

6장 프로그래밍 실습_파이썬 코드

[프로그래밍 실습 1]

```
from __future__ import print_function
from sys import stdin

def printf(str, *args):
    print(str % args, end='')

domain = 10          # 집합의 크기

# setout 함수가 집합을 출력
def setout(A):
    print("\n      A = ", end = "")
    count = 0
    if 1 not in A:
        print("\u2205")
        return
    printf("{")
    for i in range(0,domain):
        if count != 0:
            printf(",")
        printf("%3d"%A[i])
        count = 1

    printf(" }\n")

# readfile 함수가 텍스트 파일에서 관계를 읽고 adjacency에 대입
def readfile(fn, adjacency, nocheck, n2):
    adjacencymx = [[0 for j in range(len(adjacency[0]))] for i in range(len(adjacency))]
    fp = open(fn, 'r')
    printf("%5cf%d=%{' ',nocheck))
    lines = fp.readlines()          # 줄별로 파일을 읽고 lines에 대입
    i=0
    # lines에서의 관계를 adjacency에 대입하고 집합 A에 대한 함수를 출력
    for line in lines:
        values = line.strip('\n').split()
        x = int(values[0])
        y = int(values[1])
        adjacencymx[x][y] = 1
        printf("%2c(%2d, %2d)" %(' ',x,y))
        i += 1
        if i>n2:
            break

    fp.close()
    return adjacencymx

# check 함수가 관계행렬 adjacencymx가 함수인지 아닌지를 확인해서 출력
def check(adjacencymx, nocheck):
    define1=1          # 함수 여부에 대한 플래그

    for i in range(1, domain+1):
        define2 = 0
        nocol = 0
```

```

        for j in range(1, domain+1):
            if adjacencymx[i][j] == 1:
                define2 = 1
                nocol += 1

        # 정의역에 있는 원소가 관계에 포함되지 않았을 때
        if define2 == 0:
            define1 = 0
            printf("        \u25c9 %d \u2209 Dom(f%d)이다.;\n"%(i, nocheck))

        # 정의역에 있는 원소가 두 번 이상 대응될 때
        if nocol > 1:
            printf("        \u25c9 정의역에 있는 원소 %d은 딱 하나만 대응하지 않는다.; \n"%(i))

        # f_ 관계가 함수인지 아닌지를 출력
        if define1 == 1:
            printf("\n f%d은 A\u2a2fA의 부분집합이므로 함수이다.\n\n"%(nocheck))
        else:
            printf("\n 위에 나열된 이유로 f%d은 함수가 아니다.\n\n" %(nocheck))

A = range(1,11)      # 집합 A의 정의 및 초기화
setout(A)            # 집합 A를 출력

fn = "pp6-1-a.dat"   # 텍스트 파일 이름
fn2 = "pp6-1-b.dat"  # 텍스트 파일 이름

# 관계(부울)행렬을 위한 2차원 배열 정의 및 초기화
adjacencymx = [[0 for j in range(domain+1)] for i in range(domain+1)]

nocheck = 1          # 함수 번호 : f1, f2
n1 = 10               # 첫 번째 파일에서의 줄의 개수
n2 = 10               # 두 번째 파일에서의 줄의 개수

# 첫 번째 파일을 읽고 데이터를 adjacencymx에 대입
print("\n    집합 A에 관한 함수 f1은 다음과 같다.:")
adjacencymx = readfile(fn, adjacencymx,nocheck, n1)
printf("{}\n\n")

check(adjacencymx, nocheck)

nocheck += 1
printf("\n\n")

adjacencymx = [[0 for j in range(domain+1)] for i in range(domain+1)]

# 두 번째 파일을 읽고 데이터를 adjacencymx에 대입
print("\n    집합 A에 관한 함수 f2는 다음과 같다.:")
adjacencymx = readfile(fn2, adjacencymx,nocheck, n2)
printf("{}\n\n")

check(adjacencymx, nocheck)

stdin.readline()

```

[프로그래밍 실습 2]

```
# 입력 파일 형식:
# 첫 번째 줄      -> 집합 A의 원소
# 두 번째 줄      -> 집합 B의 원소
# 다음으로 집합 A에서 집합 B로의 관계들 : (예):
# 세 번째 줄      -> a 5
# 네 번째 줄      -> b 2
# .....
# 입력 데이터가 맞는 함수라고 가정하고 단사함수 여부를 판단하는 프로그램

from __future__ import print_function
from sys import stdin

A = []          # 집합 A를 위한 배열 정의
B = []          # 집합 B를 위한 배열 정의

def printf(str, *args):
    print(str % args, end='')

# setout 함수가 집합을 출력
def setout(A):
    count = 0
    if len(A) == 0:
        print("\u2205")
        return
    printf("{")
    for i in range(0, len(A)):
        if count != 0:
            printf(",")
        printf("%2c"%A[i])
        count = 1

    printf(" }\n")

# readfile 함수가 텍스트 파일에서 이항관계를 읽고 단사함수 여부를 반환
def readfile(fn):
    global A, B          # 전역 변수(A와 B)에 접근
    isInjective = True
    fp = open(fn, 'r')   # 파일 열기
    printf("\n%5cf%d={%(' ',1))
    lines = fp.readlines() # 줄별로 파일을 읽고 lines에 대입
    A = lines[0].strip('\n').split() # 첫 번째 줄을 A에 대입
    B = lines[1].strip('\n').split() # 두 번째 줄을 B에 대입
    B_hash = [0] * len(B)

    index = 2            # lines의 세 번째 줄부터 읽기 위한 인덱스

    i=0
    for index in range(2, len(lines)):
        if index < len(lines) - 1:
            values = lines[index].strip('\n').split()
            printf("(%c, %c)," %(values[0], values[1])) # 관계를 출력
            if values[1] in B:
                # 이항관계의 두 번째 원소가 공역에서 두 번 이상 나타나면
                if B_hash[B.index(values[1])] == 1:
                    isInjective = False
                    B_hash[B.index(values[1])] = 1
```

```

        values = lines[index].strip('\n').split()
        printf("(%c, %c)" %(values[0],values[1]))
        fp.close()
        return isInjective

fn = "pp6-2.dat"    # 입력 파일 이름
fp = open(fn, 'r')
lines = fp.readlines()
# 집합 2개를 파일에서 읽고 A와 B에 대입
A = lines[0].strip('\n').split()
B = lines[1].strip('\n').split()
fp.close()

# 집합들을 출력
print("\n      A = ", end = "")
setout(A)
print("\n      B = ", end = "")
setout(B)

print("\n      집합 A에서 집합 B로의 관계 f1은 다음과 같다.:")
isInjective = readfile(fn)
printf("}\n\n")

# 이항관계가 단사함수인지 아닌지 출력
if isInjective == True:
    print(" 주어진 관계가 단사함수이다!\n")
else :
    print(" 주어진 관계가 단사함수가 아니다!\n")

```