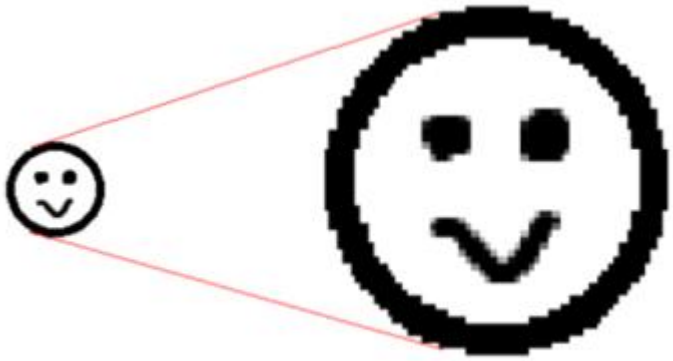
<2주차 Python 과제>

1. 업샘플링

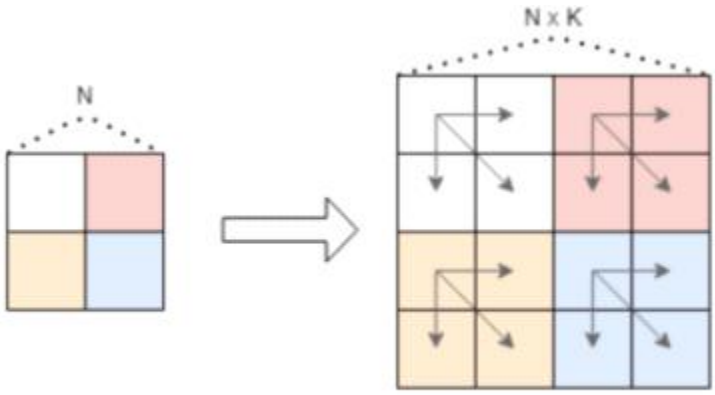
단색 비트맵 이미지에서 이미지를 구성하는 각 픽셀은 0 또는 1의 색상 정보를 가지고 있다. 비트맵 이미지는 생성 당시 이미지의 크기, 해당 이미지를 구성하는 모든 픽셀의 정보가 이미 정해져 있기 때문에, 그림 1과 같이 이미지의 크기를 늘리더라도 기존의 픽셀이 늘린 방향으로 넓게 퍼질 뿐 해상도가 늘어나진 않는다.



< 그림 1. 크기를 늘린 비트맵 이미지의 예시 >

이렇게 기존에 있던 이미지에 픽셀을 추가하여 그림을 구성하는 총 픽셀 수를 늘리는 것을 업샘플링(Up sampling)이라고 한다.

업샘플링을 하는 방법은 여러 가지가 있는데 그 중 가장 간단한 방법은 기존 픽셀의 배열을 그대로 유지한 채, 각 픽셀의 개수를 동일하게 늘리는 방법이다.



<그림 2. 그림을 가로, 세로로 K(=2) 배 늘렸을 때 추가된 픽셀의 모습 >

가로 세로의 길이가 N인 단색 비트맵 이미지를 구성하는 모든 픽셀의 정보가 주어질 때, 해당 그림의 가로와 세로 크기를 그림 2와 같이 K배 늘렸을 때, 업샘플링을 통해 늘어난 그림의 픽셀 정보들을 구해보자.

[입력]

첫 번째 줄에는 정사각형 단색 비트맵의 가로/세로 길이 N(1 <= N <= 10)과 이미지를 늘릴 배수 K(1 <= K <= 10)가 주어진다.

두 번째 줄부터 (N+1)번째 줄에는 각 줄마다 N개의 픽셀 정보가 주어진다.

[출력]

N x K 줄에 걸쳐, 늘어난 단색 비트맵 이미지의 픽셀 정보를 출력한다.

[예제 입력 1]

```
2 2
0 1
1 0
```

[예제 입력 2]

```
3 3
1 0 1
0 0 0
1 0 1
```

[예제 출력 1]

```
0 0 1 1
0 0 1 1
1 1 0 0
1 1 0 0
```

[예제 출력 2]

```
1 1 1 0 0 0 1 1 1
1 1 1 0 0 0 1 1 1
1 1 1 0 0 0 1 1 1
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 1 1 1
1 1 1 0 0 0 1 1 1
1 1 1 0 0 0 1 1 1
```

2. 하노이 탑

세 개의 장대가 있고 첫 번째 장대에는 반경이 서로 다른 n개의 원판이 쌓여 있다. 각 원판은 반경이 큰 순서대로 쌓여 있다. 이제 수도승들은 다음 규칙에 따라 첫 번째 장대에서 세 번째 장대로 옮기려 한다.

