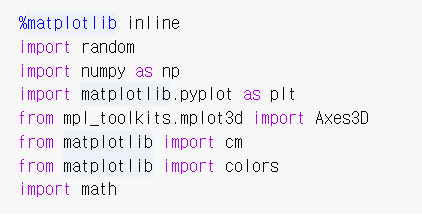
**REPORT**

12171708

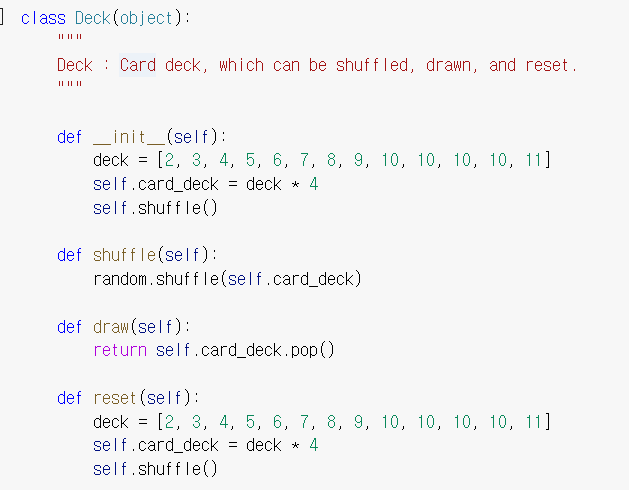
조승효

**1. 코드 설명**

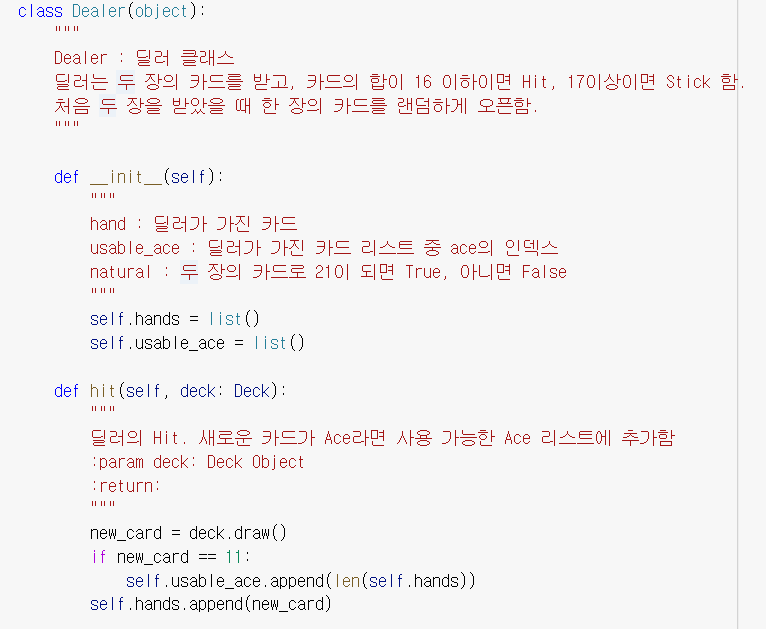
1) 기존 코드



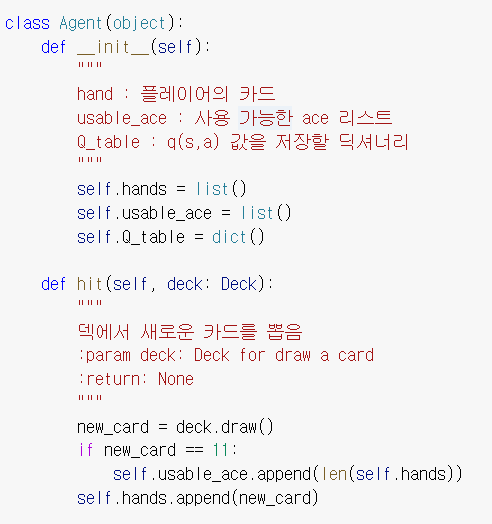
라이브러리들을 import 합니다.



Deck class를 만듭니다. 이것은 현실 세계에서 카지노의 deck과 같은 역할을 합니다.



