**OSSP 최종 보고서**

**<League of Legends 게임**

**챔피언 추천 프로그램>**

DHL

2016112190 한종호

2016112204 임정우

2016112231 정현성

2016112194 최재원

1. **프로젝트 목표**
2. <League of Legends> 게임에 익숙하지 않은 플레이어를 위해 챔피언 선택 과정에서 상대 챔피언, 아군 챔피언들을 참조하여 상황에 알맞은 챔피언을 추천해주는 프로그램을 제작한다.
3. RIOT API 로부터 받은 실제 게임 데이터들을 가지고 챔피언별 승률을 계산하고, 이를 아군의 조합별 승률, 상대 챔피언 상대 시 승률을 도출해낸다.
4. **프로젝트 개요**
5. <League of Legends>(이하 롤) 게임은 RIOT게임즈에서 출시한 현재 한국에서 가장 인기가 많은 게임이다. 출시된 지 10년이 넘었지만, 국내는 물론 세계적으로도 가장 유명하고, 세계 대회가 매년 여러 번 개최될 만큼 인기 많은 게임이다. 20대 남성이라면 한번쯤 게임을 해본 경험이 있을 것이다. 실제로 국내에서도 소규모로 대학간 대회나 피씨방간 대회가 계속 열리고 있다. 이 게임의 인기가 식지 않는 이유는 지속적인 업데이트와 매년 이루어지는 대규모 업데이트들이 아닐까 싶다. 시즌제로 진행되는 게임으로 1년마다 시즌이 바뀌고, 그 때마다 규모가 큰 업데이트를 함으로써 게임 플레이어들에게 항상 새로움을 느낄 수 있게 해준다. 또한, [챔피언]이 지속적으로 출시되는 점도 인기에 큰 영향을 미친다. “챔피언”이란, 실제로 플레이어가 한 게임에 선택해서 직접 플레이하게 되는 플레이어의 아바타 역할을 하는 것이라고 보면 된다. 롤은 한 게임당 10명의 플레이어가 진행하게 되며 (즉, 한 게임당 10개의 챔피언이 선택된다.) 5명씩 한 팀을 이루고 블루팀, 퍼플팀으로 나뉘어서 게임을 플레이한다. 한 게임당 짧게는 15분, 길게는 1시간 가까이 자신이 선택한 챔피언을 가지고 플레이한다.
6. 왼쪽에 있는 그림은 실제 게임 속 맵인데, 5명이 한팀으로 게임하는 게임이다 보니 각자 정해진 역할을 가지고 플레이한다. 맵은 크게 탑, 정글, 미드, 바텀으로 이루어져 있으며, 대각선으로 보이는 파란색 강을 기준으로 왼쪽 아래는 블루팀, 위쪽은 퍼플팀의 진영이다. 플레이어들은 역할을 ‘라인’이라고 부르고, 바텀라인은 특별히 2명이 가서 플레이 하게 된다. 이 ‘라인’은 게임이 매칭되는 순간 정해지는데, 한 팀 내에서 서로가 원하지 않는 라인이 매칭되었을 때에는 정해진 라인을 서로 바꾸어 가기도 한다.
7. 롤에는 여러 게임 모드가 있지만, 가장 많이 플레이 되는 것은 실제로 전체 플레이어 중 유저의 등수를 매겨주는 [랭크 게임]이다. 현재 랭크를 알려주는 ‘티어’에는 높은 것부터 순서대로 챌린저, 그랜드 마스터, 마스터, 다이아, 플래티넘, 골드, 실버, 브론즈, 아이언 이 있고, 대회에 참가하는 프로게이머들은 대부분 챌린저~그랜드 마스터 구간에 분포되어 있다. 따라서 티어는 현재 실력이 비슷한 유저 10명이 한 게임에 참여할 수 있게 해주는 지표이다.
8. 랭크 게임은 일반 게임과 다르게 챔피언을 금지하는 과정(밴)이 있고, 챔피언을 고르는(픽) 순서가 정해져 있다. 밴의 경우 10명이 각자 1개의 챔피언을 밴할 수 있고, 밴을 당한 챔피언은 해당 게임에서 선택될 수 없다. 이후 블루팀 1> 퍼플팀 2> 블루팀 2> 퍼플팀 2> 블루팀 2> 퍼플팀 1 개씩 챔피언을 선택하는 과정을 거친 후 게임이 시작된다. 챔피언을 선택하는 순서가 굉장히 중요한데, 이는 밑에서 설명한다.
9. 현재 롤에는 152개의 챔피언이 있고, 각 챔피언마다 어울리는 라인이 정해져 있다. (물론, 정해진 역할과 다르게 특이한 라인으로 시도해보는 플레이어도 있지만 대부분 이 시도는 실패한다.) 때문에, 각 라인에서 상대로 만나는 챔피언들도 대부분 정해져 있고, 상대로 만나는 챔피언간 상성이 굉장히 중요하다. 만약 나와 같은 라인에 서게 될 상대방이 챔피언을 먼저 선택할 경우, 해당 챔피언에 유리한 챔피언을 선택하면 승률이 올라가기 때문에 챔피언을 선택하는 순서가 매우 중요하다.
10. 이를 확실히 보여주는 경우가 한 챔피언만 플레이하는 유저일 경우이다. 한 챔피언만 플레이하는 유저 중 높은 티어에 있는 유저들을 이른바 해당 챔피언의 ‘장인’이라고 부르는데, 이들의 밴픽 과정을 보면 모든 경기마다 특정 챔피언만 밴하는 것을 볼 수 있다. 이는 한 챔피언의 숙련도가 매우 높은 유저도 같은 라인에 상대로 도저히 이길 수 없는 챔피언이 존재한다는 것을 뜻한다. 즉, 챔피언간 상성이 승률에 가장 큰 요인으로 작용한다는 것을 알 수 있다.
11. 우리가 만들고자 한 프로그램은 위에 설명한 것처럼 중요한 챔피언 과정에서 챔피언 상성에 대해서 잘 모르는 유저를 위해 본인의 순서 이전에 나온 챔피언들을 참조하여 가장 승률이 높은 챔피언을 추천해주는 프로그램이다.
12. **프로젝트 구성**
13. API로부터 받은 데이터
14. 매치 데이터