1. 한 번에 한 개의 원판만을 다른 탑으로 옮길 수 있다.
2. 쌓아 놓은 원판은 항상 위의 것이 아래의 것보다 작아야 한다.

이 작업을 수행하는데 필요한 이동 순서를 출력하는 프로그램을 작성하라. 단, 이동 횟수는 최소가 되어야 한다. 아래 그림은 원판이 5개인 경우의 예시이다.

[입력]

첫째 줄에 첫 번째 장대에 쌓인 원판의 개수 N(1<=N<=20)이 주어진다.

[출력]

첫째 줄에 옮긴 횟수 K를 출력한다.

두 번째 줄부터 수행 과정을 출력한다. 두 번째 줄부터 K개의 줄에 걸쳐 두 정수 A B를 빈칸을 사이에 두고 출력하는데, 이는 A번째 탑의 가장 위에 있는 원판을 B번째 탑의 가장 위로 옮긴다는 뜻이다.

[예제 입력 1]

```
3
```

[예제 출력 1]

```
7
1 3
1 2
3 2
1 3
2 1
2 3
1 3
```

3. 최소 합계와 최대 합계

5개의 양의 정수가 주어지면 5개의 정수 중 정확히 4개를 더해서 계산할 수 있는 최소값과 최대값을 구해라. 그런 다음 각 최소값과 최대값을 공백으로 구분된 두 개의 정수로 한 줄로 출력한다.

[문제]

아래 코드에서 miniMaxSum 함수를 완성해라.

miniMaxSum에는 다음 매개변수가 있다.

- arr : 배열의 길이가 5인 정수

```
import math
import os
import random
import re
import sys


def miniMaxSum(arr):
    # Write your code here


if __name__ == '__main__':
    arr = list(map(int, input().rstrip().split()))

    miniMaxSum(arr)
```

[입력]

공백으로 구분된 5개의 정수를 한 줄로 입력한다.

[출력]

5개의 정수 중 정확히 4개를 더하여 계산할 수 있는 각각의 최소값과 최대값을 공백으로 구분된 두 개의 정수로 한 줄에 출력한다.

[예시 입력 1]

```
1 2 3 4 5
```

[예시 출력 2]

```
10 14
```

[예시 입력 2]

```
1 3 5 7 9
```

[예시 출력 2]

```
16 24
```

4. Class 계산기

Class 계산기를 만들어보자.

[문제]

AISL에 입사한 혜령은 계산을 원활하게 하기 위해서 class calculator 프로그램을 만들어보려고 한다. 두 수를 입력하면, 기존 기능인 덧셈, 뺄셈, 곱셈, 나눗셈은 기본으로 들어가고 여기에 제곱기능과 최대공약수를 구할 수 있는 기능까지 추가하고 계산 시 에러가 나는 부분까지 해결하려고 한다.

[활용사항]

- 기존 calculator 클래스에 improved_calculator 클래스를 추가할 것.
- 제곱과 최대공약수 기능을 추가하기 위해 파이썬 모듈을 사용할 것.
- 분모에 0이 온 경우 에러가 난다. 이 경우도 고려하여 처리할 것.
- 다음 형식에 맞춰서 작성할 것.

```
class calculator:
    """

    """

class improved_calculator(calculator):
    """

    """

def main():
    """

    """
main()
```

[실행결과]

```
아래에 사용을 원하시는 사칙연산을 선택해주세요!
1. 더하기
2. 빼기
3. 곱하기
4. 나누기
5. 제곱
6. 최대 공약수
7. 종료
>> 1
두 숫자를 입력해주세요: 3 5
8

아래에 사용을 원하시는 사칙연산을 선택해주세요!
1. 더하기
2. 빼기
3. 곱하기
4. 나누기
5. 제곱
6. 최대 공약수
7. 종료
>> 2
두 숫자를 입력해주세요: 3 5
-2

아래에 사용을 원하시는 사칙연산을 선택해주세요!
1. 더하기
2. 빼기
3. 곱하기
4. 나누기
5. 제곱
6. 최대 공약수
7. 종료
>> 4
두 숫자를 입력해주세요: 4 0
0으로 나눌 수 없습니다.
None

아래에 사용을 원하시는 사칙연산을 선택해주세요!
1. 더하기
2. 빼기
3. 곱하기
4. 나누기
5. 제곱
6. 최대 공약수
7. 종료
>> 7
계산기 프로그램을 종료합니다.
```