 이전에 수행되는 작업

# close()

## ◆ File close

- 사용이 완료된 파일을 닫기
- 사용 예

```
# Text file open(), write(), read(), close()

f1 = open("data.txt", 'w')
f1.write("Sample text line.\n")
f1.close()

f2 = open("data.txt", 'r')
while True: # repeat until the end of file
    read_data = f2.read()
    if read_data == "":
        break
    print(read_data)

f2.close()
```

## 텍스트 파일 입력과 출력

## 텍스트 파일 Open/Close

### ◆ Text File Open/Close (mode = 'w', 'r')

```
# Text file open(), write(), read(), close()
```

```
f1 = open("data.txt", 'w')
f1.write("Lee 80 90 95\n")
f1.write("Kim 85 75 70\n")
f1.write("Park 70 80 90\n")
f1.write("Hong 90 85 95\n")
f1.write("Yoon 85 85 95\n")
f1.close()
```

```
f2 = open("data.txt", 'r')
while True: # repeat until the end of file
    read_data = f2.read()
    if read_data == '':
        break
    print(read_data)
```

```
f2.close()
```

```
Lee 80 90 95
Kim 85 75 70
Park 70 80 90
Hong 90 85 95
Yoon 85 85 95
```



# Text File Open/Close

## ◆ Text File Open/Close 'with', 'as'

```
# Text file - with ~ as ~

with open("data.txt", 'w') as f1:
    f1.write("Lee 80 90 95\n")
    f1.write("Kim 85 75 70\n")
    f1.write("Park 70 80 90\n")
    f1.write("Hong 90 85 95\n")
    f1.write("Yoon 85 85 95\n")
f1.close()

with open("data.txt", 'r') as f2:
    while True: # repeat until the end of file
        read_data = f2.read()
        if read_data == '':
            break
        print(read_data)
f2.close()
```

Lee	80	90	95
Kim	85	75	70
Park	70	80	90
Hong	90	85	95
Yoon	85	85	95



# tell()과 seek() 함수를 read()와 write() 함수와 사용하여 텍스트 파일 입력과 출력

## ◆ tell(), seek()

```
# Text file - with ~ as ~, tell(), seek() (part 1)
```

```
with open("data.txt", 'w') as f1:
    f1.write("Lee 80 90 95\n")
    f1.write("Kim 85 75 70\n")
    f1.write("Park 70 80 90\n")
    f1.write("Hong 90 85 95\n")
    f1.write("Yoon 85 85 95\n")
f1.close()

f2 = open("data.txt", 'r')
while True: # repeat until the end of file
    read_data = f2.read()
    if read_data == '':
        break
    print(read_data)

print("f2.tell() = ", f2.tell())
f2.seek(0)
print("after f2.seek(0), f2.tell() = ", f2.tell())
```

```
Lee 80 90 95
Kim 85 75 70
Park 70 80 90
Hong 90 85 95
Yoon 85 85 95

f2.tell() = 85
after f2.seek(0), f2.tell() = 0
list_name = ['Lee', 'Kim', 'Park', 'Hong', 'Yoon']
list_kor = [80, 85, 70, 90, 85]
list_eng = [90, 75, 80, 85, 85]
list_math = [95, 70, 90, 95, 95]
student[ 0] = ('Lee', 80, 90, 95)
student[ 1] = ('Kim', 85, 75, 70)
student[ 2] = ('Park', 70, 80, 90)
student[ 3] = ('Hong', 90, 85, 95)
student[ 4] = ('Yoon', 85, 85, 95)
```



```
# Text file - with ~ as ~, tell(), seek() (part 2)
```

```
list_name = []  
list_kor = []  
list_eng = []  
list_math = []
```

```
for line in f2.readlines():  
    name, kor, eng, math = line.split()  
    list_name.append(name)  
    list_kor.append(int(kor))  
    list_eng.append(int(eng))  
    list_math.append(int(math))  
f2.close()
```

```
print("list_name = ", list_name)  
print("list_kor = ", list_kor)  
print("list_eng = ", list_eng)  
print("list_math = ", list_math)  
student_records = list(zip(list_name, list_kor, list_eng, list_math))  
for i in range(len(student_records)):  
    print("student[{:2}] = {}".format(i, student_records[i]))
```

```
Lee 80 90 95  
Kim 85 75 70  
Park 70 80 90  
Hong 90 85 95  
Yoon 85 85 95
```

```
f2.tell() = 85  
after f2.seek(0), f2.tell() = 0  
list_name = ['Lee', 'Kim', 'Park', 'Hong', 'Yoon']  
list_kor = [80, 85, 70, 90, 85]  
list_eng = [90, 75, 80, 85, 85]  
list_math = [95, 70, 90, 95, 95]  
student[ 0] = ('Lee', 80, 90, 95)  
student[ 1] = ('Kim', 85, 75, 70)  
student[ 2] = ('Park', 70, 80, 90)  
student[ 3] = ('Hong', 90, 85, 95)  
student[ 4] = ('Yoon', 85, 85, 95)
```





# 텍스트 파일 입출력 버퍼링과 flush()

## ◆ Buffering과 flush()

```
# text file buffering, flush()

f1 = open("test.txt", 'w')
f1.write("abcdefghij")
print('f1.write("abcdefghij")')

f2 = open("test.txt", 'r')
print("before f1.flush(), f2.read() : ", f2.read())

f1.flush()
print("after f1.flush(), f2.read() : ", f2.read())
f1.close(); f2.close()
```

```
f1.write("abcdefghij")
before f1.flush(), f2.read() :
after f1.flush(), f2.read() :  abcdefghij
```

# readline(), readlines()

## ◆ readline(), readlines()

```
# Student data processing with text file input/output (part 1)
```

```
student_records = [  
    ('Lee', 80, 90, 95),  
    ('Kim', 85, 75, 70),  
    ('Park', 70, 80, 90),  
    ('Hong', 90, 85, 95),  
    ('Yoon', 85, 85, 95)  
]
```

```
def fread_data(file_name):  
    data_list = []  
    f = open(file_name, 'r')  
    for line in f: #f2.readlines()  
        name, kor, eng, math = line.split()  
        tmp = [name, int(kor), int(eng), int(math)]  
        data_list.append(tmp)  
    f.close()  
    return data_list
```



```
# Student data processing with text file input/output (part 2)
```

```
def fwrite_data(file_name, data_list):
```

```
    f = open(file_name, 'w')
```

```
    f.write(" name, kor, eng, math, sum, avg\n")
```

```
    f.write("-----\n")
```

```
    for data in data_list:
```

```
        s = "{0:<8}".format(data[0])
```

```
        s += "{0:>8}".format(data[1])
```

```
        s += "{0:>8}".format(data[2])
```

```
        s += "{0:>8}".format(data[3])
```

```
        s += "{0:4d}".format(data[4])
```

```
        s += "{0:6.2f}".format(data[5])
```

```
        s += '\n'
```

```
        f.write(s)
```

```
    f.close()
```

```
def calculate_score(data_list):
```

```
    i = 0
```

```
    for name, kor, eng, math in data_list:
```

```
        sumScore = kor + eng + math
```

```
        data_list[i].append(sumScore)
```

```
        data_list[i].append(sumScore/3.0)
```

```
        i = i + 1
```