게임에서 실제로 진행된 매치들에 관한 정보.

* + 1. 한 유저에 대한 매치 데이터 (유저별 매치 데이터)

해당 유저가 플레이한 챔피언, 게임 ID, 라인 등 정보가 있다.

* + 1. 전체 매치 데이터 (10명의 유저 포함)

매치 정보(게임 ID, 진행된 시간, 게임 버전, 참가한 유저 등), 게임에 참여한 유저 10명과 유저들이 플레이한 정보(유저의 소환사 ID, 한 유저가 상대방에게 가한 피해량, 상대로부터 받은 피해량 등), 팀 단위로 플레이한 정보(승패 여부, 한 팀이 특정 오브젝트를 처치한 횟수 등) 등이 있다.

# 소환사란 게임 내에서 유저를 지칭하는 표현이다.

1. 유저 데이터

챌린저 ~ 마스터 티어들에 존재하는 유저에 대한 정보. (실력이 높은 유저들의 매치들만 받아오는 이유는 티어가 낮은 유저들의 매치에서는 실수들이 나오는 경우가 많고 이 때문에 승패가 뒤집히는 경우가 많기 때문이다.)

티어, 계정 ID, 소환사 닉네임, 소환사 ID, 가장 많이 플레이한 챔피언 정보가 있다. (소환사 닉네임, ID는 게임 내에서 변경 가능)

1. 챔피언 데이터

RIOT게임즈에서 제공하는 챔피언 정보 (이름과 key 값을 가짐)

key값을 사용하는 이유는 다른 모든 데이터에서 챔피언 정보가 key값으로 저장되어 있기 때문이다. 각 챔피언마다 가장 많이 가는 라인을 데이터프레임에 입력하여 이후 라인별 상대 챔피언에 대한 승률을 구할 때 정해진 라인에 한해서만 계산하도록 한다.

1. 전처리한 데이터
2. 위와 같이 API로부터 받은 데이터중 필요한 데이터를 분류한다.

