1. Python 수·과학 융합 프로젝트 2021

가. 동기 및 목적

4차 산업혁명이 도래하면서 수·과학과 정보 시스템의 연결이 중요해지고 있다. 그리하여 컴퓨터 프로그래밍 언어인 파이썬을 활용하여 수학과 과학에서 배우는 내용들을 검증 또는 구현해 보기로 결심하게 되었다.

나. 활동내용

1) 요세푸스 순열

가) 코드

```
while True:
    k=int(input('사람 수:'))
    n,m=list(map(int,(input('살리는 사람과 제거할 사람 수를
입력하세요(형식:3,2)').split(','))))
    alive=[i for i in range(1,k+1)]

    kill=[]
    index_kill=0

    while len(alive) > 1:
        index_kill += n

        if len(alive)<m:
            print('남은 사람의 수가 제거해야 할 사람의 수보다 적습니다.
최종적으로 남은 사람은 다음과 같습니다.')
            break

        if index_kill >= len(alive):
            index_kill = index_kill%len(alive)

        for p in range(0,m):
            if index_kill >= len(alive):
                index_kill = index_kill%len(alive)
            kill.append( str(alive.pop(index_kill)) )
    print('최종:', alive)
```

나) 코드 설명

1. 변수(variable) 지정 및 리스트(list) 정의

가. 매개변수 k,n,m에 각각 사람 수, 살리는 사람 수, 제거할 사람 수를 저장

나. 원에 둘러앉은 사람을 구현하기 위해 1부터 k까지 숫자가 있는 alive라는 이름의 리스트

생성
 다. 제거하는 사람을 순서대로 저장하기 위해 kill이라는 빈 리스트 생성
 라. alive에서 제거할 index값을 나타내는 변수 index_kill 생성

2. 사람 제거 반복 시행 (리스트 alive의 길이가 1, 즉 남은 사람이 1명일 때까지 계속)
 가. index_kill에 n을 더함
  - 건너뛴 index값의 해당하는 사람은 살아남게 됨
  - 만약 제거할 사람의 수가 리스트 alive의 길이보다 크다면, 제거하는 시행을 더 이상 실행할 수 없으므로 반복문을 종료 및 현재 남아 있는 사람을 바로 출력할 수 있도록 함.
  - 만약 index_kill의 값이 리스트 alive의 길이보다 길다면 원에 둘러 앉아있는 상황이기에 alive의 길이로 나눈 나머지 값을 다시 index_kill에 저장

 나. index_kill에 해당하는 사람을 제거 및 리스트 kill에 추가하는 것을 m번 실행
  - m번 실행함으로써 m명의 사람을 제거하게 됨
  - 만약 index_kill의 값이 리스트 alive의 길이보다 길다면 원에 둘러 앉아있는 상황이기에 alive의 길이로 나눈 나머지 값을 다시 index_kill에 저장
  - pop 메서드(method)를 이용하여 index_kill에 해당하는 사람을 m번 제거하면 최종적으로 리스트 alive에서 index_kill번째부터 index_kill+m번째의 사람을 제거할 수 있도록 구현

3. 리스트 alive 출력

다) 입출력 예시

```
사람 수:30
살리는 사람과 제거할 사람 수를 입력하세요(형식:3,2)2,3
최종: []
사람 수:30
살리는 사람과 제거할 사람 수를 입력하세요(형식:3,2)2,4
남은 사람의 수가 제거해야 할 사람의 수보다 적습니다. 최종적으로 남은 사람은 다음과 같습니다.
최종: [19, 20]
사람 수:41
살리는 사람과 제거할 사람 수를 입력하세요(형식:3,2)3,2
최종: [17]
```