# # Student data processing with text file input/output (part 3)

#####

# Application

f\_st\_name = "student\_records.txt"

f\_st = open(f\_st\_name, 'w')

st\_count = 0

for st in student\_records:

    print("student\_records[{:}] = {}".format(st\_count, st))

    st\_str = "{} {} {} {} \n".format(st[0], st[1], st[2], st[3])

    f\_st.write(st\_str)

    st\_count += 1

f\_st.close()

students = fread\_data(f\_st\_name)

print("\nStudent records read from {} :".format(f\_st\_name))

for st in students:

    print(st)

calculate\_score(students)

print("\nAfter calculate\_student\_score(students)")

for st in students:

    s = ""

    s = "{0:<5s}".format(st[0])

    s += "{0:>3}".format(st[1])

    s += "{0:>3}".format(st[2])

    s += "{0:>3}".format(st[3])

    s += "{0:4d}".format(st[4])

    s += "{0:6.2f}".format(st[5])

    print(s)

fwrite\_data("result\_student.txt", students)

```
student_records[0] = ('Lee', 80, 90, 95)
student_records[1] = ('Kim', 85, 75, 70)
student_records[2] = ('Park', 70, 80, 90)
student_records[3] = ('Hong', 90, 85, 95)
student_records[4] = ('Yoon', 85, 85, 95)
```

Student records read from student\_records.txt :

```
['Lee', 80, 90, 95]
['Kim', 85, 75, 70]
['Park', 70, 80, 90]
['Hong', 90, 85, 95]
['Yoon', 85, 85, 95]
```

After calculate\_student\_score(students)

```
Lee , 80, 90, 95, 265, 88.33
Kim , 85, 75, 70, 230, 76.67
Park , 70, 80, 90, 240, 80.00
Hong , 90, 85, 95, 270, 90.00
Yoon , 85, 85, 95, 265, 88.33
```

Content of result\_student.txt:

name,	kor,	eng,	math,	sum,	avg
Lee	, 80,	90,	95,	265,	88.33
Kim	, 85,	75,	70,	230,	76.67
Park	, 70,	80,	90,	240,	80.00
Hong	, 90,	85,	95,	270,	90.00
Yoon	, 85,	85,	95,	265,	88.33



```
# Student data processing with text file input/output (part 4)
```

```
print("\nContent of {}".format("result_student.txt"))
```

```
f_st = open("result_student.txt", 'r')
```

```
while True:
```

```
    line = f_st.readline()
```

```
    if line == '':
```

```
        break
```

```
    print(line, end='')
```

```
f_st.close()
```

```
student_records[0] = ('Lee', 80, 90, 95)
student_records[1] = ('Kim', 85, 75, 70)
student_records[2] = ('Park', 70, 80, 90)
student_records[3] = ('Hong', 90, 85, 95)
student_records[4] = ('Yoon', 85, 85, 95)
```

```
Student records read from student_records.txt :
```

```
['Lee', 80, 90, 95]
['Kim', 85, 75, 70]
['Park', 70, 80, 90]
['Hong', 90, 85, 95]
['Yoon', 85, 85, 95]
```

```
After calculate_student_score(students)
```

```
Lee , 80, 90, 95, 265, 88.33
Kim , 85, 75, 70, 230, 76.67
Park , 70, 80, 90, 240, 80.00
Hong , 90, 85, 95, 270, 90.00
Yoon , 85, 85, 95, 265, 88.33
```

```
Content of result_student.txt:
```

name,	kor,	eng,	math,	sum,	avg
Lee	,	80,	90,	95,	265, 88.33
Kim	,	85,	75,	70,	230, 76.67
Park	,	70,	80,	90,	240, 80.00
Hong	,	90,	85,	95,	270, 90.00
Yoon	,	85,	85,	95,	265, 88.33



# readlines(), writelines()

```
# Text file readlines() and writelines()
f1 = open("test1.txt", 'w+')
str_lines_1 = ["01234", "abcde", "56789", "ABCED"]
f1.writelines(str_lines_1)
f1.flush()
f1.seek(0)

lines1 = f1.readlines()
for i in range(len(lines1)):
    print("lines1[{:}] = {}".format(i, lines1[i]))
f1.close()

f2 = open("test2.txt", 'w+')
str_lines_2 = ["01234\n", "abcde\n", "56789\n", "ABCED\n"]
f2.writelines(str_lines_2)
f2.flush()
f2.seek(0)
lines2 = f2.readlines()
for i in range(len(lines2)):
    print("lines2[{:}] = {}".format(i, lines2[i]), end='')
f2.close()
```

```
lines1[0] = 01234abcde56789ABCED
lines2[0] = 01234
lines2[1] = abcde
lines2[2] = 56789
lines2[3] = ABCED
```



## encoding, os.path.getsize()

```
# Text IO with various encodings (part 1)
import os
import os.path #for os.path.getsize()
import shutil # for copy()

f = open("data.txt", 'w')
f.write("0123456789ABCDEFGHIJ")
f.close()
f = open("data.txt", 'r')
lines = f.read()
print("contents of data.txt : ", lines)
f.close()

f1 = open("data.txt", 'r') # encoding = 'cp949'
f2 = open("data_utf8.txt", 'w', encoding='utf-8') # 'utf-16', 'utf-16be', 'utf-16le'
f3 = open("data_utf16.txt", 'w', encoding='utf-16')
f2.write(f1.read())
f1.seek(0)
f3.write(f1.read())
f1.flush(); f2.flush(); f3.flush()
f1.close(); f2.close(); f3.close()
print("data_utf8.txt : ", f2)
print("data_utf16.txt : ", f3)
```

```
contents of data.txt : 0123456789ABCDEFGHIJ
data_utf8.txt : <_io.TextIOWrapper name='data_utf8.txt' mode='w' encoding='utf-8'>
data_utf16.txt : <_io.TextIOWrapper name='data_utf16.txt' mode='w' encoding='utf-16'>
```



```
# Text IO with various encodings (part 2)
```

```
f1_size = os.path.getsize("data.txt")
print("size of data.txt = ", f1_size)
f2_size = os.path.getsize("data_utf8.txt")
print("size of data_utf8.txt = ", f2_size)
f3_size = os.path.getsize("data_utf16.txt")
print("size of data_utf16.txt = ", f3_size)

f1 = open("data_utf8.txt", 'r', encoding='utf-8')
f2 = open("data_cp949.txt", 'w', encoding = 'cp949')
f2.write(f1.read())
print(f2)
f2.flush(); f2.close()
f2_size = os.path.getsize("data_cp949.txt")
print("size of data_cp949.txt = ", f2_size)
```

```
contents of data.txt : 0123456789ABCDEFGHIJ
data_utf8.txt : <_io.TextIOWrapper name='data_utf8.txt' mode='w' encoding='utf-8'>
data_utf16.txt : <_io.TextIOWrapper name='data_utf16.txt' mode='w' encoding='utf-16'>
size of data.txt = 20
size of data_utf8.txt = 20
size of data_utf16.txt = 42
<_io.TextIOWrapper name='data_cp949.txt' mode='w' encoding='cp949'>
size of data_cp949.txt = 20
```





