오픈소스 개발프로젝트 과제 2 - 정보통신공학부 2021039057 이규하

1. 서론

과제 2 요구사항을 반영하여 개선된. 제가 직접 작성한 코드와와 ChatGPT 코드 간의 차이점을 비교하고, 각 코드의 주요 기능을 설명하고자 합니다. 둘 다 동일한 기능을 구현하지만, 코드 구조와 세부적인 구현 방식에서 미미한 차이가 존재합니다. 이를 바탕으로 각 코드의 특성을 분석하고, 기능 구현 측면에서의 장단점을 비교하고자 합니다.

2. 직접 작성한 코드의 전반적인 특징 (code_myself_2021039057 이규하.py)

2.1 코드 기능 요약

- 카드 덱 생성: 52장의 전체 카드를 생성하고 z 리스트에 저장합니다.
- 플레이어 수 입력: playerNum 변수에 1~5명의 플레이어 수를 입력받습니다.
- 카드 배분 및 중복 확인: 각 플레이어에게 무작위로 5장의 카드를 배분하고, 숫자가 중복된 경우 "pair"를 출력합니다.
- 반복 및 종료 처리: Enter를 입력하면 다음 판이 진행되고, z를 입력하면 게임이 종료됩니다.

2.2 코드 구조 및 특징

- 카드 생성: x와 y 배열을 사용하여 52장의 카드 쌍을 z 리스트에 저장합니다.
- 플레이어 수 입력: 1~5명의 플레이어 수를 입력받고, 입력받은 플레이어 수에 따라 카드 배분을 다르게 처리합니다.
- 게임 종료 및 반복: 사용자가 Enter를 입력하면 다음 판이 시작되며, z를 입력하면 게임이 종료됩니다.
- 카드 분배: 입력받은 플레이어 수에 맞게 random.sample(z, k=num_players * 5)로 전체 플레이어에게 중복 없이 카드를 배분합니다.
- 중복 확인 및 pair 출력: 카드의 숫자만을 분리하여 selectedNum 리스트에 저장하고, set(selectedNum)을 통해 숫자 중복 여부를 확인합니다. 숫자가 중복된 경우 "pair"를 출력하여 중복 상태를 나타냅니다.
- 게임 종료 및 반복 처리: 사용자가 Enter 키를 누르면 다음 판이 진행되며, z를 입력하면 게임이 종료됩니다.

3. ChatGPT가 작성한 코드의 전반적인 특징 (code_ChatGPT_2021039057 이규하.py)

3.1 ChatGPT에게 요구한 사항

3.2 코드 기능 요약

- 플레이어 수 입력: 프로그램 시작 시 1~5명 사이의 플레이어 수를 입력받아 num_players 변수에 저장합니다. 잘못된 입력이 발생하지 않도록 유효성 검사를 통해 올바른 범위 내의 숫자만 입력되도록 제한합니다.
- 카드 덱 생성 및 분배: 총 52장의 카드(13개의 숫자 * 4개의 무늬)를 z 리스트에 저장하고, random.sample()을 사용하여 각 플레이어에게 중복되지 않도록 5장씩 카드를 배분합니다. 이때, 모든 플레이어에게 필요한 전체 카드 수를 한 번에 추출하여 효율적으로 카드가 분배됩니다.

- 중복 카드 확인 및 표시: 각 플레이어의 카드 중 숫자가 중복되는 경우 "pair"라는 상태를 표시하여, 중복 여부를 쉽게 확인할 수 있도록 합니다. 중복 확인은 각 카드의 숫자를 추출한 후, count() 함수를 사용하여 이루어집니다.
- 게임 반복 및 종료 처리: 게임은 Enter를 누르면 다음 판으로 넘어가며, z를 입력하면 게임이 종료됩니다. 각 판이 종료될 때마다 새로운 카드를 분배하여 게임을 반복 진행할 수 있습니다.

