**BLACKJACK 과제**

컴퓨터공학과 3학년

12171645 심규범

어느 방법이 맞는 것인지 몰라 두 가지의 방법으로 Blackjack 과제1, 2, 3을 진행하였습니다.

각각의 방식을 다른 .ipynb 파일로 나누어 이름을 ‘Blackjack source code\_1’과 ‘Blackjack source code\_2’로 나누었습니다.

원래의 코드와 달라진 부분들만 보고서로 정리하여 제출 드립니다.

**Blackjack source code\_1**

1. MonteCarlo class에 초기값을 설정하는 \_\_init\_\_ method와 train을 변형한 game method를 추가하였습니다.
2. Method: \_\_init\_\_(self)

플레이어의 초기 자금 숫자열(player\_money) 10000달러와, 초기 승률 (win\_rate) 0%를 설정합니다.

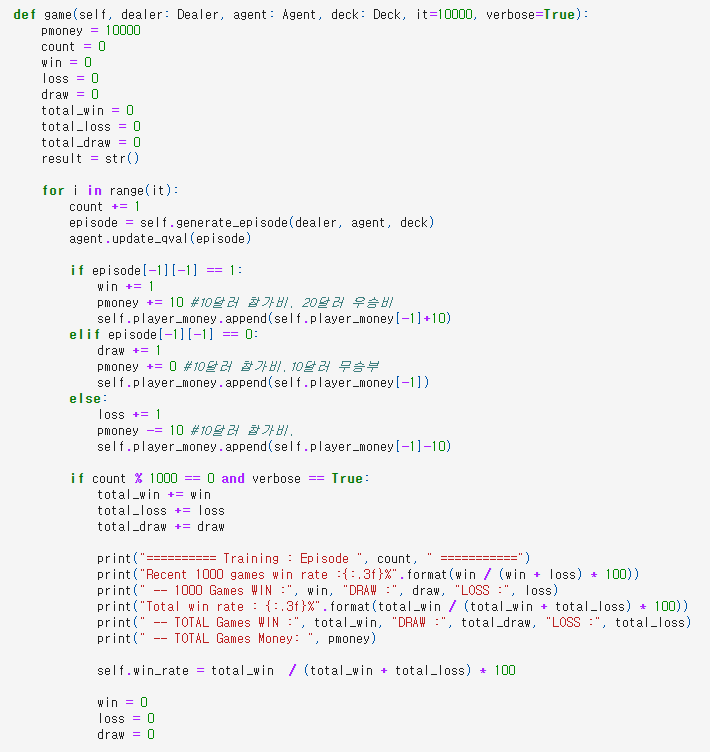


1. Method: game(dealer, agent, deck, it)

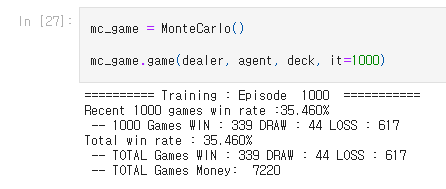
Train method에 코드를 추가하여 만든 game method입니다.

print하기 위한 플레이어 초기자금 pmoney 10000달러를 추가했습니다.

* 플레이어가 승리할 시 pmoney에 우승비 20달러 – 참가비 10달러인 10달러를 추가합니다. player\_money 숫자열에는 이전 값에 10을 추가한 숫자를 append합니다.
* 플레이어가 무승부 일 경우 무승부비 10달러 – 참가비 10달러인 0달러를 받기에 pmoney는 그대로이며, player\_money에는 이전 값과 같은 값이 append 됩니다.
* 플레이어가 승리할 시 pmoney에 참가비 10달러를 뺍니다. player\_money 숫자열에는 이전 값보다 10 작은 숫자를 append합니다.



