2022-1 컴퓨팅사고와 파이썬 프로그래밍

중간고사 문제풀이

2022. 4. 16.

교수 김 영 탁

영남대학교 정보통신공학과

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

Exam1A

◆ Exam1A. (40점, 시험 시간 (09:00 ~ 09:45) 45분)

(1) 요구사항

- 지정된 개수 (num_data)의 실수 (float) 데이터를 한 줄로 입력받고, 이를 리스트에 담아 반환하여 주는 함수 inputFloatData(num_data)를 작성하라.
- 인수로 전달된 데이터 리스트의 최소값, 최대값, 평균값, 분산 (variance) 및 표준편차(standard deviation)를 계산하여 반환하여 주는 함수 getStatistics(L)을 작성하라. 데이터 리스트에 대한 통계분석 기능구현에서 math 모듈의 sqrt() 함수는 사용가능하나, 그 이외의 파이썬의 내장 함수 (min(), max(), sum())를 사용하지 말고 모두 직접 구현할 것.
- 인수로 전달된 데이터 리스트를 오름차순으로 정렬하여 주는 함수 mergeSort(L)을 작성하라.
- 10개의 실수 데이터 입력, 입력된 데이터의 출력, 입력된 데이터의 통계 분석 결과 출력, 오름차순 정렬된 결과 출력을 차례로 수행하는 main() 함수를 아래 예시와 같이 작성하라.
- 모든 실수 데이터의 출력은 소숫점 이하 2자리까지 출력하도록 할 것.
- 프로그램 출력 첫 부분에 "2022-1 컴사파 Exam1A 학번: 00000000, 성명: 홍길동" 양식으로 본인 학번과 이름을 출력할 것.

(2) main() 함수 (예시)

```
# 기본 주석
. . . 필요한 함수 구현
def inputFloatData(num_data): # 함수는 직접 구현
def printList(name, L): # 함수는 직접 구현
def getStatistics(L): # 함수는 직접 구현
def mergeSort(L): # 함수는 직접 구현, 필요한 추가 함수 구현
def main():
    print("2022-1 컴사파 Exam1A 학번: 22000000, 성명: 홍00")
    L = inputFloatData(10)
    printList("L_data", L)
    L_min, L_max, L_avg, L_var, L_std = getStatistics(L)
    print("L_min = {:.2f}, L_max = {:.2f}, L_avg = {:.2f}, L_var = {:.2f}, L_std = {:.2f}".format(L_min, L_max, L_avg, L_var, L_std))
    L_sorted = mergeSort(L)
    printList("L_sorted", L_sorted)

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

(3) shell 입출력 (예시)

```
2022-1 컴사파 Exam1A 학번: 22000000, 성명: 홍00
input 10 float data : 9.9 8.8 3.3 4.4 5.5 6.6 7.7 1.1 2.2 10.1
L_data =
   9.90   8.80   3.30   4.40   5.50   6.60   7.70   1.10   2.20  10.10
L_min = 1.10, L_max = 10.10, L_avg = 5.96, L_var = 9.16, L_std = 3.03
L_sorted =
   1.10   2.20   3.30   4.40   5.50   6.60   7.70   8.80   9.90  10.10
```

```
# Exam1A - float data input, list, sorting, print_output (1)
# Author :
# Date :
# Brief description :

import math

def inputFloatData(num_data):
    L = list(map(float, input("input {} float data : ".format(num_data)).split(' ')))
    return L

def printList(name, L):
    print("{} = ".format(name))
    for i in range(len(L)):
        print("{:7.2f}".format(L[i]), end='')
    print()

def getStatistics(L):
    L_min = L_max = L_sum = L[0]
    for i in range(1, len(L)):
        if L_min > L[i]:
            L_min = L[i]
        if L_max < L[i]:
            L_max = L[i]
        L_sum += L[i]
    L_avg = L_sum / len(L)
    sum_diff_sq = 0.0
    for i in range(len(L)):
        diff = L[i] - L_avg
        sum_diff_sq += diff * diff
    L_var = sum_diff_sq / len(L)
    L_std = math.sqrt(L_var)
    return L_min, L_max, L_avg, L_var, L_std
```