## 한글 문장의 파일 입출력

```
# Text IO with various encodings
import os
import os.path #for os.path.getsize()
file_name = "file_kor.txt"
f_kr = open(file_name, 'w')
f_kr.write("가나다라마바사")
f_kr.close()

f = open(file_name, 'r')
lines = f.read()
print("contents of {} : {}".format(file_name, lines))
print("{} : {}".format(file_name, f))
print("size of ({}) = {}".format(file_name, os.path.getsize(file_name)))
f.close()

contents of file_kor.txt : 가나다라마바사
file_kor.txt : <_io.TextIOWrapper name='file_kor.txt' mode='r' encoding='cp949'>
size of (file_kor.txt) = 14
```



**파일 및 디렉토리 관리,  
zipfile, json**

## 파일 및 디렉토리 관리를 위한 파이썬 모듈

Module	Related Methods	설명
os	mkdir()	디렉토리의 생성
	rmdir()	디렉토리의 삭제
	walk()	디렉토리 및 서브 디렉토리에 포함된 모든 파일들의 이름을 반환, top-down 또는 bottom-up으로 실행
	remove()	지정된 파일의 삭제
os.path	exists()	파일 및 디렉토리가 존재하는지 확인
	isfile()	지정된 항목이 파일인지 확인
	isdir()	지정된 항목이 디렉토리인지 확인
	getsize()	지정된 디렉토리 또는 파일의 크기를 확인
	join()	디렉토리 이름과 파일 이름을 결합
shutil	copy()	파일의 복제
	copytree()	디렉토리 및 서브 디렉토리를 복제
	rmtree()	디렉토리 및 서브 디렉토리를 삭제
zipfile	ZipFile()	ZIP 파일을 오픈, 읽기 모드 ('r'), 쓰기 모드 ('w'), 첨부 모드 ('a')
	write()	전달된 파일을 압축하여 지정된 아카이브에 포함시킴
	namelist()	압축 파일에 포함된 파일들의 이름 목록을 반환
	extractall()	압축된 파일의 항목들을 추출
	close()	압축 (아카이브) 파일 파일을 닫음

## 디렉토리 확인, 생성, 복사, 삭제

```
# File and Directory Handling (part 1)

import shutil
import os
import os.path

PyTemp_dir = "C:/PyTemp"
file_1 = "C:/PyTemp/test_file_1.txt"
file_2 = "C:/PyTemp/test_file_2.txt"
if os.path.exists(PyTemp_dir) == False:
    print("Directory {} is not existing, so creating ...".format(PyTemp_dir))
    os.mkdir(PyTemp_dir)

if os.path.exists(file_1) == True:
    print("File already exists, so delete it now.")
    os.remove(file_1)
else:
    print("{} is not existing.".format(file_1))
    print("Creating {}".format(file_1))
    with open(file_1, 'w') as f1:
        f1.write("Test file 1.")
    file_size = os.path.getsize(file_1)
    print("File size of {} = {}".format(file_1, file_size))
    f1.close()
```

```
Directory C:/PyTemp is not existing, so creating ...
C:/PyTemp/test_file_1.txt is not existing.
Creating C:/PyTemp/test_file_1.txt
File size of C:/PyTemp/test_file_1.txt = 12
Test file 1.
```



## # File and Directory Handling (part 2)

```
print("\nCopy from {} to {}".format(file_1, file_2))
shutil.copy(file_1, file_2)
file_size = os.path.getsize(file_2)
print("File size of {} = {}".format(file_2, file_size))

print("\nFiles in {} :".format(PyTemp_dir))
for dirName, subDirList, fnames in os.walk(PyTemp_dir):
    for fname in fnames:
        print(os.path.join(dirName, fname))

print("\nDelete both test files and directory")
os.remove(file_1)
os.remove(file_2)
os.rmdir(PyTemp_dir)
```

```
Directory C:/PyTemp is not existing, so creating ...
C:/PyTemp/test_file_1.txt is not existing.
Creating C:/PyTemp/test_file_1.txt
File size of C:/PyTemp/test_file_1.txt = 12
Test file 1.
```

```
Copy from C:/PyTemp/test_file_1.txt to C:/PyTemp/test_file_2.txt
File size of C:/PyTemp/test_file_2.txt = 12
```

```
Files in C:/PyTemp :
C:/PyTemp\test_file_1.txt
C:/PyTemp\test_file_2.txt
```

```
Delete both test files and directory
```



## 파일 압축 – ZipFile()

```
# ZipFile - ZipFile(), write(), namelist(), close()

import os
import os.path
import shutil
import zipfile

PyTemp_dir = "C:/PyTemp"
file_1 = "C:/PyTemp/test_file_1.txt"
file_2 = "C:/PyTemp/test_file_2.txt"
if os.path.exists(PyTemp_dir) == False:
    print("Directory {} is not existing, so creating ...".format(PyTemp_dir))
    os.mkdir(PyTemp_dir)

if os.path.exists(file_1) == False:
    with open(file_1, 'w') as f1:
        f1.write("Test file 1.")
    f1.close()

if os.path.exists(file_2) == False:
    print("\nCopy from {} to {}".format(file_1, file_2))
    shutil.copy(file_1, file_2)
```



```

# ZipFile - ZipFile(), write(), namelist(), close()

print("\nFiles in {} :".format(PyTemp_dir))
for dirName, subDirList, fnames in os.walk(PyTemp_dir):
    for fname in fnames:
        print(os.path.join(dirName, fname))

zf_name = "Q:/PyTemp_zip.zip"
zf = zipfile.ZipFile(zf_name, mode='w', compression=zipfile.ZIP_DEFLATED)
zf.write(file_1)
zf.write(file_2)
zf.close()

zf = zipfile.ZipFile(zf_name, mode='r')
files = zf.namelist()
print("type(files) = ", type(files))
for file in files:
    print("file in zf : ", file)
zf.close()

print("\nDelete both test files and directory")
os.remove(file_1)
os.remove(file_2)
os.rmdir(PyTemp_dir)
os.remove(zf_name)

```

```

Files in C:/PyTemp :
C:/PyTemp/test_file_1.txt
C:/PyTemp/test_file_2.txt
type(files) = <class 'list'>
file in zf :  PyTemp/test_file_1.txt
file in zf :  PyTemp/test_file_2.txt