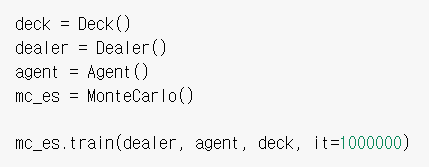
Dealer Class를 만듭니다. Dealer의 역할을 수행하게 됩니다.



Agent class를 만듭니다. Agent 역할을 수행합니다.

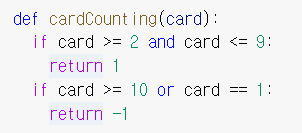


MonteCarlo class를 만듭니다. 이 클래스는 하나의 에피소드(게임)을 생성합니다. 또한 생성한 에피소드를 이용해 training하는 데에도 쓰입니다.

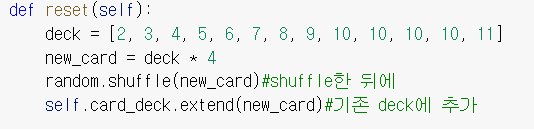


인스턴스들을 이용해 트레이닝 하는 부분입니다. 1,000,000번 게임을 진행해 거기에서 나온 결과를 토대로 training 하는 것을 알 수 있습니다.

2) 카드 카운팅 을 포함한 코드

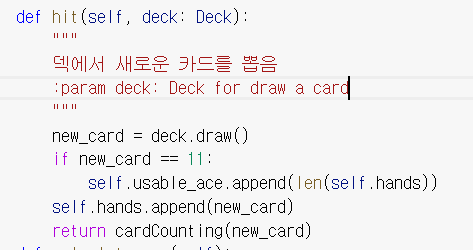


카드 카운팅 값을 돌려주는 함수를 만듭니다. 플레이어의 카드 카운팅 값이 높아질수록 플레이어에게 유리하다고 가정하였습니다. 이때 카드가 2~9가 나온다는 것은 10, A가 나올 확률이 커진다는 것이며, dealer가 burst할 확률이 높아진다는 것을 의미합니다. 즉 플레이어에게 유리해짐을 의미하여 그래서 값을 높여가게 1을 return하게 하였습니다. 카드가 10,A가 나온다는 것은 10,A가 나올 확률이 작아진다는 것이며, dealer가 burst할 확률이 낮아진다는 것을 의미합니다. 즉 플레이어에게 불리해짐을 의미하여 그래서 값을 낮추게 -1을 return하게 하였습니다.



기존의 deck에서 reset부분을 수정하였습니다.

기존에는 게임이 끝나면 기존의 deck을 다 지우고 새로운 카드를 52장 채웠습니다. 하지만 카드 카운팅에서는 카드가 일정 수 이하일 때에만 shuffle된 52장의 카드를 deck에 추가하도록 하였습니다.



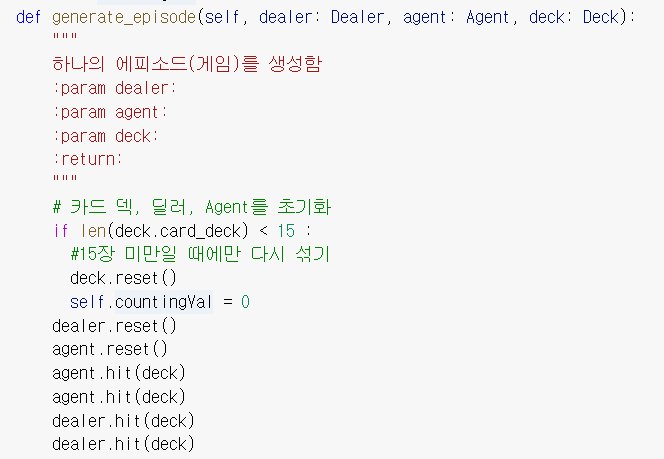
Agent의 hit을 수정하였습니다. 이 때 agent가 카드를 뽑았을 때 카드 카운팅 값을 리턴하게 하였습니다.