1. 과제 1: 1000번의 게임 진행

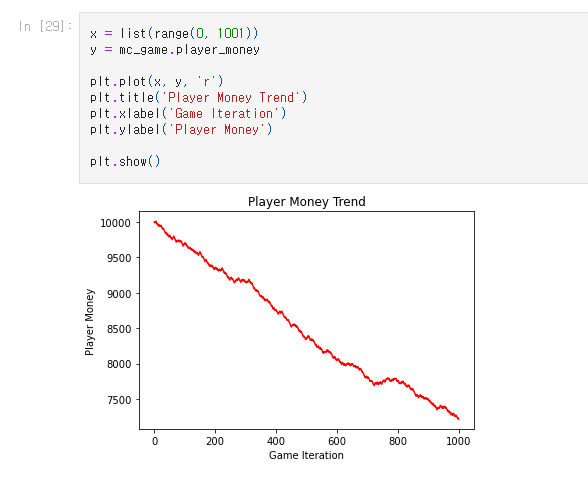


1000000번의 episode로 학습된 agent의 policy로 Iteration을 1000으로 하여 1000번의 게임을 진행하였습니다.

1. 과제 2: 플레이어의 승률을 계산



1. 과제 3: 플레이어의 소지금 변화를 그래프로 시각화



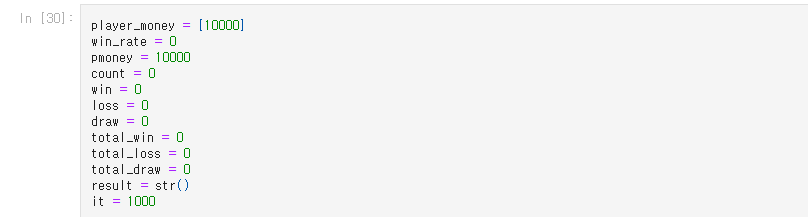
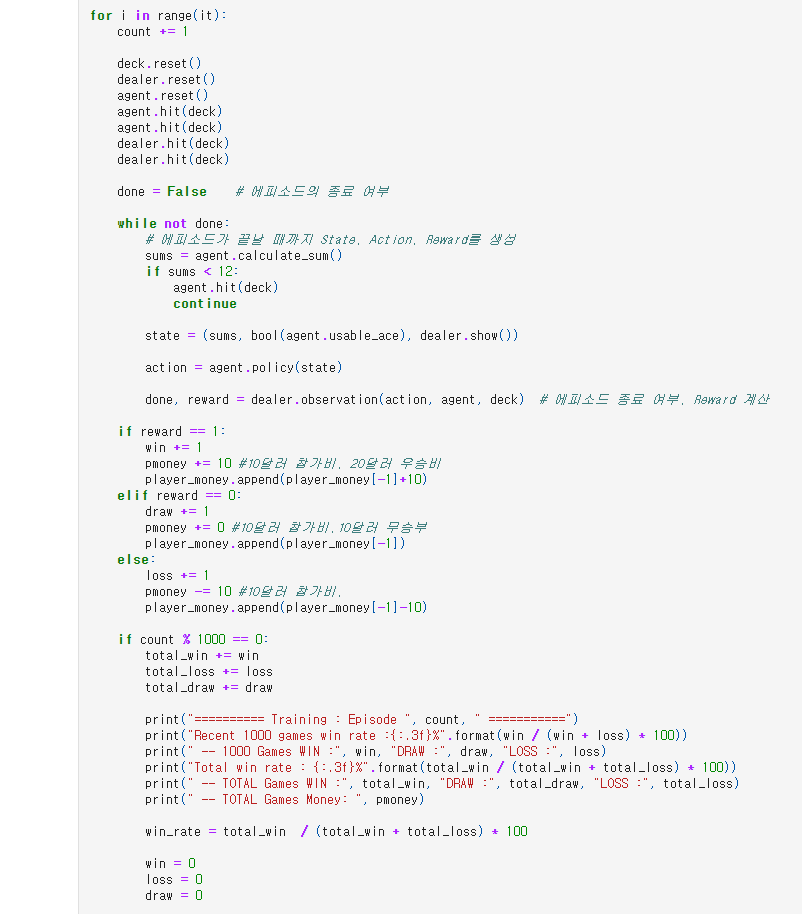
X축: 1번째에서~1000번째 list로 range를 설정합니다

Y축: player\_money의 값으로 설정합니다

그래프가 비교적 일정하게 우하향하는 모습을 볼 수 있으며, 플레이어가 전반적으로 승리하는 경우는 적습니다. 전체적으로 보았을 때 플레이어가 계속해서 패배하는 양상입니다.