3.3 코드 구조 및 특징
- 카드 생성 및 초기화: x와 y 배열을 사용하여 각각 13개의 숫자와 4개의 무늬를 조합해 z 리스트에 52장의 카드 덱을 생성합니다. 카드 덱 초기화는 기존 코드와 동일한 방식으로 이루어지며, z 리스트에 모든 카드가 저장됩니다.
- 플레이어 수 입력 및 유효성 검사: while 루프와 try-except 구문을 통해 사용자로부터 플레이어 수를 입력받고, 1~5명 범위 내에서만 유효한 입력으로 간주합니다. 올바른 입력이 들어올 때까지 반복적으로 입력을 요청하여, 프로그램이 예기치 않게 종료되는 것을 방지합니다.
- 카드 분배 및 중복 확인: random.sample(z, k=num_players * 5)를 사용하여 입력된 플레이어 수에 맞게 총 필요한 카드 수(플레이어 수 * 5)를 무작위로 추출합니다. 추출한 카드를 각 플레이어에게 5장씩 분배하고, 숫자 중복 여부를 card_numbers.count()를 사용해 확인합니다. 중복이 발생하면 "pair"라는 상태를 출력하여 알림을 제공합니다.
- 출력 형식 및 게임 종료 처리: 각 플레이어의 카드를 ,로 구분하여 출력하며, 중복이 있는 경우 pair 상태를 함께 출력합니다. input()을 통해 사용자의 입력을 받아 Enter 키가 입력되면 다음 판을 진행하고, z가 입력되면 게임을 종료하는 흐름으로 프로그램을 제어합니다.

4. 두 코드 간의 공통점 및 차이점 분석

4.1 공통점

* 카드 덱 생성 방식
- 카드 덱 생성: 두 코드 모두 52장의 전체 카드를 생성하기 위해 x와 y 리스트를 사용합니다.
  x = ["A", "2", "3", ..., "K"] * 4로 13개의 숫자를 네 번 반복하고, y = ["H", "D", "S", "C"] * 13으로 4개의 무늬를 13번 반복하여 전체 덱을 구성합니다.
  z.append(x[i] + y[i])를 사용하여 52장의 카드를 z 리스트에 저장합니다.
* 플레이어 수 입력 및 제어
- 플레이어 수 입력: 사용자로부터 1~5명 사이의 플레이어 수를 입력받고, 이를 playerNum 또는 num_players 변수에 저장하여 이후 카드 분배에 사용합니다. 플레이어 수를 기반으로 카드 분배를 결정하고, 플레이어 수가 입력되지 않으면 프로그램이 진행되지 않도록 제어합니다.
* 카드 배분 및 중복 확인
- 카드 배분: 두 코드 모두 플레이어별로 5장의 카드를 무작위로 추출하여 플레이어에게 분배합니다.
  random.sample() 함수를 사용하여 덱에서 카드 5장을 비복원 추출하여 중복 없이 카드가 추출되도록 합니다.
- 중복 확인: 각 플레이어의 카드 중 중복된 숫자가 있으면 이를 확인하고 "pair"라는 문구를 출력합니다.
  중복 여부를 확인하기 위해 각 카드의 숫자만 분리하고, 숫자가 중복되는지 확인하는 로직이 포함되어 있습니다.
* 출력 형식 및 게임 반복
- 출력 형식: 두 코드 모두 각 플레이어의 카드를 쉼표(,)로 구분하여 출력하고, 중복이 있는 경우 pair라는 문구를 출력합니다. 플레이어의 카드가 중복 없이 정렬되어 출력되며, 중복 여부에 따라 출력 형식이 약간 달라지는 부분도 동일합니다.
- 게임 반복 및 종료: 게임을 반복적으로 진행하거나 종료하는 조건을 갖추고 있습니다. Enter를 입력하면 다음 판이 진행되고, z를 입력하면 게임이 종료되는 구조를 두 코드 모두 사용합니다.
* 비복원 추출
- 중복 방지: 두 코드 모두 한 판의 게임이 끝난 후에도 이미 분배된 카드를 회수하여 덱으로 돌리지 않습니다.
  매 판 새로운 게임이 진행될 때마다 덱에서 중복되지 않은 새로운 카드가 분배되도록 유지하고 있습니다.
* 기본 게임 흐름 및 로직
  게임 흐름: 두 코드 모두 while 루프를 사용하여 게임의 흐름을 제어하며, 플레이어 수를 입력받고, 카드를 배분한

후 중복 여부를 확인하고, 최종적으로 결과를 출력하는 일련의 로직을 따릅니다.
코드의 전체적인 게임 흐름은 비슷하며, 카드 덱 초기화 -> 플레이어 수 입력 -> 카드 분배 -> 중복 확인 및 출력 -> 반복 or 종료의 순서로 진행됩니다.