* 매치 데이터 중 게임 ID, 플레이한 챔피언, 승패 여부
* 유저 데이터 중 소환사 ID
* 챔피언 데이터

1. 매치 데이터 중 참가자에는 유저 데이터에서의 소환사 ID가 참조된다. 즉, 유저 데이터에서 얻어낸 높은 티어의 유저들의 소환사 ID를 통해 매치 데이터를 얻어낼 수 있다.
2. 유저 별로 얻어낸 대량의 매치 데이터에서 게임 ID를 사용해 전체 매치 데이터를 불러온다. 이는 유저별 매치 데이터에는 같은 라인에 오는 상대 챔피언을 알 수 없기 때문이다. 전체 매치 데이터에는 10명에 해당하는 챔피언이 모두 저장되어 있기 때문에, 상대나 아군 조합을 따져 볼 수 있다.
3. 위에서 불러온 전체 매치 데이터에서 각 챔피언별 승률을 구한다. 챔피언 데이터에 저장된 챔피언별 라인을 통해 각 라인에 맞는 챔피언들끼리의 승률을 계산한다. 또한, 아군 조합을 참조하기 위해 같은 팀에 특정 챔피언이 있을 경우의 승률도 계산한다.
4. 추천 과정
5. 전처리된 데이터들을 통해 라인마다 챔피언별 상대 승률과 아군 특정 챔피언과의 조합 승률을 사용한다. 상대 승률의 경우, 라인별로 한 챔피언에 대해 같은 라인에 올 수 있는 모든 상대별 승률을 계산해서 pickle 파일로 저장해 주었다.

하지만, 매치 데이터에서 많이 나오지 않는 챔피언의 경우는 승률을 계산했을 때 문제가 발생한다. 만약 ‘a’ 챔피언과 ‘b’ 챔피언이 같은 매치에 나온 횟수가 1번이라면, 이 승률은 그렇게 믿을 만한 근거가 되지 않는다. 따라서 이런 경우를 대비해 평균 경기수보다 적은 경우의 상대 승률은 한 쪽으로 치우치지 않도록 0.5 로 고정시켜 주었다.

1. 아군 조합의 경우 본인을 제외한 4명의 챔피언에 대해서 승률을 계산하는 것이 최선이라고 생각했다. (챔피언 선택에는 순서가 있기 때문에 최대 4명까지 참조 가능) 이 과정에서 생각한 방법들은 아래에 있다.
2. 가장 먼저 생각한 방식은 베이즈 정리를 사용한 방식이다. 베이즈 정리란, 두 가지의 확률 변수의 사전 확률과 사후 확률 사이의 관계를 나타낸다.

베이즈 정리 공식이다. P(H|E)는 사후 확률, P(H)는 사전 확률이다. 이 공식을 우리 프로그램의 추천 방식에 적용하면, P(H|E)는 해당 챔피언을 선택했을 시 예측되는 승률이며, P(E|H)는 내 순서 이전에 선택된 챔피언들과 내가 고를 챔피언이 포함되어 있는 경기에서의 승률이다. 즉, 내 순서가 4번째라면, 이전에 선택되어 있는 3개의 챔피언과 추천해줄 챔피언까지 총 4개의 챔피언이 포함되어 있는 매치를 모두 분석하여 승률을 계산해주는 것이다. P(H)는 추천해줄 챔피언의 승률이다. P(E)는 추천해줄 챔피언이 포함되었을 때의 승률과 다른 챔피언이 포함되었을 경우의 확률을 더해주는 것이다.

즉, 내가 3번째 순서이면, 앞서 선택된 챔피언은 아군 1개, 적팀 1개이고, P(E)는 총 152개의 챔피언 중 선택된 2개의 챔피언을 제외한 150개의 챔피언에서 (‘a’라는 챔피언이 포함된 매치들의 승률) + (‘a’를 제외한 149개의 챔피언이 포함된 매치들의 승률의 평균)으로 구해진다.

베이즈 정리를 사용한 이 방식의 장점은, 이미 선택되어 있는 모든 챔피언들과 같이 계산이 되기 때문에, 보다 정확한 승률이 예측된다는 것이다. 우리가 수집한 데이터에서 특정 경우를 찾아내서 계산하는 것이기 때문에 한 가지 조합에 한해서 더 확실한 결과를 내준다.