```
# Exam1A - float data input, list, sorting, print_output (2)

def _merge(L_left, L_right):
    L_res = []
    i, j = 0, 0
    len_left, len_right = len(L_left), len(L_right)
    while i < len_left and j < len_right:
        if L_left[i] < L_right[j]:
            L_res.append(L_left[i])
            i += 1
        else:
            L_res.append(L_right[j])
            j += 1
    while (i < len_left):
        L_res.append(L_left[i])
        i += 1
    while (j < len_right):
        L_res.append(L_right[j])
        j += 1
    return L_res

def mergeSort(L): # merge_sort in increasing order
    if len(L) < 2:
        return L[:]
    else:
        middle = len(L) // 2
        L_left = mergeSort(L[:middle])
        L_right = mergeSort(L[middle:])
        return _merge(L_left, L_right)
```

```
# Exam1A - float data input, list, sorting, print_output (3)

def main():
    print("2022-1 컴사파 Exam1A 학번: 22000000, 성명: 홍00")
    L = inputFloatData(10)
    printList("L_data", L)
    L_min, L_max, L_avg, L_var, L_std = getStatistics(L)
    print("L_min = {:.2f}, L_max = {:.2f}, L_avg = {:.2f}, L_var = {:.2f}, L_std = {:.2f}"₩
          .format(L_min, L_max, L_avg, L_var, L_std))
    L_sorted = mergeSort(L)
    printList("L_sorted", L_sorted)

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

```
2022-1 컴사파 Exam1A 학번: 22000000, 성명: 홍00
input 10 float data : 9.9 8.8 3.3 4.4 5.5 6.6 7.7 1.1 2.2 10.1
L_data =
   9.90   8.80   3.30   4.40   5.50   6.60   7.70   1.10   2.20  10.10
L_min = 1.10, L_max = 10.10, L_avg = 5.96, L_var = 9.16, L_std = 3.03
L_sorted =
   1.10   2.20   3.30   4.40   5.50   6.60   7.70   8.80   9.90  10.10
```

Exam1B

◆ Exam1B. (35점, 시험 시간 (09:45 ~ 10:30) 45분)

(1) 요구사항

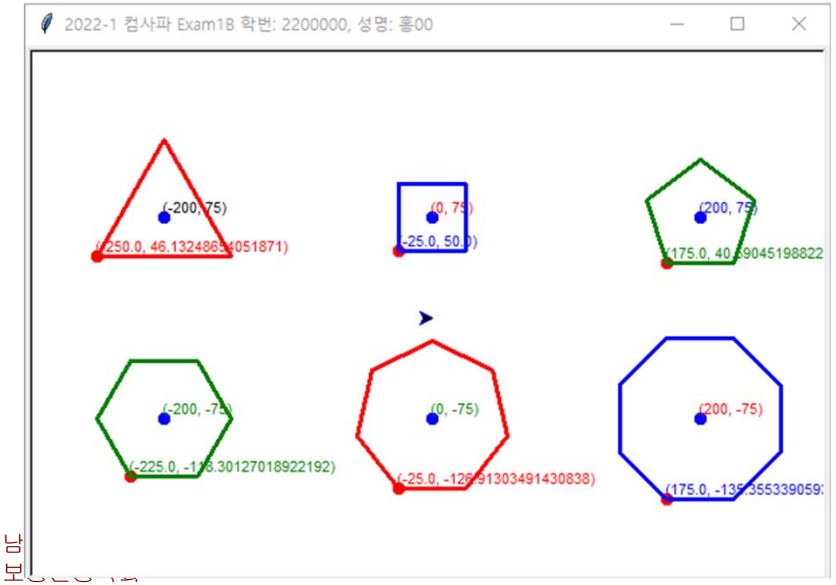
- 다각형의 이름 (polygon_name)을 인수로 전달받아 꼭지점 개수를 반환하는 함수 getNumVertices(polygon_name)을 작성하라. polygon_name으로는 triangle, square, pentagon, hexagon, heptagon, octagon, nonagon, decagon 등이 되도록 할 것.
- 다각형의 꼭지점 개수 (num_vertices), 중심 좌표 (center_x, center_y)를 입력 받고, 주어진 선 길이의 다각형을 지정된 중심점에 그리는 함수 drawPolygon(t, num_vertices, center_x, center_y, line_length, color)을 작성하라. t는 터틀 객체로 전달된다. 도형의 선 굵기는 3으로 할 것.
- 각 다각형의 중심점 (center_x, center_y)에는 굵기 10의 파란색 점을 표시하고, 좌표를 표시하며, 각 다각형의 시작 꼭지점에는 굵기 10의 붉은색 점을 표시하고, 좌표를 표시하라. 시작점 좌표 계산에서는 math 모듈 사용할 것.
- 다각형을 그린 후에는 터틀의 home 위치인 좌표 (0, 0)으로 이동하게 할 것.
- 다각형 정보 (다각형 이름, 중심 좌표, 한 변의 길이, 색상)를 튜플로 구성하고,, 이 튜플을 리스트로 준비하고, drawPolygon() 함수를 사용하여 지정된 위치에 다각형을 그리는 응용 프로그램의 main() 함수를 아래 예시를 참조하여 작성하라.
- 터틀 그래픽의 윈도우 타이틀에 "2022-1 검사파 Exam1B 학번: 00000000, 성명: 홍길동" 양식으로 본인 학번과 이름을 출력할 것.

