Project

지능기전공학부 스마트기기공학전공
18011793 이정민

- algorithm for guessing numbers(서버 기준)

```
myguess = [1, 2, 3] #server가 예측한 client의 정답
my_SB = [0, 0] #myguess의 strike, ball 개수
clientguess = [0, 0, 0] #client가 예측한 client의 정답
client_SB = [0, 0] #client의 strike, ball 개수
guess = [1, 2, 3, 4, 5, 6, 7, 8, 9]  #client의 정답이 될 가능성 있는 숫자
data = [] # [myguess][my_SB] 넣어놓는 list -> flag = 10이 되면 [myguess]를 넣어놓는 list
flag = -1 #server의 첫순서 pass 및 client의 정답에 해당하는 3개의 숫자를 찾았는지 판별
```

선언한 변수들은 위의 사진과 같다.(무한루프에 들어가기 전 변수 선언 및 초기화)

1. 먼저 숫자를 예측할 때 1,2,3번째 시도는 각각 123, 456, 789로 고정시킨다.
   그리고 data에 순서대로 [1, 2, 3, S, B], [4, 5, 6, S, B], [7, 8, 9, S, B]를 append한다.
   123, 456, 789를 처음에 시도하는 이유는 각각의 숫자들에 몇 개의 숫자가 포함되어 있는지 알아보기 위함이다.

```
elif len(data) < 3 : #만약 예측한 횟수가 3번째 이하이면
      myguess = [i for i in range(3*len(data)+1,3*(len(data)+1)+1)]
      #첫번째 : [1, 2, 3], 두번째 : [4, 5, 6], 세번째 : [7, 8, 9]
```

```
#3번째 시도까지 그리고 flag = 0일때,  [myguess + my_SB] 추가
if len(data)<3 and flag == 0:
    data.append(myguess+my_SB)
```

   (append코드는 무한루프 안에 있다. 그리고 flag는 서버의 첫순서를 pass시킨 후 0으로 바뀐다.)

2. 4번째 시도부터 [1, 2, 3], [4, 5, 6], [7, 8, 9]에서 각각의 strike + ball 개수만큼 숫자를 random하게 뽑는다. 예를 들어 정답이 348이라면 3개의 그룹에서 1개의 숫자씩 랜덤하게 뽑게 된다.

(data = [[1, 2, 3, S, B], [4, 5, 6, S, B], [7, 8, 9, S, B]])

```
else:
      myguess = random.sample(data[0][0:3], data[0][3]+data[0][4]) #[1, 2, 3]에서 SB갯수만큼 랜덤 뽑기
      myguess = myguess+random.sample(data[1][0:3], data[1][3]+data[1][4]) #[4, 5, 6]에서 SB갯수만큼 랜덤 뽑기
      myguess = myguess+random.sample(data[2][0:3], data[2][3]+data[2][4]) #[7, 8, 9]에서 SB갯수만큼 랜덤 뽑기
```

3. 1, 2번과 별개로 만약 정답에 해당하는 3개의 숫자를 찾았을 경우는 2가지의 경우로 나뉜다.

3-1. 예측한 숫자가 ball이 3개일 경우

즉 예측한 숫자의 자리가 모두 일치하지 않는다는 것이다. 이때 이전에 예측한 숫자의 각각의 자리를 오른쪽으로 swap해준 수로 예측한다. 예를들어 정답이 512였고 예측한 수가 251이었다면 이 수를 오른쪽으로 swap한 125가 그 다음 예측값이 되고 또 그 다음 예측값은 512가 된다.

3-2. 예측한 숫자가 ball이 3개가 아닐 경우

정답에 해당하는 3개의 숫자에 대한 순서를 random하게 뽑아 3자리로 만든다. 하지만 이전의 뽑았던 숫자가 저장되어 있는 data list에 포함되어 있지 않은 수로 선택한다. 즉, 3개의 숫자를 찾게 되면 예측하는 수의 중복을 피한다.

```
if len(guess) == 3: #server의 정답에 해당하는 3개의 숫자를 찾았을 때
      if my_SB[1] == 3 : #내가 예측한 숫자가 ball이 3개인 경우 3자리를 오른쪽으로 swap
            tmp = myguess[0]
            for i in range(2):
                  myguess[i] = myguess[i+1]
            myguess[2] = tmp
      else : #ball 3개가 아닌 경우
            while True : #이전에 예측한 숫자를 제외한 guess list안에서 랜덤으로 3자리 숫자 찾기
                  myguess = random.sample(guess, 3)
                  if myguess not in data :
                        break
```