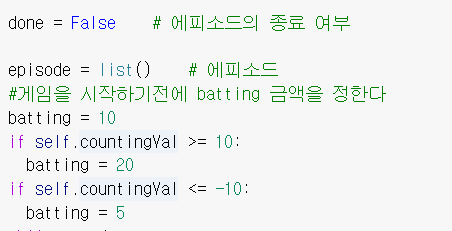
Dealer의 observation을 수정하였습니다. 이 함수는 dealer가 agent의 행동을 감시하는 부분입니다. 이때 counting도 함께 리턴함을 알 수 있습니다. MonteCalro class의 generate\_episode를 보면 agent가 직접 hit을 하는 부분이 없음을 확인 할 수 있습니다. 대신에 dealer가 agent의 action을 받아서 대신 agent의 hit을 진행하는 부분이 있습니다. 한편 MonteCalro에는 agent가 반환하는 카드 카운팅 값을 필요로 합니다. 그러나 dealer가 대신해서 hit을 처리해주니 결과적으로 MonteCalro는 dealer에게 agent가 반환한 값을 받아야 합니다. 따라서 counting을 return하게 됩니다.



MonteCalro 클래스에서 카드 카운팅 값을 저장하는 부분입니다.



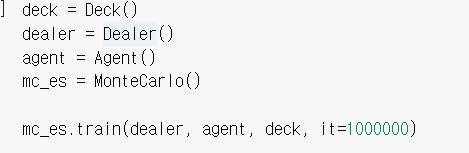
Generate\_episode를 수정한 부분입니다. 이때 deck reset이 카드 수가 15장 미만일 때에만 수행됨을 확인할 수 있습니다.



게임을 시작하기 전에 batting값을 설정합니다. 임의로 정하였습니다. countingVal이 10이상이면 플레이어에게 유리함을 의미합니다. 그래서 batting값을 2배로 설정하였습니다. countingVal이 -10이하이면 플레이어에게 불리함을 의미합니다. 그래서 batting을 절반으로 설정하였습니다.



State에 countingState가 추가되었음을 확인할 수 있습니다. 이는 플레이어의 유불리를 나타냅니다. countingVal이 게임이 진행됨에 따라 추가됨을 확인할 수 있습니다. 그리고 마지막에 batting도 리턴하는 부분도 있음을 확인할 수 있습니다.



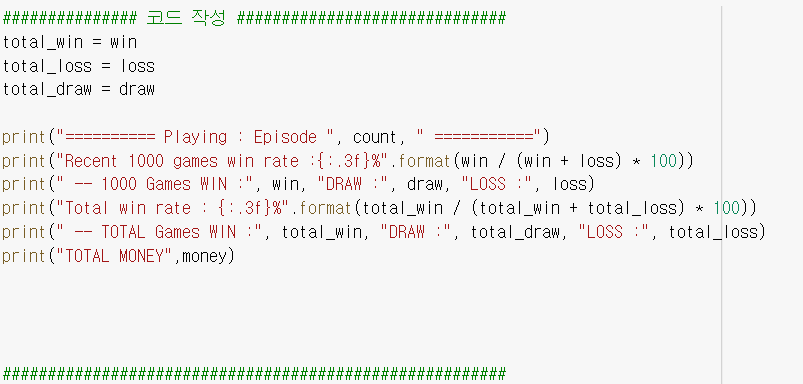
Training 하는 부분입니다.

**2. 실험 결과 및 고찰**

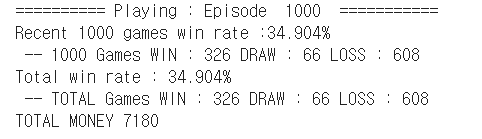
1)기존 코드

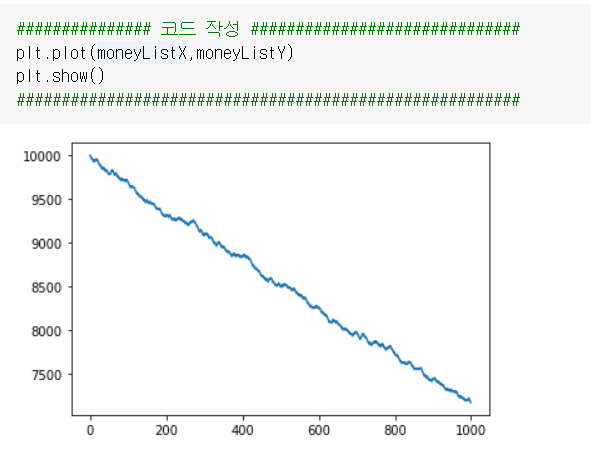


에피소드를 1000개 만들어 이기고 짐에 따라 이긴 횟수, 비긴 횟수, 진 횟수, 딴 돈을 기록하고 있습니다.