**Blackjack source code\_2**

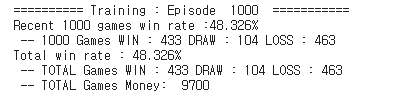
1. 과제 1: 1000번의 게임 진행

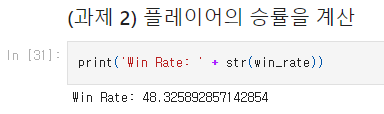
MonteCarlos class의 generate\_episode method와 train method를 섞어서 episode대신 reward로 state를 policy에 입력하는 코드를 만들었습니다. 이미 100만번의 episode로 학습된 policy가 있기에 학습용인 episode가 필요가 없다고 판단하여 episode를 reward로 대체를 하였습니다.

1. Blackjack source code\_1과 마찬가지로 플레이어의 자금을 print해줄 변수 pmoney를 10000으로, 승률 win\_rate를 0%로 초기화합니다.
2. 나머지 train method에서 쓰인 변수들을 선언 및 초기화합니다.
3. Iteration 1000동안 이루어질 게임을 만듭니다. For I in range(it), it = 1000
4. generate\_episode에서 쓰인 것처럼 deck, dealer, agent를 초기화하고 agent와 dealer을 각각 2번 hit 합니다.
5. done=FALSE를 통해 게임(episode)가 종료되었는지를 확인함. done=TRUE 일 경우 게임(episode)이 종료됩니다.
6. done=FALSE일 동안 while문 안에서 state, action, reward를 생성하고 결정합니다.
7. Reward가 계산이 되면 reward 값으로 win, draw, lose를 결정하고 그에 따라 print용 플레이어 자금 pmoney와 플레이어 자금 숫자열인 player\_money를 계산합니다.

* Win: reward == 1. pmoney에 20달러 우승비 - 10달러 참여비인 10달러를 추가합니다. player\_money는 이전 값보다 10달러가 추가된 값이 append 됩니다.
* Draw: reward == 0. pmoney에 10달러 무승부비 - 10달러 참여비로 그대로 유지됩니다. player\_money에는 이전 값과 동일한 값이 append 됩니다.
* Lose: reward == -1. pmoney에 10달러 참여비가 빠져나가 이전 값에서 10달러가 줄어듭니다. player\_money에는 이전 값보다 10이 작은 값이 append 됩니다.

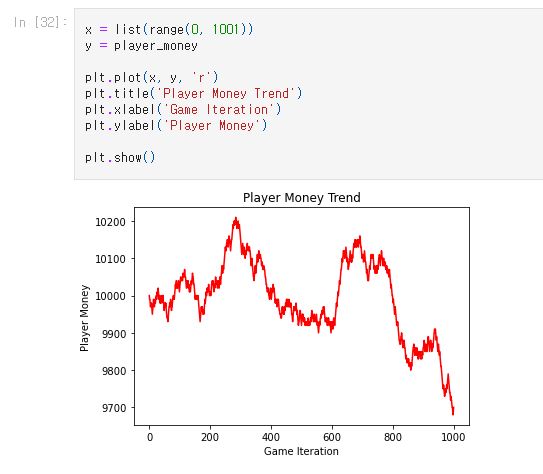


1. 과제 2: 플레이어의 승률 계산



win\_rate를 출력합니다. 승률 약 48.33%

1. 과제 3: 플레이어의 소지금 변화를 그래프로 시각화



X축: 1~1000번째 list로 range를 설정합니다.

Y축: player\_money 값으로 설정합니다.

그래프가 Blackjack source code\_1의 방식에 비해 역동적이며, 플레이어가 연속적으로 승리하는 양상도 여러 번 보입니다. 그리고 1000번의 게임이 끝났을 때 플레이어의 자금이 훨씬 많습니다.

정리: source code\_1의 방식보다 source code\_2의 방식이 승률이 48.33%로 약 13%더 높습니다. Source code\_1의 방식은 플레이어가 전반적으로 패배하는 양상이나 source code\_2의 방식은 player가 연속적인 승리와 짧은 연속적인 패배로 이루어져 있으나 후반에는 대체로 패배하는 양상입니다. 하지만 그럼에도 불구하고 승리하는 경우가 많았기에 플레이어의 자금은 약 9700달러로 첫 번째 방식인 7220보다 2480달러가 더 높습니다.