2) DNA의 반보존적 복제

1) 코드

```
#보존 함수 정의
```

```
def conservative(n):
    global dna
    if n ==1:
        for i in range(len(dna)):
            dna.append([14,14])
    if n==2:
        for i in range(len(dna)):
            dna.append([15,15])
    return dna

#반보존 함수 정의
def semiconservative(n):
    global dna
    dna1=[]
    if n==1:
        for i in range(len(dna)-1,-1,-1):
            slice=dna[i]
            dna1.append([14,slice[0]])
            dna1.append([14,slice[1]])
    if n==2:
        for i in range(len(dna)-1,-1,-1):
            slice=dna[i]
            dna1.append([15,slice[0]])
            dna1.append([15,slice[1]])
    return dna1

#분산 함수 정의
def dispersive(n):
    global dna
    dna=list(i/2 for i in dna)
    if n==1:
        dna[0]=dna[0]+0.5
    if n==2:
        dna[1]=dna[1]+0.5
    return dna



#보존/반보존 출력 함수
```

```
def print_dna_count():
    global dna
    n14_14 = dna.count([14,14])
    n14_15 = dna.count([14,15])+dna.count([15,14])
    n15_15 = dna.count([15,15])
    gcd1=ggccdd(n14_14,n14_15,n15_15)

    print("dna는",dna)
    print()
    print("14-14: %d개, 14-15: %d개, 15-15: %d개"%(n14_14, n14_15,
n15_15))
    print("비율 - 14-14 : 14-15 : 15-15 = %d : %d : %d"%(n14_14/gcd1,
n14_15/gcd1, n15_15/gcd1))

#분산형 모델 출력 함수
def print_dna_ratio():
    global dna
    global baji
    dna1=list(int(i*(10**len(baji))) for i in dna)
    gcd1=ggccdd(dna1[0], dna1[1], 0)
    print('한 dna의 14N과 15N은 %f : %f으로 비율이 %d : %d입니다.'
%(dna[0], dna[1], dna1[0]/gcd1, dna1[1]/gcd1) )

#최대공약수 구하는 함수
def ggccdd(n,m,k):
    a=gcd(n,m)
    b=gcd(a,k)
    return b

# 입출력 부분
while True:
    from math import gcd
    t=int(input("보존이면 1, 반보존이면 2, 분산이면 3을 입력하세요:"))
    a= int(input("처음 배지를 입력하세요(14 or 15)"))

    baji=list(map(int,input("차례로 입력(14n 배지 : 1, 15n배지: 2)")))

    if t==1:
```

```
        dna=[]
        dna.append([a,a])
        for i in baji:
            dna=conservative(i)
        print_dna_count()

    if t==2:
        dna=[]
        dna.append([a,a])
        for i in baji:
            dna=semiconservative(i)
        print_dna_count()

    if t==3:
        if a==14:
            dna=[1,0]
        if a==15:
            dna=[0,1]
        for i in baji:
            dna=dispersive(i)
        print_dna_ratio()
```

2) 코드 설명

1. 보존 함수 conservative(n)

- 14n 배지이면(n=1) 리스트 dna에 [14,14]를 추가한다.
- 15n 배지이면(n=2) 리스트 dna에 [15,15]를 추가한다.
- 리스트 dna를 리턴

2. 반보존 함수 semiconservative(n)

- dna1이라는 새로운 리스트 생성
- 14n 배지이면(n=1) 현재 dna 리스트에 있는 요소([a,b]의 형식)들을 하나씩 꺼내어 [14,a]와 [14,b]를 리스트 dna1에 추가한다.
- 15n 배지이면(n=2) 현재 dna 리스트에 있는 요소([a,b]의 형식)들을 하나씩 꺼내어 [15,a]와 [15,b]를 리스트 dna1에 추가한다.
- 리스트 dna를 리턴

3. 분산 함수 dispersive(n)

- 리스트 dna([0,1] 또는 [1,0])의 각 요소를 2로 나눈다.
- 14n 배지이면(n=1) 리스트 dna의 첫 번째 수에 0.5를 더한다.
- 15n 배지이면(n=2) 리스트 dna의 두 번째 수에 0.5를 더한다.