(2) **main() 함수 (예시)**

```
# 주석문
# 필요한 모듈 추가

def getNumVertices(polygon_name):
  # 함수 내용은 직접 구현

def drawPolygon(t, num_vertices, center_x, center_y, line_length, color):
  # 함수 내용은 직접 구현

def main():
    turtle.setup(600, 400)
    turtle.title("2022-1 컴사파 Exam1B 학번: 2200000, 성명: 홍00")
    t = turtle.Turtle()
    t.shape("classic")
    shapes = [("triangle", -200, 75, 100, "red"), ("square", 0, 75, 50, "blue"),₩
      ("pentagon", 200, 75, 50, "green"), ("hexagon", -200, -75, 50, "green"),₩
      ("heptagon", 0, -75, 50, "red"), ("octagon", 200, -75, 50, "blue")]
    for shape in shapes:
        sh, cx, cy, length, color = shape[0], int(shape[1]), int(shape[2]), int(shape[3]), shape[4]
        num_vertices = getNumVertices(sh)
        drawPolygon(t, num_vertices, cx, cy, length, color)
if __name__ == "__main__":
    main()
```

```
# 2022-1 Exam1B  (1)
# Turtle graphic - Polygons with given center position
# Author :
# Date :
# Brief description :

import turtle
import math

def getNumVertices(shape_name):
    polygons = {"triangle":3, "square":4, "pentagon":5, "hexagon":6, "heptagon":7, "octagon":8, "nonagon":9, "decagon":10}
    num_vertices = polygons[shape_name]
    return num_vertices

def drawPolygon(t, num_vertices, center_x, center_y, line_length, color):
    center_pos = (center_x, center_y)
    line_width = 3
    t.up(); t.goto(center_pos); t.dot(10, "blue"); t.write(center_pos)
    t.width(line_width)
    t.pencolor(color)
    start_x = center_x - line_length/2
    theta = math.radians((180 - 360/num_vertices)/2)
    h = line_length * math.tan(theta) / 2
    start_y = center_y - h
    t.penup(); t.goto(start_x, start_y); t.pendown()
    t.dot(10, "red"); t.write((start_x, start_y))
    for i in range(num_vertices):
        t.forward(line_length)
        #t.dot(10, "red"); t.write(t.pos())
        t.left(360 / num_vertices)
    t.up(); t.home(); t.down()
```

```
# 2022-1 Exam1B  (2)

def main():
    turtle.setup(600, 400)
    turtle.title("2022-1 컴사파 Exam1B 학번: 2200000, 성명: 홍00")
    t = turtle.Turtle()
    t.shape("classic")
    shapes = [("triangle", -200, 75, 100, "red"), ("square", 0, 75, 50, "blue"), ("pentagon", 200, 75, 50, "green"),₩
              ("hexagon", -200, -75, 50, "green"), ("heptagon", 0, -75, 50, "red"), ("octagon", 200, -75, 50, "blue")]
    for shape in shapes:
        sh, cx, cy, length, color = shape[0], int(shape[1]), int(shape[2]), int(shape[3]), shape[4]
        num_vertices = getNumVertices(sh)
        drawPolygon(t, num_vertices, cx, cy, length, color)

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



Exam1C

◆ Exam1C. (35점, 시험 시간 (10:30 ~ 11:15) 45분)

(1) 요구사항

- 도시와 도시 간의 거리 (Km 단위)를 인수로 전달받아 딕셔너리 dict_ICD에 포함시키는 함수 setInterCityDist(dict_ICD, city_1, city_2, dist)를 구현하라. 함수 setInterCityDist()에서는 도시 이름 2개 (city_1, city_2)의 문자열과 그 도시들 간 km 단위의 거리를 정수형 인수 dist로 인수로 전달 받은 후, 딕셔너리 dict_ICD에 {(city_1, city_2) : dist} 형식으로 추가하며, 추가된 도시와 거리 정보를 출력하여 확인하도록 한다.
- 딕셔너리 dict_ICD로부터 두 도시 간의 거리를 찾아 반환하는 함수 getInterCityDist(dict_ICD, city_1, city_2)를 구현하라. 함수 getInterCityDist(dict_ICD, city_1, city_2)에서 인수로 전달되는 dict_ICD는 도시 간 거리 정보를 가지는 딕셔너리이며, city_1과 city_2는 두 개의 도시 이름이다. 두 개의 도시 이름 순서가 바뀌어도 해당 도시간의 거리를 찾아 반환할 수 있게 구현할 것.
- 인수로 전달된 딕셔너리 dict_ICD에 포함된 도시들의 거리표를 아래 예제에서 보는 테이블 형식으로 출력하는 함수 printDistanceTable(dict_ICD)을 작성하라. 거리표는 2차원 배열 형식으로 출력되며, 도시를 나타내는 행과 열에 해당하는 도시간 거리를 출력한다.
- main() 함수에서는 아래 예와 같은 도시간 거리 정보 튜플의 집합 S_ICD을 준비하고, 이 튜플 집합 S_ICD을 출력하도록 하라. 또한 S_ICD에 포함된 정보를 사용하여 setInterCityDist() 함수를 호출하고, 딕셔너리 dict_ICD에 추가하며, 구정된 딕셔너리 dict_ICD를 printDistanceTable()에 전달하여 도시 간의 거리표를 출력하도록 하라. main() 함수는 __name__ 변수가 "__main__"일 때 실행될 수 있도록 구정할 것.
- 프로그램 출력 첫 부분에 본인 학번과 이름을 "2022-1 컴사파 Exam1C 학번: 00000000, 성명: 홍길동" 양식으로 출력할 것.

(2) main() 함수 예시