```

```

Delete both test files and directory

```



# json 모듈

```
# json module for text file
import os
import os.path
import json # JavaScript Object Notation
```

```
ListSize = 10000
L1 = list(range(ListSize))
print("L1[:10] = ", L1[:10])
print("L1[ListSize-10:ListSize:1] = \n", L1[ListSize-10:ListSize:1])
f1 = open("L_json.txt", 'w')
json.dump(L1, f1)
f1.close()
f1_size = os.path.getsize("L_json.txt")
print("size of L_json.txt : ", f1_size)
print()
```

```
f2 = open("L_json.txt", 'r')
L2 = json.load(f2)
print("L2[:10] = ", L2[:10])
print("L2[ListSize-10:ListSize:1] = \n", L2[ListSize-10:ListSize:1])
f2.close()
```

```
L1[:10] = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
L1[ListSize-10:ListSize:1] =
[9990, 9991, 9992, 9993, 9994, 9995, 9996, 9997, 9998, 9999]
size of L_json.txt : 58890
```

```
L2[:10] = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
L2[ListSize-10:ListSize:1] =
[9990, 9991, 9992, 9993, 9994, 9995, 9996, 9997, 9998, 9999]
```





**이진 파일 입력 및 출력**  
**(Binary File Input/Output)**

# Binary File Input/Output

## ◆ seek(), read(), tell()

```
# Binary file open(), close(), write(), seek(), read(), tell() (part 1)
import os
import os.path
```

```
TEST_STR_BYTES = b"0123456789abcdefghijklmnopqrstuvwxyz"
print("TEST_STR_BYTES = ", TEST_STR_BYTES)
f_bin = open("test.bin", 'wb+')
f_bin.write(TEST_STR_BYTES)
f_bin.flush()
f_bin_size = os.path.getsize("test.bin")
print("f_bin ({{}}) with size of {{}} bytes.\
      format(f_bin, f_bin_size))
for i in range(f_bin_size):
    f_bin.seek(i)
    ch = f_bin.read(1)
    print("f_bin[{:2d}]: {{}} ".format(i, ch), end=" ")
    if ((i+1) % 5) == 0:
        print()
print()
```

```
SEEK_POS = 10
print("f_bin.seek({{}}) : {}".format(SEEK_POS, f_bin.seek(SEEK_POS)))
ch = f_bin.read(1)
```

```
TEST_STR_BYTES = b'0123456789abcdefghijklmnopqrstuvwxyz'
f_bin (<io.BufferedRandom name='test.bin>) with size of 36 bytes
f_bin[ 0]: b'0' f_bin[ 1]: b'1' f_bin[ 2]: b'2' f_bin[ 3]: b'3' f_bin[ 4]: b'4'
f_bin[ 5]: b'5' f_bin[ 6]: b'6' f_bin[ 7]: b'7' f_bin[ 8]: b'8' f_bin[ 9]: b'9'
f_bin[10]: b'a' f_bin[11]: b'b' f_bin[12]: b'c' f_bin[13]: b'd' f_bin[14]: b'e'
f_bin[15]: b'f' f_bin[16]: b'g' f_bin[17]: b'h' f_bin[18]: b'i' f_bin[19]: b'j'
f_bin[20]: b'k' f_bin[21]: b'l' f_bin[22]: b'm' f_bin[23]: b'n' f_bin[24]: b'o'
f_bin[25]: b'p' f_bin[26]: b'q' f_bin[27]: b'r' f_bin[28]: b's' f_bin[29]: b't'
f_bin[30]: b'u' f_bin[31]: b'v' f_bin[32]: b'w' f_bin[33]: b'x' f_bin[34]: b'y'
f_bin[35]: b'z'
f_bin.seek(10) : 10
10-th character in file test.bin : b'a'
f_bin.tell() : 11
f_bin.seek(2, os.SEEK_CUR) : 13
f_bin.read(1) : b'd'
f_bin.tell() : 14
f_bin.seek(-1, os.SEEK_END) : 35
f_bin.read(1) : b'z'
f_bin.seek(-3, os.SEEK_END) : 33
f_bin.read(1) : b'x'
f_bin.tell() : 34
```



```
# Binary file open(), close(), write(), seek(), read(), tell() (part 2)
```

```
print("{}-th character in file test.bin : {}".format(SEEK_POS, ch))
print("f_bin.tell() : ", f_bin.tell())
print("f_bin.seek(2, os.SEEK_CUR) : ", f_bin.seek(2, os.SEEK_CUR))
print("f_bin.read(1) : ", f_bin.read(1))
print("f_bin.tell() : ", f_bin.tell())
print("f_bin.seek(-1, os.SEEK_END) : ", f_bin.seek(-1, os.SEEK_END))
print("f_bin.read(1) : ", f_bin.read(1))
print("f_bin.seek(-3, os.SEEK_END) : ", f_bin.seek(-3, os.SEEK_END))
print("f_bin.read(1) : ", f_bin.read(1))
print("f_bin.tell() : ", f_bin.tell())
f_bin.close()
```

```
TEST_STR_BYTES = b'0123456789abcdefghijklmnopqrstuvwxyz'
f_bin (<io.BufferedRandom name='test.bin') with size of 36 bytes
f_bin[ 0]: b'0'  f_bin[ 1]: b'1'  f_bin[ 2]: b'2'  f_bin[ 3]: b'3'  f_bin[ 4]: b'4'
f_bin[ 5]: b'5'  f_bin[ 6]: b'6'  f_bin[ 7]: b'7'  f_bin[ 8]: b'8'  f_bin[ 9]: b'9'
f_bin[10]: b'a'  f_bin[11]: b'b'  f_bin[12]: b'c'  f_bin[13]: b'd'  f_bin[14]: b'e'
f_bin[15]: b'f'  f_bin[16]: b'g'  f_bin[17]: b'h'  f_bin[18]: b'i'  f_bin[19]: b'j'
f_bin[20]: b'k'  f_bin[21]: b'l'  f_bin[22]: b'm'  f_bin[23]: b'n'  f_bin[24]: b'o'
f_bin[25]: b'p'  f_bin[26]: b'q'  f_bin[27]: b'r'  f_bin[28]: b's'  f_bin[29]: b't'
f_bin[30]: b'u'  f_bin[31]: b'v'  f_bin[32]: b'w'  f_bin[33]: b'x'  f_bin[34]: b'y'
f_bin[35]: b'z'
f_bin.seek(10) : 10
10-th character in file test.bin : b'a'
f_bin.tell() : 11
f_bin.seek(2, os.SEEK_CUR) : 13
f_bin.read(1) : b'd'
f_bin.tell() : 14
f_bin.seek(-1, os.SEEK_END) : 35
f_bin.read(1) : b'z'
f_bin.seek(-3, os.SEEK_END) : 33
f_bin.read(1) : b'x'
f_bin.tell() : 34
```