4.2 차이점

* 중복 숫자 확인
- code_**ChatGPT**_2021039057 이규하.py: 중복 확인을 위해 card_numbers라는 리스트를 사용하여 각 플레이어의 카드 숫자만을 추출하고, count() 함수를 사용해 중복 여부를 판단합니다. 중복이 발생하면 바로 pair라는 문구를 출력하여, 중복 여부를 더 직관적이고 간단하게 처리할 수 있도록 구현했습니다.
- code_**myself**_2021039057 이규하.py: 중복 확인을 위해 selectedNum 리스트에 숫자만 저장하고, set()을 사용하여 중복을 확인합니다. len(selectedNum) != len(set(selectedNum)) 조건을 통해 중복 여부를 판별합니다.

* 출력 형식
- code_**ChatGPT**_2021039057 이규하.py: 출력 형식에서 각 플레이어의 카드를 , 로 구분하여 문자열로 변환한 뒤, 중복 여부에 따라 pair를 출력하는 구조입니다. 이를 통해 각 플레이어의 카드를 보기 쉽게 정리할 수 있습니다. hand_str = ", ".join(hand)를 사용하여 카드를 쉼표로 구분된 하나의 문자열로 만들었습니다.
- code_**myself**_2021039057 이규하.py: print()문 안에서 중복 여부 확인과 카드를 동시에 출력합니다. 마지막 카드인지 여부를 if s is outputCard[-1]:로 확인하여 마지막 카드일 때만 출력 형식을 다르게 처리하고 있습니다.

* 게임 종료 및 반복 방식
- code_**ChatGPT**_2021039057 이규하.py: Enter 키를 누르면 다음 판이 진행되고, z를 입력하면 게임이 종료됩니다. input("\nPress Enter to continue, or 'z' to quit: ").lower()로 명확하게 입력을 구분하였습니다. while True 루프 내에서 게임 종료와 반복 조건을 명확하게 구분하였고, 다음 판 진행 시에는 카드를 다시 섞어서 배분합니다.
- code_**myself**_2021039057 이규하.py: nextValue = input()으로 입력을 받아, Enter 입력 시 다음 판으로 진행하고, z를 입력하면 게임이 종료됩니다.

* 유효성 검사 및 예외 처리
- code_**ChatGPT**_2021039057 이규하.py: 플레이어 수 입력에서 try-except 블록을 사용하여 잘못된 입력이 들어왔을 때 예외를 처리하고 재입력을 요청합니다.
- code_**myself**_2021039057 이규하.py: 플레이어 수를 입력받을 때 try-except 블록이 없어 잘못된 입력이 들어올 때 프로그램이 비정상적으로 종료될 수 있습니다. 범위(1~5)를 벗어난 플레이어 수 입력에 대한 유효성 검사가 존재하지 않아 프로그램의 예외 처리 부분이 부족하다고 생각합니다.

5. 결론

본 과제에서는 두 코드의 공통점과 차이점을 분석하고, 각 코드의 기능과 구조를 비교하여 살펴보았습니다. 두 코드 모두 기본적인 게임 흐름과 카드 분배, 중복 확인 기능을 잘 구현하였으며, 과제의 요구사항을 만족하고 있다 생각합니다. 그러나 코드의 효율성과 가독성, 입력 처리 방식 등에서 세부적인 차이가 존재하여, 각 코드의 장단점이 명확하게 드러났습니다. 저의 코드와 GPT가 작성해 준 코드의 비교를 바탕으로, 저의 코드를 더욱 간결하고 효율적으로 개선할 수 있는 가능성을 확인할 수 있었으며, 코드를 작성할 때 효율적인 알고리즘 선택과 예외 처리에 대해 더 생각해 볼 수 있었습니다.