```
# 주석문
def setInterCityDist(dict_ICD, c1, c2, dist): # 함수는 직접 구현
def getInterCityDist(dict_ICD, c1, c2): # 함수는 직접 구현
def printDistanceTable(dict_ICD): # 함수는 직접 구현
#-------------------------
def main():
    print("2022-1 컴사파 Exam1C 학번: 2200000, 성명: 홍00")
    dict_ICD = dict()
    S_ICD = {("Seoul", "Daejon", 150), ("Daejon", "Daegu", 150), ("Seoul", "Daegu", 300),\
       ("Daegu", "Busan", 130), ("Seoul", "Busan", 430), ("Daegu", "Gwangju", 180),\
       ("Daejon", "Gwangju", 160), ("Seoul", "Gwangju", 310), ("Gwangju", "Busan", 210),\
       ("Daejon", "Busan", 280)}
    print("S_ICD = ", S_ICD)
    for icd in S_ICD:
        (c1, c2, d) = icd
        #print("inter-city distance({}, {}) : {} km".format(c1, c2, d))
        setInterCityDist(dict_ICD, c1, c2, d)
    print("\nInter_City_Distance_Table : ")
    printDistanceTable(dict_ICD)
if __name__ == "__main__":
main()
```

```
2022-1 컴사파 Exam1C 학번: 2200000, 성명: 홍00
S_ICD =  {('Gwangju', 'Busan', 210), ('Daegu', 'Gwangju', 180), ('Seoul', 'Daejon', 150),
('Seoul', 'Gwangju', 310), ('Seoul', 'Daegu', 300), ('Daejon', 'Gwangju', 160), ('Seoul',
'Busan', 430), ('Daejon', 'Daegu', 150), ('Daejon', 'Busan', 280), ('Daegu', 'Busan', 130)
}