# ByteIO

## ◆ io.BytesIO class

```
# io.BytesIO

import io

f = io.BytesIO(b"abcdefghij")
print("type(f) = ", type(f))

print("f.getvalue() : ", f.getvalue())
buffer = f.getbuffer()
buffer[2:4] = b'34'
print("after modification, f.getvalue() : ", f.getvalue())

type(f) = <class '_io.BytesIO'>
f.getvalue() : b'abcdefghij'
after modification, f.getvalue() : b'ab34efghij'
```

## **pickle** 모듈을 사용한 이진 파일 입출력

```
# pickle module for binary file
```

```
import pickle
import os.path
```

```
class A:
    name = "ABCDEDEF"
    value = 15
```

```
a = A()
print("a : ", a)
print("a.name = ", a.name)
print("a.value = ", a.value)
```

```
f1 = open("pickle_test.bin", 'wb')
pickle.dump(a, f1)
f1.close()
f1_size = os.path.getsize("pickle_test.bin")
print("size of pickle_test.bin : ", f1_size)
```

```
f2 = open("pickle_test.bin", 'rb')
b = pickle.load(f2)
print("b : ", b)
print("b.name = ", b.name)
print("b.value : ", b.value)
f2.close()
```

```
a :  <__main__.A object at 0x02A09688>
a.name =  ABCDEDEF
a.value =  15
size of pickle_test.bin :  32
b :  <__main__.A object at 0x02A6A190>
b.name =  ABCDEDEF
b.value :  15
```

## json 파일과 pickle 파일의 비교

```
# Measurement of size (number of bytes) for array and list (part 1)
from array import *
from datetime import *
import random, time, os, json, pickle
```

```
def genRandArray(arr, size):
    for x in range(0, size):
        arr.append(x)
    random.shuffle(arr)
```

```
array_size = 20000
print("Generating an Array of {0} random integer elements ....".format(array_size))
A = array('i')
genRandArray(A, array_size)
print("sys.getsizeof(A) : ", sys.getsizeof(A))

print("Generating a List of {0} random integer elements ....".format(array_size))
L = []
genRandArray(L, array_size)
print("sys.getsizeof(L) : ", sys.getsizeof(L))

#f1 = open("A_json.txt", 'w')
#json.dump(A, f1) # array object cannot be serialized by json
#f1.close()
#f1_size = os.path.getsize("A_json.txt")
#print("size of A_json.txt : ", f1_size)
```

```
Generating an Array of 20000 random integer elements ....
sys.getsizeof(A) : 84056
Generating a List of 20000 random integer elements ....
sys.getsizeof(L) : 89008
size of A_pickle.bin : 80083
size of L_json.txt : 128890
size of L_pickle.bin : 59798
```



```
# Measurement of size (number of bytes) for array and list (part 2)
```

```
f2 = open("A_pickle.bin", 'wb')
pickle.dump(A, f2)
f2.close()
f2_size = os.path.getsize("A_pickle.bin")
print("size of A_pickle.bin : ", f2_size)
```

```
f3 = open("L_json.txt", 'w')
json.dump(L, f3)
f3.close()
f3_size = os.path.getsize("L_json.txt")
print("size of L_json.txt : ", f3_size)
```

```
f4 = open("L_pickle.bin", 'wb')
pickle.dump(L, f4)
f4.close()
f4_size = os.path.getsize("L_pickle.bin")
print("size of L_pickle.bin : ", f4_size)
```

```
Generating an Array of 20000 random integer elements ....
sys.getsizeof(A) : 84056
Generating a List of 20000 random integer elements ....
sys.getsizeof(L) : 89008
size of A_pickle.bin : 80083
size of L_json.txt : 128890
size of L_pickle.bin : 59798
```



시계열 데이터 분석  
(Analysis of Time Series Data)



# 데이터 분석 (data analysis) 이란?

## ◆ 데이터 분석의 기본 기능

- 주어진 데이터의 특성을 파악하는 것
- 평균, 최대값, 최소값, 분산 (variance), 표준 편차 (standard deviation)을 파악

## ◆ 시계열 데이터 분석 (time series data analysis)

- 시계열(time series) 데이터는 관측치가 시간적 순서를 가짐
- 일정 시점에 조사된 데이터를 횡단(cross-sectional) 자료라고 함
- 시계열 데이터의 예: ○○전자 주가, △△기업 월별 매출액, 소매물가 지수, 실업률, 환율
- 시계열 데이터 분석의 목적:
  - 미래 값의 예측: (예) 향후 일주일간 주가 예측, 다음 달 매출액 예측
  - 시계열 데이터의 특성 파악: 경향(trend), 주기(cycle), 계절성(seasonality), 불규칙성(irregularity) 등

# 데이터 분석에 관련된 파이썬 함수

## ◆ 데이터 통계 분석에 관련된 파이썬 함수

모듈	파이썬 내장 함수	설명
파이썬 내장함수	max(L)	반복 가능한 자료형 데이터 L를 전달받아 최댓값을 찾아 반환
	min(L)	반복 가능한 자료형 데이터 L를 전달받아 최솟값을 찾아 반환
	sorted(L)	인수 L을 정렬하여 리스트로 반환
	sum(L)	리스트 L에 포함된 모든 항목들을 합산하여 반환
math	sqrt()	제곱근 계산
pandas	min()	최솟값
	max()	최댓값
	mean()	평균(average)
	median()	중간값
	sum()	합산
	var()	분산 (variance)
	std()	표준편차 (standard deviation)
	quantile()	백분위
	describe()	데이터 프레임의 통계 요약

# 파이썬 내장 함수를 사용한 데이터 통계 분석

## ◆ 평균, 분산, 표준편차 계산

```
# Simple statistics - min, max, avg, var, std_dev
import math
```

```
def statistics(L):
    sum_L, len_L = sum(L), len(L)
    min_L, max_L = min(L), max(L)
    avg_L = sum_L / len_L
    sq_diff_sum = 0
    for i in range(len_L):
        diff = L[i] - avg_L
        sq_diff_sum += pow(diff, 2)
    var = sq_diff_sum / len_L
    std_dev = math.sqrt(var)
    return min_L, max_L, avg_L, var, std_dev
```

```
L = [-2.5, 0.2, 3.4, 3.6, 2.7, 2.2, 1.4, 7.0, 7.9, 3.2, 2.2, 3.3, 2.7, 2.1, 1.6, 0.9, 0.3, 1.4, 3.1, 3.2, 4.6, 1.9, 2.9, 5.0, 6.1, 7.2, 6.7, 6.2, 7.4]
print("List of data : ", L)
```

```
Min, Max, Avg, Var, Std_dev = statistics(L)
print("Min= {:.5.2f}, Max= {:.5.2f}".format(Min, Max))
print("Avg= {:.5.2f}, Var={:.5.2f}, Std_dev={:.5.2f}".format(Avg, Var, Std_dev))
```

```
>>>
= RESTART: C:/MyPyPackage/TextBook - 2020/ch 21 (SW and Computational Thinking)
Data Analysis, NumPy/(1-3) simple statistics - avg, var, std with embedded functions.py
List of data : [-2.5, 0.2, 3.4, 3.6, 2.7, 2.2, 1.4, 7.0, 7.9, 3.2, 2.2, 3.3, 2.7, 2.1, 1.6, 0.9, 0.3, 1.4, 3.1, 3.2, 4.6, 1.9, 2.9, 5.0, 6.1, 7.2, 6.7, 6.2, 7.4]
Min= -2.50, Max= 7.90
Avg= 3.38, Var= 6.00, Std_dev= 2.45
>>>
```