방금까지 설명한 코드 중 append코드를 제외하고 숫자를 예측하기 위한 guessNumber 함수에서 실행된다. 예측한 myguess 리스트를 반환한다.

```
def guessNumber(guess, my_SB, data, myguess): #숫자 예측
     if len(guess) == 3: #server의 정답에 해당하는 3개의 숫자를 찾았을 때
          if my_SB[1] == 3 : #내가 예측한 숫자가 ball이 3개인 경우 3자리를 오른쪽으로 swap
               tmp = myguess[0]
               for i in range(2):
                    myguess[i] = myguess[i+1]
               myguess[2] = tmp
          else : #ball 3개가 아닌 경우
               while True : #이전에 예측한 숫자를 제외한 guess list안에서 랜덤으로 3자리 숫자 찾기
                    myguess = random.sample(guess, 3)
                    if myguess not in data :
                         break
     elif len(data) < 3 : #만약 예측한 횟수가 3번째 이하이면
          myguess = [i for i in range(3*len(data)+1,3*(len(data)+1)+1)]
          #첫번째 : [1, 2, 3], 두번째 : [4, 5, 6], 세번째 : [7, 8, 9]
     else:
          myguess = random.sample(data[0][0:3], data[0][3]+data[0][4]) #[1, 2, 3]에서 SB갯수만큼 랜덤 뽑기
          myguess = myguess+random.sample(data[1][0:3], data[1][3]+data[1][4]) #[4, 5, 6]에서 SB갯수만큼 랜덤 뽑기
          myguess = myguess+random.sample(data[2][0:3], data[2][3]+data[2][4]) #[7, 8, 9]에서 SB갯수만큼 랜덤 뽑기
     return myguess
```

현재 서버와 클라이언트는 무한 루프 안에서 서로의 정답을 예측하는 상태이다.
무한루프 안에서도 guessNumber를 도와서 점수를 예측하는데 도움이 되는 코드들이 실행된다.

```
if my_SB[0]+my_SB[1] == 0 and flag == 0: #내가 예측한 숫자들 중 strike나 ball이 하나도 없을 경우
      for i in range(3):
            if myguess[i] in guess :
                  del guess[guess.index(myguess[i])] #guess 리스트에서 숫자 삭제
elif my_SB[0]+my_SB[1] == 3 and flag == 0 : #내가 예측한 숫자들의 strike와 ball의 개수가 3일 경우
      guess = myguess #guess list에 myguess list 복사
      data = [] #data list 초기화
      flag = 1 #server의 정답에 해당하는 3개의 숫자 찾았기 때문에 flag = 1

#3번째 시도까지 그리고 flag = 0일때, [myguess + my_SB] 추가
if len(data)<3 and flag == 0:
      data.append(myguess+my_SB)

#[myguess] 추가
if flag == 1 :
      data.append(myguess)
```

1. server가 예측한 숫자들 중에 strike나 ball이 하나도 없을 경우

```
guess = [1, 2, 3, 4, 5, 6, 7, 8, 9]  #client의 정답이 될 가능성 있는 숫자
```

guess는 처음에 이렇게 초기화된다. 처음에는 1~9 모두 정답이 될 가능성이 있는 숫자에 포함된다.
코드를 실행시키다 내가 예측한 숫자들의 strike, ball이 0개일 경우는 모두 정답이 될 가능성이 없는 숫자들로 이루어졌다는 것이다. 따라서 guess 리스트에서 내가 예측한 숫자들을 삭제한다.

2. server가 예측한 숫자들 중에 strike와 ball의 개수가 3일 경우
이 경우는 정답에 해당하는 3개의 숫자를 모두 찾았다는 것이다. 따라서 guess에 내가 예측한 숫자인 myguess 리스트를 복사한다. 그리고 data 리스트를 초기화 시켜준다. 그리고 flag 값을 1로 바꾼다.
위 그림에서 가장 밑의 if문에서 flag가 1이라는 것은 정답에 해당하는 3개의 숫자를 모두 찾았다는 것을 의미한다. 이때 data에는 내가 예측한 3자리 숫자인 myguess 리스트가 append된다. 이 과정을 통해서 위의 3-2에서 예측하는 수의 중복을 피한다.