Inter_City_Distance_Table :
          |   Gwangju     Busan     Daegu     Seoul    Daejon
----------+---------------------------------------------------
 Gwangju  |         0       210       180       310       160
  Busan   |       210         0       130       430       280
  Daegu   |       180       130         0       300       150
  Seoul   |       310       430       300         0       150
  Daejon  |       160       280       150       150         0
```

```
# Exam1C - Handling Intercity_Distance_Dictionary  (1)
# Author :
# Date :
# Brief description :

def setInterCityDist(dict_ICD, c1, c2, dist):
    dict_ICD[(c1, c2)] = dist
    #print("Inserting (({}, {}) : {}) into dict_ICD".format(c1, c2, dist))
    #print("dict_ICD = ", dict_ICD)

def getInterCityDist(dict_ICD, c1, c2):
    if (c1, c2) in dict_ICD:
        dist = dict_ICD[(c1, c2)]
    elif (c2, c1) in dict_ICD:
        dist = dict_ICD[(c2, c1)]
    else:
        dist = None
    return dist

def printDistanceTable(dict_ICD):
    keys = dict_ICD.keys()
    col_width = 10
    L_city = []
    for key in keys:
        (city_1, city_2) = key
        if city_1 not in L_city:
            L_city.append(city_1)
        if city_2 not in L_city:
            L_city.append(city_2)
    #print("Cities : ", L_city)
```

```
# Exam1C - Handling Intercity_Distance_Dictionary  (2)

    # print inter-city distance table
    print(" " * col_width + "|", end='')
    for city in L_city:
        print("{:>10s}".format(city), end='')
    print("\n"+'-' * col_width + '+', end='')
    for i in range(len(L_city)):
        print("-" * col_width, end='')
    print()
    for i in range(len(L_city)):
        print("{:^10s}|".format(L_city[i]), end='')
        for j in range(len(L_city)):
            dist = getInterCityDist(dict_ICD, L_city[i], L_city[j])
            if (dist == None) and (L_city[i] == L_city[j]):
                print("{:10d}".format(int(0)), end='')
            elif (dist == None) and (L_city[i] != L_city[j]):
                print("{:>10s}".format("unknown"), end='')
            else:
                print("{:10d}".format(dist), end='')
        print()
```

```
# Exam1C - Handling Intercity_Distance_Dictionary  (3)

#-------------------------------------------------
def main():
    print("2022-1 컴사파 Exam1C 학번: 2200000, 성명: 홍00")
    dict_ICD = dict()
    S_ICD = {("Seoul", "Daejon", 150), ("Daejon", "Daegu", 150), ("Seoul", "Daegu", 300), ("Daegu", "Busan", 130),\
             ("Seoul", "Busan", 430),  ("Daegu", "Gwangju", 180), ("Daejon", "Gwangju", 160), ("Seoul", "Gwangju", 310),\
             ("Gwangju", "Busan", 210), ("Daejon", "Busan", 280)}
    print("S_ICD = ", S_ICD)
    for icd in S_ICD:
        (c1, c2, d) = icd
        #print("inter-city distance({}, {}) : {} km".format(c1, c2, d))
        setInterCityDist(dict_ICD, c1, c2, d)
    print("\nInter_City_Distance_Table : ")
    printDistanceTable(dict_ICD)

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

```
2022-1 컴사파 Exam1C 학번: 2200000, 성명: 홍00
S_ICD =  {('Daejon', 'Busan', 280), ('Daejon', 'Daegu', 150), ('Daegu', 'Gwangju', 180),
('Daegu', 'Busan', 130), ('Seoul', 'Gwangju', 310), ('Seoul', 'Daejon', 150), ('Seoul',
'Daegu', 300), ('Gwangju', 'Busan', 210), ('Seoul', 'Busan', 430), ('Daejon', 'Gwangju',
160)}

Inter_City_Distance_Table :
          |    Daejon     Busan     Daegu   Gwangju     Seoul
----------+-------------------------------------------------
  Daejon  |         0       280       150       160       150
  Busan   |       280         0       130       210       430
  Daegu   |       150       130         0       180       300
 Gwangju  |       160       210       180         0       310
  Seoul   |       150       430       300       310         0
```