# 시계열 데이터 분석의 예 – 최근 10년간 기온 분석 (1)

## ◆ 기상청 기온 측정 데이터 분석

- <https://data.kma.go.kr>
- 기후통계분석 → 통계분석 → 기온분석
- 자료구분 (일), 시작일자-종료일자, 지역/지점 설정
- 검색
- csv 다운로드



# 시계열 데이터 분석의 예 – 최근 10년간 기온 분석 (2)

## ◆ 대구 지역의 2000년 1월 1일 ~ 2020년 1월 27일 기온 측정 데이터

1	날짜	지점	평균기온(°C)	최저기온(°C)	최고기온(°C)
2	2000-01-01	143	4.7	0	8.5
3	2000-01-02	143	6.5	3.1	11.5
4	2000-01-03	143	2.9	0	6.8
5	2000-01-04	143	2.3	-2.4	7.5
6	2000-01-05	143	4.9	-0.9	9.4
7	2000-01-06	143	6	1.2	9.6
8	2000-01-07	143	-1.7	-4	1.2
9	2000-01-08	143	-0.5	-5.3	4.1
10	2000-01-09	143	0	-1.6	1.9
11	2000-01-10	143	2.4	-1	6.8
12	2000-01-11	143	2.4	-3.7	7.2
13	2000-01-12	143	6.3	4.6	8.4
14	2000-01-13	143	4.9	0.9	7.6
15	2000-01-14	143	1.6	-0.1	5.6
16	2000-01-15	143	1.1	-3.9	6.2
17	2000-01-16	143	3.7	0.3	8.2
18	2000-01-17	143	2.4	-2.6	7.3
19	2000-01-18	143	3	0.6	6.8
20	2000-01-19	143	-1.7	-4.1	0.7
21	2000-01-20	143	-4.6	-6.7	-1.5
22	2000-01-21	143	-3.6	-7.8	1.5
23	2000-01-22	143	-1.6	-6.1	2
24	2000-01-23	143	2.7	-0.6	6.8
25	2000-01-24	143	2.8	0.8	5.7
26	2000-01-25	143	-1.5	-5.2	2.4
27	2000-01-26	143	-4.5	-7.3	-0.8
28	2000-01-27	143	-2.7	-7.2	3
29	2000-01-28	143	-0.7	-7.9	6.4
30	2000-01-29	143	-0.1	-5.5	4.9
31	2000-01-30	143	-0.4	-4.4	3.4
32	2000-01-31	143	-3.8	-7.1	0.2

3662	2010-01-01	143	-3.3	-7.4	1.9
3663	2010-01-02	143	0.5	-5.4	7
3664	2010-01-03	143	-0.8	-3.9	3
3665	2010-01-04	143	-1	-3.1	1.5
3666	2010-01-05	143	-3.1	-6.4	-0.1
3667	2010-01-06	143	-5.4	-8.4	-0.7
3668	2010-01-07	143	-4.5	-7.5	-0.4
3669	2010-01-08	143	-2.8	-8.6	3.2
3670	2010-01-09	143	-0.1	-4.1	4.1
3671	2010-01-10	143	0.6	-2.8	3.3
3672	2010-01-11	143	0.2	-1.9	3.2
3673	2010-01-12	143	-3.2	-5.2	-0.2
3674	2010-01-13	143	-6.3	-8.9	-2.9
3675	2010-01-14	143	-4.3	-8.6	1.2
3676	2010-01-15	143	-1.2	-7	4.8
3677	2010-01-16	143	-0.7	-7.6	5.7
3678	2010-01-17	143	0.4	-7.2	7.4
3679	2010-01-18	143	1.8	-4.4	9.9
3680	2010-01-19	143	3.6	-3.5	10.9
3681	2010-01-20	143	9.1	6.4	13
3682	2010-01-21	143	4.6	-2.3	9.8
3683	2010-01-22	143	-2	-4.4	1.9
3684	2010-01-23	143	-1.5	-4.8	2.9
3685	2010-01-24	143	1.7	-3	7.4
3686	2010-01-25	143	2	-2.2	7.3
3687	2010-01-26	143	-0.5	-4.7	5.7
3688	2010-01-27	143	1.8	-4.2	6.3

7323	2020-01-10	143	2.2	-3.6	9.2
7324	2020-01-11	143	3.3	-0.2	8
7325	2020-01-12	143	2.7	0.2	6.5
7326	2020-01-13	143	2.1	-0.6	5.5
7327	2020-01-14	143	1.6	-0.9	5.8
7328	2020-01-15	143	0.9	-2.8	5.2
7329	2020-01-16	143	0.3	-4.7	7.2
7330	2020-01-17	143	1.4	-2.1	8
7331	2020-01-18	143	3.1	-1.1	8.7
7332	2020-01-19	143	3.2	-3.2	9
7333	2020-01-20	143	4.6	2.4	8.1
7334	2020-01-21	143	1.9	-2	7.9
7335	2020-01-22	143	2.9	0.2	5.7
7336	2020-01-23	143	5	2.7	8.6
7337	2020-01-24	143	6.1	0.5	11.8
7338	2020-01-25	143	7.2	5.4	9.8
7339	2020-01-26	143	6.7	1.7	11.2
7340	2020-01-27	143	6.2	4.5	8.5

# CSV 데이터 읽기 – pandas.read\_csv()

```
# pandas - handling CSV data
```

```
import pandas as pd
```

```
Temp_DG = pd.read_csv("ta_20210113.csv")
print("Temp_DG = \n", Temp_DG)
```

```
Temp_DG =
      Date      Avg      Low      High
0  2000-01-01    4.7    0.0    8.5
1  2000-01-02    6.5    3.1   11.5
2  2000-01-03    2.9    0.0    6.8
3  2000-01-04    2.3   -2.4    7.5
4  2000-01-05    4.9   -0.9    9.4
...      ...      ...      ...
7678 2021-01-08  -10.4  -13.6   -5.8
7679 2021-01-09   -8.0  -11.4   -3.1
7680 2021-01-10   -4.7  -10.8    1.2
7681 2021-01-11   -4.2   -8.5   -0.8
7682 2021-01-12   -1.5   -8.8    4.3

[7683 rows x 4 columns]
```