4. 보존/반보존 모델 출력 함수 print_dna_count()

- 보존/반보존 함수에 의하여 만들어진 리스트 dna([[14,14],[14,15],…]의 형식)에서 [14,14], [14,15]와 [15,14], [15,15]의 개수를 각각 세어 출력하고, 이들의 최대공약수를

구한 후 나누어 비율도 출력한다.

5. 분산형 모델 출력 함수 print_dna_ratio()

- 리스트 dna ([float, float]의 형식)에서 두 실수의 최대공약수를 구하여(10의 거듭제곱을 곱하여 정수로 만든 후 시행) 출력한다.

6. 최대공약수 구하는 함수 ggccdd(n,m,k)

- Python 3.7 이하의 버전에서 내장 함수인 최대공약수 함수 (gcd(n,m))가 3개의 수를 받아들이지 못하여 추가한 코드이다.
- n과 m의 최대공약수를 구하고 이와 k의 최대공약수를 다시 구하는 방식이다.

7. 입출력

- 보존, 반보존, 분산 중 어느 가설을 실험할 것인지 변수 t에 입력받는다.
- 처음 배지를 변수 a에 입력받는다.
- 배지 이동 순서를 광역변수 baji에 리스트 형태로 입력받는다.
- t의 값에 따라 해당하는 함수가 실행된다.

3) 입출력 예시

보존이면 1, 반보존이면 2, 분산이면 3을 입력하세요:1
처음 배지를 입력하세요(14 or 15)14
차례로 입력(14n 배지 : 1, 15n배지: 2)1212112
dna는 [[14, 14], [14, 14], [15, 15], [15, 15], [14, 14], [14, 14], [14, 14], [14, 14], [15, 15], [15, 15], [15, 15], [15, 15], [15, 15], [15, 15], [15, 15], [15, 15], [14, 14], [15, 15]]

14-14: 54개, 14-15: 0개, 15-15: 74개
비율 - 14-14 : 14-15 : 15-15 = 27 : 0 : 37

보존이면 1, 반보존이면 2, 분산이면 3을 입력하세요:2
처음 배지를 입력하세요(14 or 15)15
차례로 입력(14n 배지 : 1, 15n배지: 2)1212112

dna는 [[15, 14], [15, 15], [15, 14], [15, 14], [15, 14], [15, 15], [15, 14], [15,
14], [15, 14], [15, 15], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14],
[15, 14], [15, 15], [15, 14], [15, 14], [15, 14], [15, 15], [15, 14], [15, 14], [15,
14], [15, 15], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14],
[15, 15], [15, 14], [15, 14], [15, 14], [15, 15], [15, 14], [15, 14], [15, 14], [15,
15], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 15],
[15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 15], [15,
14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 15], [15, 14],
[15, 14], [15, 14], [15, 15], [15, 14], [15, 14], [15, 14], [15, 15], [15, 14], [15,
14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 15], [15, 14], [15, 14],
[15, 14], [15, 15], [15, 14], [15, 14], [15, 14], [15, 15], [15, 14], [15, 14], [15,
14], [15, 14], [15, 14], [15, 14], [15, 14], [15, 15], [15, 14], [15, 14], [15, 14],
[15, 15], [15, 14], [15, 14], [15, 14], [15, 15], [15, 14], [15, 14], [15, 14], [15,
14], [15, 14], [15, 14], [15, 14], [15, 15], [15, 14], [15, 14], [15, 14], [15, 14],
[15, 14], [15, 14], [15, 14], [15, 15], [15, 14], [15, 14], [15, 14], [15, 14], [15,
14], [15, 14]]

14-14: 0개, 14-15: 106개, 15-15: 22개
비율 - 14-14 : 14-15 : 15-15 = 0 : 53 : 11

보존이면 1, 반보존이면 2, 분산이면 3을 입력하세요:3
처음 배지를 입력하세요(14 or 15)14
차례로 입력(14n 배지 : 1, 15n배지: 2)1212112
한 dna의 14N과 15N은 0.421875 : 0.578125으로 비율이 27 : 37입니다.