Exam1D

◆ Exam1D. (40점, 시험 시간 (11:15 ~ 12:00) 45분)

(1) 요구사항

- 행렬 (matrix)의 생성 및 연산에 사용할 수 있는 class Mtrx를 구현하라. class Mtrx에는 생성 및 초기화를 위한 __init__(self, name, n_row, n_col, lst_data), (추가: 행렬 이름을 설정하기 위한 set_name(self, name)), 출력을 위한 __str__(self), 연산자를 사용한 operator-overloading 덧셈 연산을 위한 __add__(self, other), 뺄셈을 위한 __sub__(self, other), 곱셈을 위한 __mul__(self, other) 함수를 포함하도록 하라.
- main() 함수에서는 3 x 5 크기의 2차원 리스트 (data1_3x5, data2_3x5)를 준비하고, 이들을 사용하여 행렬 M1과 M2를 생성하라. 또한, 5 x 3 크기의 2차원 리스트 (data3_5x3)을 준비하여 행렬 M3를 생성하라. 각 행렬은 class Mtrx의 생성자를 사용하여 생성하고, 생성 한 후, 이들을 출력하라.
- main() 함수에서 M4 = M1 + M2, M5 = M1 - M2, M6 = M1 * M3를 각각 실행하고, 그 결과들을 출력하라.
- main() 함수는 __name__ 변수가 "__main__"일 때 실행될 수 있도록 구성할 것.
- 프로그램 출력 첫 부분에 본인 학번과 이름을 "2022-1 컴사파 Exam1D 학번: 00000000, 성명: 홍길동"양식으로 출력할 것.

```
# 주석문

class Mtrx:
# 클래스 직접 구현

#-------------------------
def main():
    print("2022-1 컴사파 Exam1D 학번: 22000000, 성명: 홍00")
    data1_3x5 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
    data2_3x5 = [1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]
    data3_5x3 = [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
    M1 = Mtrx("M1", 3, 5, data1_3x5)
    print("M1 = ", M1)
    M2 = Mtrx("M2", 3, 5, data2_3x5)
    print("M2 = ", M2)
    M3 = Mtrx("M3", 5, 3, data3_5x3)
    print("M3 = ", M3)
    M4 = M1 + M2

    print("M4 = M1 + M2 =", M4)
    M5 = M1 – M2
    print("M5 = M1 – M2 =", M5)
    M6 = M1 * M3
    print("M6 = M1 * M3 =", M6)
if __name__ == "__main__":
    main()
```

```
# Exam1D - class Mtrx, operator overloading and Application Program (1)
# Author :
# Date :
# Brief description :

class Mtrx:
    def __init__(self, name, n_row, n_col, lst_data):
        self.n_row = n_row
        self.n_col = n_col
        lst_row = []
        self.rows = []
        self.name = name
        index = 0
        for i in range(0, self.n_row):
            for j in range(0, self.n_col):
                lst_row.append(lst_data[index])
                index = index + 1
            self.rows.append(lst_row)
            lst_row = []
    def set_name(self, name):  # 이 부분은 추가되었음
        self.name = name

    def __str__(self):
        s = "\n"   # set_name()이 구현된 경우, 아래의 기능 사용
        # s = "{} = \n".format(self.name)
        for i in range(0, self.n_row):
            for j in range(0, self.n_col):
                s += "{:3d}".format(self.rows[i][j])
            s += "\n"
        return s
```

```
# Exam1D - class Mtrx, operator overloading and Application Program (2)

    def __add__(self, other): # operator overloading of '+'
        lst_res = []
        for i in range(0, self.n_row):
            for j in range(0, self.n_col):
                r_ij= self.rows[i][j] + other.rows[i][j]
                lst_res.append(r_ij)
        return Mtrx("R", self.n_row, self.n_col, lst_res)

    def __sub__(self, other): # operator overloading of '-'
        lst_res = []
        for i in range(0, self.n_row):
            for j in range(0, self.n_col):
                r_ij= self.rows[i][j] - other.rows[i][j]
                lst_res.append(r_ij)
        return Mtrx("R", self.n_row, self.n_col, lst_res)

    def __mul__(self, other): # operator overloading of '*'
        lst_res = []
        for i in range(0, self.n_row):
            for j in range(0, other.n_col):
                r_ij= 0
                for k in range(0, self.n_col):
                    r_ij = r_ij + self.rows[i][k] * other.rows[k][j]
                lst_res.append(r_ij)
        return Mtrx("R", self.n_row, other.n_col, lst_res)
```

```
# Exam1D - class Mtrx, operator overloading and Application Program (3)

#--------------------------
def main():    # set_name() 이 구현되지 않은 경우
    print("2022-1 컴사파 Exam1D 학번: 22000000, 성명: 홍00")
    data1_3x5 = [1, 2, 3, 4, 5,  6, 7, 8, 9, 10,  11, 12, 13, 14, 15]
    data2_3x5 = [1, 0, 0, 0, 0,  0, 1, 0, 0,  0,   0,  0,  1,  0,  0]
    data3_5x3 = [1, 0, 0,  0, 1, 0,  0, 0, 1,  0, 0, 0,  0, 0, 0]

    M1 = Mtrx("M1", 3, 5, data1_3x5)
    print("M1 = ", M1)

    M2 = Mtrx("M2", 3, 5, data2_3x5)
    print("M2 = ", M2)

    M3 = Mtrx("M3", 5, 3, data3_5x3)
    print("M3 = ", M3)

    M4 = M1 + M2
    print("M4 = M1 + M2 = ", M4)

    M5 = M1 - M2
    print("M5 = M1 - M2 = ", M5)

    M6 = M1 * M3
    print("M6 = M1 * M3 = ", M6)

if __name__ == "__main__":
    main(
```

```
2022-1 컴사파 Exam1D 학번: 22000000, 성명: 홍00
M1 =
  1  2  3  4  5
  6  7  8  9 10
 11 12 13 14 15

M2 =
  1  0  0  0  0
  0  1  0  0  0
  0  0  1  0  0

M3 =
  1  0  0
  0  1  0
  0  0  1
  0  0  0
  0  0  0

M4 = M1 + M2 =
  2  2  3  4  5
  6  8  8  9 10
 11 12 14 14 15

M5 = M1 - M2 =
  0  2  3  4  5
  6  6  8  9 10
 11 12 12 14 15

M6 = M1 * M3 =
  1  2  3
  6  7  8
 11 12 13
```