	A	B	C	D
1	Date	Avg	Low	High
2	2000-01-01	4.7	0.0	8.5
3	2000-01-02	6.5	3.1	11.5
4	2000-01-03	2.9	0.0	6.8
5	2000-01-04	2.3	-2.4	7.5
6	2000-01-05	4.9	-0.9	9.4
7	2000-01-06	6	1.2	9.6
8	2000-01-07	-1.7	-4	1.2
9	2000-01-08	-0.5	-5.3	4.1
10	2000-01-09	0	-1.6	1.9
11	2000-01-10	2.4	-1	6.8
12	2000-01-11	2.4	-3.7	7.2
13	2000-01-12	6.3	4.6	8.4
14	2000-01-13	4.9	0.9	7.6
15	2000-01-14	1.6	-0.1	5.6
16	2000-01-15	1.1	-3.9	6.2
17	2000-01-16	3.7	0.3	8.2
18	2000-01-17	2.4	-2.6	7.3
19	2000-01-18	3	0.6	6.8
20	2000-01-19	-1.7	-4.1	0.7
21	2000-01-20	-4.6	-6.7	-1.5
22	2000-01-21	-3.6	-7.8	1.5
23	2000-01-22	-1.6	-6.1	2
24	2000-01-23	2.7	-0.6	6.8
25	2000-01-24	2.8	0.8	5.7
26	2000-01-25	-1.5	-5.2	2.4
27	2000-01-26	-4.5	-7.3	-0.8
28	2000-01-27	-2.7	-7.2	3

# 시계열 데이터 파일 읽기 및 분석

```
# pandas - handling CSV data

import pandas as pd

Temp_DG = pd.read_csv("ta_20210113.csv")
print("Temp_DG = \n", Temp_DG)
#Avg_temp_DG = Temp_DG['Avg']
#print("Avg_Temp_DG =\n", Avg_temp_DG)

print("Temp_DG.describe() =")
print(Temp_DG.describe())
temp_DG_highest = Temp_DG['High'].max()
print("temp_DG_highest = ", temp_DG_highest)
temp_DG_lowest = Temp_DG['Low'].min()
print("temp_DG_lowest = ", temp_DG_lowest)

Temp_DG_highest_day = Temp_DG[Temp_DG.High >=
temp_DG_highest]
print("Temp_DG_highest_day =\n", Temp_DG_highest_day)

Temp_DG_lowest_day = Temp_DG[Temp_DG.Low <=
temp_DG_lowest]
print("Temp_DG_lowest_day =\n", Temp_DG_lowest_day)
```

```
Temp_DG =
      Date      Avg      Low      High
0  2000-01-01    4.7    0.0    8.5
1  2000-01-02    6.5    3.1   11.5
2  2000-01-03    2.9    0.0    6.8
3  2000-01-04    2.3   -2.4    7.5
4  2000-01-05    4.9   -0.9    9.4
...      ...      ...      ...
7678 2021-01-08  -10.4  -13.6   -5.8
7679 2021-01-09   -8.0  -11.4   -3.1
7680 2021-01-10   -4.7  -10.8    1.2
7681 2021-01-11   -4.2   -8.5   -0.8
7682 2021-01-12   -1.5   -8.8    4.3

[7683 rows x 4 columns]
Temp_DG.describe() =
      Avg      Low      High
count  7681.000000  7683.000000  7682.000000
mean    14.539188    10.013276    19.774863
std     9.533006     9.726442     9.767435
min    -10.400000   -13.900000   -7.600000
25%     6.200000     1.400000    11.400000
50%    15.400000    10.400000    21.100000
75%    22.700000    18.700000    27.900000
max    33.100000    28.600000    39.200000
temp_DG_highest = 39.2
temp_DG_lowest = -13.9
Temp_DG_highest_day =
      Date      Avg      Low      High
6782 2018-07-27    32.4    28.6    39.2
Temp_DG_lowest_day =
      Date      Avg      Low      High
6601 2018-01-27   -5.6   -13.9     2.8
```



## **Homework 8**



# Homework 8.1

8.1 최소 10개 국가의 기본 정보 (국가 이름, 수도 이름, 인구수, 면적) 데이터를 텍스트 파일 (demography.txt)에 준비하고, 이 파일의 국가 기본 정보를 읽어 들인 후 순차적으로 화면에 출력하는 파이썬 프로그램을 작성하라. 한 줄에 한 국가씩 출력할 것. 데이터 파일로 부터 읽어 들인 국가들의 기본 정보에서 인구 수를 기준으로 내림차순 정렬을 하고, 그 순서대로 국가 기본 정보를 출력하는 파이썬 프로그램을 작성하라.

참고:

- 국가별 국토 면적, 위키백과 [https://ko.wikipedia.org/wiki/%EB%A9%B4%EC%A0%81%EC%88%9C\\_%EB%82%98%EB%9D%BC\\_%EB%AA%A9%EB%A1%9D](https://ko.wikipedia.org/wiki/%EB%A9%B4%EC%A0%81%EC%88%9C_%EB%82%98%EB%9D%BC_%EB%AA%A9%EB%A1%9D)
- 국가별 인구 수, 위키백과 [https://ko.wikipedia.org/wiki/%EC%9D%B8%EA%B5%AC%EC%88%9C\\_%EB%82%98%EB%9D%BC\\_%EB%AA%A9%EB%A1%9D](https://ko.wikipedia.org/wiki/%EC%9D%B8%EA%B5%AC%EC%88%9C_%EB%82%98%EB%9D%BC_%EB%AA%A9%EB%A1%9D)
- 국가별 수도 이름 : 위키백과, [https://ko.wikipedia.org/wiki/%EB%82%98%EB%9D%BC\\_%EC%9D%B4%EB%A6%84%EC%88%9C\\_%EC%88%98%EB%8F%84\\_%EB%AA%A9%EB%A1%9D](https://ko.wikipedia.org/wiki/%EB%82%98%EB%9D%BC_%EC%9D%B4%EB%A6%84%EC%88%9C_%EC%88%98%EB%8F%84_%EB%AA%A9%EB%A1%9D)

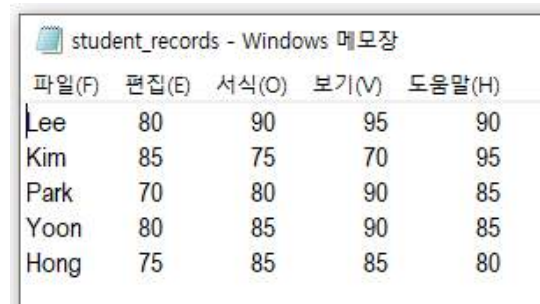
```
List of countries:
Country[ 0] : ('Korea', 'Seoul', 51780579, 220918)
Country[ 1] : ('USA', 'WashingtonDC', 329479633, 983517)
Country[ 2] : ('Canada', 'Ottawa', 36488800, 9984670)
Country[ 3] : ('China', 'Beijing', 1402727120, 9596960)
Country[ 4] : ('India', 'NewDelhi', 1360657785, 3287263)
Country[ 5] : ('Brazil', 'Brazilia', 211349952, 8515767)

List of countries sorted by demography(number of people):
Country[ 0] : ('China', 'Beijing', 1402727120, 9596960)
Country[ 1] : ('India', 'NewDelhi', 1360657785, 3287263)
Country[ 2] : ('USA', 'WashingtonDC', 329479633, 983517)
Country[ 3] : ('Brazil', 'Brazilia', 211349952, 8515767)
Country[ 4] : ('Korea', 'Seoul', 51780579, 220918)
Country[ 5] : ('Canada', 'Ottawa', 36488800, 9984670)
```