결과를 출력하는 부분입니다. 출력된 결과는 아래와 같습니다.



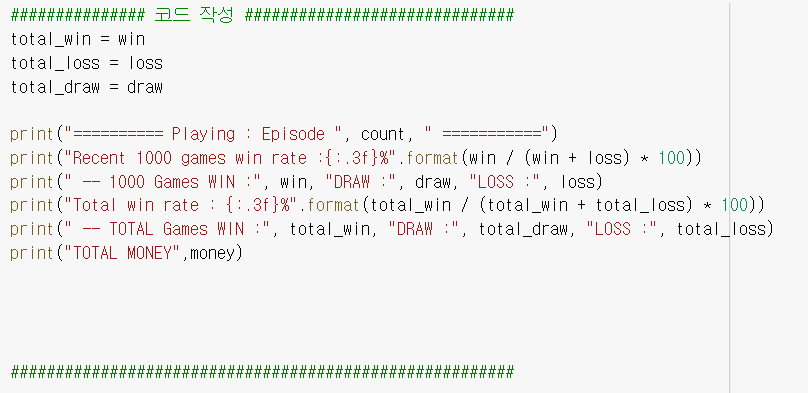


소지금의 변화율을 출력하는 부분입니다.

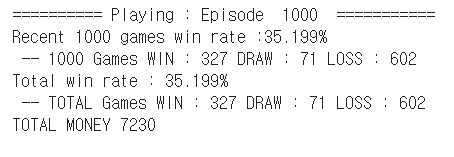
2)카드 카운팅

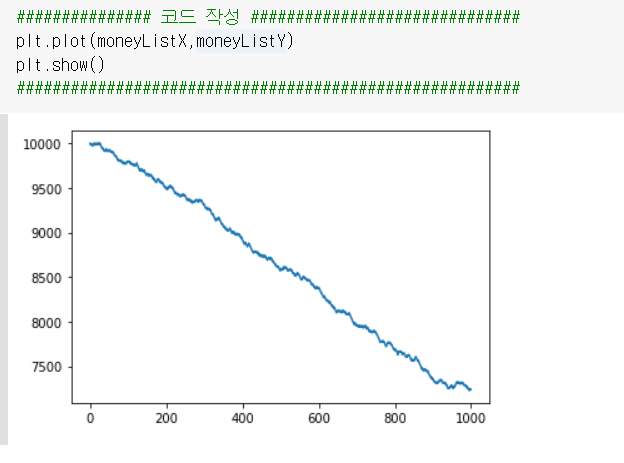


Generate\_episode에서 batting을 리턴한 부분이 있었는데 바로 이 부분에서 쓰입니다. 게임의 규칙은 이기면 배팅한 금액의 두 배 비기면 배팅한 금액만, 지면 못 돌려받는 것이었습니다. 근데 episode마다 배팅한 금액이 다를 수 있습니다. 따라서 episode에서 배팅한 금액을 알아야 하는데 이러한 것을 위해 generate\_episode에서 batting을 return한 것이었습니다.



결과를 출력하는 부분입니다. 결과는 아래와 같습니다.





소지금 변화를 출력한 부분입니다.

이와 같은 결과를 보았을 때 카드 카운팅을 하는 것이 근소하게 앞섬을 확인할 수 있습니다. 그러나 그다지 유의미한 차이가 있는 것은 아니었습니다. 이유는 Agent가 Hit, Stick만 할 수 있었기 때문입니다. 만약에 Agent가 Hit, Stick외에도 Stop, Doubling을 할 수 있다면 유의미한 차이가 만들어 졌을지도 모릅니다. 사실 blackjack이라는 것은 카드의 합이 어느정도 되냐, a카드가 있느냐에 따라 승률이 달라지는 게임입니다. 그래서 카드가 나온 현재 상황에 따라 hit을 해야 할지 stick을 해야할지 정해집니다. 이 규칙들은 생각 외로 복잡합니다. 하지만 blackjack을 강화학습을 이용해서 계산하면 state, reward만으로 최적의 policy가 나옴을 확인할 수 있습니다.