Exam1D - class Mtrx, operator overloading and Application Program (3)

```
#-------------------------
def main():    # set_name() 이 구현되어 있는 경우
    print("2022-1 컴사파 Exam1D 학번: 22000000, 성명: 홍00")
    data1_3x5 = [1, 2, 3, 4, 5,  6, 7, 8, 9, 10,  11, 12, 13, 14, 15]
    data2_3x5 = [1, 0, 0, 0, 0,  0, 1, 0, 0,  0,   0,  0,  1,  0,  0]
    data3_5x3 = [1, 0, 0,  0, 1, 0,  0, 0, 1,  0, 0, 0,  0, 0, 0]

    M1 = Mtrx("M1", 3, 5, data1_3x5)
    print(M1)

    M2 = Mtrx("M2", 3, 5, data2_3x5)
    print(M2)

    M3 = Mtrx("M3", 5, 3, data3_5x3)
    print(M3)

    M4 = M1 + M2
    M4.set_name("M4 = M1 + M2");     print(M4)

    M5 = M1 - M2
    M5.set_name("M5 = M1 - M2");   print(M5)

    M6 = M1 * M3
    M6.set_name("M6 = M1 * M3");     print(M6)

if __name__ == "__main__":
    main(
```

```
2022-1 컴사파 Exam1D 학번: 22000000, 성명: 홍00
M1 =
  1  2  3  4  5
  6  7  8  9 10
 11 12 13 14 15

M2 =
  1  0  0  0  0
  0  1  0  0  0
  0  0  1  0  0

M3 =
  1  0  0
  0  1  0
  0  0  1
  0  0  0
  0  0  0

M4 = M1 + M2 =
  2  2  3  4  5
  6  8  8  9 10
 11 12 14 14 15

M5 = M1 - M2 =
  0  2  3  4  5
  6  6  8  9 10
 11 12 12 14 15

M6 = M1 * M3 =
  1  2  3
  6  7  8
 11 12 13
```