## Homework 8.2

8.2 최소 10명의 학생 정보인 (학생 이름, 국어점수, 영어점수, 수학점수, 과학점수) 데이터를 텍스트 파일 student\_records.txt에 준비하고, 이 파일을 읽어 들인 후, 각 학생의 평균점수를 계산하여 학생 정보에 추가하고, 각 과목별로 학생들의 성적을 종합하여 평균 점수를 계산한 후, 결과를 output.txt 텍스트 파일에 출력하는 파이썬 프로그램을 작성하라.



파일(F)	편집(E)	서식(O)	보기(V)	도움말(H)
Lee	80	90	95	90
Kim	85	75	70	95
Park	70	80	90	85
Yoon	80	85	90	85
Hong	75	85	85	80

```
output - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
|name, kor, eng, math, sci, sum, avg
-----
Lee      : 80, 90, 95, 90, 355, 88.75
Kim      : 85, 75, 70, 95, 325, 81.25
Park     : 70, 80, 90, 85, 325, 81.25
Yoon     : 80, 85, 90, 85, 340, 85.00
Hong     : 75, 85, 85, 80, 325, 81.25
```

```
['Lee', 80, 90, 95, 90]
['Kim', 85, 75, 70, 95]
['Park', 70, 80, 90, 85]
['Yoon', 80, 85, 90, 85]
['Hong', 75, 85, 85, 80]
```

```
After calculate_scores(students)
name kor eng math sci sum avg
Lee : 80, 90, 95, 90, 355, 88.75
Kim : 85, 75, 70, 95, 325, 81.25
Park : 70, 80, 90, 85, 325, 81.25
Yoon : 80, 85, 90, 85, 340, 85.00
Hong : 75, 85, 85, 80, 325, 81.25
```

```
Average score of each class :
Kor_avg = 78.00
Eng_avg = 83.00
Math_avg = 86.00
Sci_avg = 87.00
```

## Homework 8.3

8.3 행렬 (matrix)의 초기화 및 덧셈, 뺄셈, 곱셈 및 출력 기능을 메소드로 가지는 class MyMtrx를 파이썬 프로그램으로 작성하고, 이를 myClassMtrx.py에 저장하라. class MyMtrx의 초기화를 담당하는 멤버함수 `__init__(self, name, num_rows, num_cols, list_data)`에는 행렬의 이름 (name), 행의 개수 (num\_rows), 열의 개수 (num\_cols), 원소 데이터를 포함하는 리스트 (list\_data)가 전달된다. 행렬의 덧셈, 뺄셈, 곱셈 연산은 '+', '-', '\*' 연산자를 사용할 수 있도록 연산자 오버로딩 (operator overloading) 함수로 구현하며, `__add__(self, other)`, `__sub__(self, other)`, `__mul__(self, other)`의 함수 원형을 가지며, 연산 결과를 class MyMtrx의 객체로 반환하도록 구현하여야 한다. 행렬의 크기 num\_rows과 num\_columns을 정수 데이터로 지정하며, num\_rows × num\_columns 크기의 행렬 데이터들을 실수 자료형으로 포함하는 텍스트 파일 (matrix\_data.txt)을 준비하라. 텍스트 파일에선 2개의 행렬 데이터를 가지며, 각 행렬마다 행의 개수 (num\_rows)과 열의 개수 (num\_columns) 및 num\_rows×num\_columns개의 실수형 (float) 행렬 원소를 가진다. 텍스트 파일로부터 2개를 행렬 크기 및 행렬 데이터를 읽어class MyMtrx의 객체 mA와 mB에 각각 저장하고,  $mC = mA + mB$ ,  $mD = mA - mB$ ,  $mE = mA \times mB$ 를 각각 계산하여 화면으로 출력하는 파이썬 응용 프로그램 (test\_myClassMtrx.py)을 작성하라.



# Homework 8.2

## 8.3 (계속)

matrix\_data.dat (예)

```
*matrix_data - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
5 5
1.0 2.0 3.0 4.0 5.0
6.0 7.0 8.0 9.0 10.0
11.0 12.0 13.0 14.0 15.0
16.0 17.0 18.0 19.0 20.0
21.0 22.0 23.0 24.0 25.0
5 5
1.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 1.0
```

```
n_row = 5, n_col = 5
mA =
1.0 2.0 3.0 4.0 5.0
6.0 7.0 8.0 9.0 10.0
11.0 12.0 13.0 14.0 15.0
16.0 17.0 18.0 19.0 20.0
21.0 22.0 23.0 24.0 25.0
n_row = 5, n_col = 5
mB =
1.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 1.0
mC = mA + mB =
2.0 2.0 3.0 4.0 5.0
6.0 8.0 8.0 9.0 10.0
11.0 12.0 14.0 14.0 15.0
16.0 17.0 18.0 20.0 20.0
21.0 22.0 23.0 24.0 26.0
mD = mA - mB =
0.0 2.0 3.0 4.0 5.0
6.0 6.0 8.0 9.0 10.0
11.0 12.0 12.0 14.0 15.0
16.0 17.0 18.0 18.0 20.0
21.0 22.0 23.0 24.0 24.0
mE = mA * mB =
1.0 2.0 3.0 4.0 5.0
6.0 7.0 8.0 9.0 10.0
11.0 12.0 13.0 14.0 15.0
16.0 17.0 18.0 19.0 20.0
21.0 22.0 23.0 24.0 25.0
```



## Homework 8.3

## 8.3 시계열 데이터 분석

- 기상청 기온 분석 시계열 데이터 (대구지역, 2000. 1. ~ 2021. 12.)를 <https://data.kma.go.kr>로 부터 csv파일로 다운 받고, 이 csv 파일의 데이터를 입력 받아 월별 온도 분포 (평균, 분산, 표준편차)를 분석하라.
- 입력데이터: <https://data.kma.go.kr> -> 기후통계분석 -> 기온분석  
지역: 대구(143), 기간: 2000. 1. 1. ~ 2021. 12. 31., CSV 파일 다운로드 (daegu\_Jan2000\_Dec2021.csv)
- 출력 결과: 각 월별로 최저기온, 평균기온, 최고기온