하지만, 이 방식에는 너무 큰 단점이 있었다. 데이터 수의 부족이다. 챔피언의 수가 152개나 되기 때문에 특정 조합이 매치에서 나올 확률이 매우 적기 때문이다. 롤이라는 게임의 특성상 현재 패치 버전에 특히 좋은 챔피언들이 정해져 있기 때문에 그러한 챔피언들의 등장확률은 높은 편이지만, 좋지 않다고 생각되는 챔피언의 경우 전체 매치 데이터에서 찾기 힘든 경우가 있다. 때문에, 한 팀의 5명중 먼저 선택하는 1~2명 까지만 정확한 승률을 예측해준다. 선택된 챔피언이 많아질수록 해당 챔피언들이 모두 포함된 (팀까지 맞춰서) 경우가 희박해지기 때문에 실용성이 떨어진다고 판단하였다.

이 방식의 또다른 문제점은 유저의 상황에 맞지 않는 챔피언도 고려한다는 점이다. 각 상황에서 현재 나오지 않은 모든 챔피언의 승률을 예측하여 추천해주는 방식이기 때문에, 유저가 한 매치에서 정해진 라인에 어울리지 않는 챔피언을 추천해주는 경우가 발생한다. 가령 유저가 탑 라인에 매칭이 되었으나 추천되는 챔피언이 정글 라인에 맞는 챔피언일 경우 이 추천은 필요가 없어지게 된다.

1. 위처럼 베이즈 정리를 사용한 방식이 명확한 한계점을 가지기 때문에 차선책으로 생각한 방법이 각 라인별 챔피언을 독립적으로 분석하는 것이다. 앞에서 계산한 같은 라인에서 상대 챔피언별 승률과, 같은 팀에서 다른 라인의 챔피언이 존재할 때의 승률을 계산하여 추천 챔피언을 계산해내는 방식이다.

이 방식에서 가장 문제가 되었던 점은 아군의 조합을 고려할 상황이다. 만약 내 차례가 우리 팀에서 마지막이라면, 앞서 선택된 아군의 4개의 챔피언이 존재할 때의 승률을 계산해야 한다. 하지만 이는 베이즈 정리를 사용한 방식에서 나타난 한계처럼 데이터가 적어지는 문제를 야기한다. 그래서 우리가 생각한 방식은 각 라인별로 가장 영향을 많이 주거나 받는 라인만을 참조하는 것이다.

즉, 탑 라인을 예로 들면, 탑 라인은 다른 라인의 플레이에 큰 영향을 주는 경우가 적다. 하지만, 정글 라인으로부터 도움을 받는 경우가 많다. 따라서, 탑 라인 챔피언의 조합을 계산할 때 필요한 정보는 아군 정글 라인의 챔피언과 함께 매치에 등장했을 때의 승률이다.

이처럼 각 라인별로 가장 크게 영향을 주는 라인은 1) 탑 – 정글, 2) 미드 – 정글, 3) 바텀, (바텀 라인은 2명이 함께 플레이하는 라인으로 이 2명이 서로에게 가장 큰 영향을 준다) 4) 정글 – 탑 , 5) 정글 – 미드 이다. 정글 라인의 경우 맵 전체를 돌아다니면서 여러 라인에 도움을 주는 역할을 하기 때문에 하나의 라인이 아닌 2개의 라인과 함께 생각하는 것이 좋다고 판단하였다.

1. 위에서 정해진 방식으로 아군 조합에서의 승률까지 계산하여 추천해줄 챔피언을 찾는데, 다음으로 생각할 점은 유저가 챔피언을 선택할 순서이다. 앞서 게임 설명에서 말한 것처럼 각 팀당 한 번에 2명까지 선택을 번갈아 가면서 하게 되고, 이 속에서 유저가 어떤 위치에 있는지가 챔피언을 추천해줄 때 계산을 다르게 해야 하는 부분이다.

만약, 유저가 양팀에서 제일 처음 선택을 하게 된다면, 상대도 아직 없고, 아군도 아직 없는 상태이기 때문에, 참조할 만한 승률이 없다. 이렇게 경우의 수를 나누어 보면, [1. 상대도 없고 아군도 없을 때, 2. 상대는 있지만 아군이 없을 때, 3. 상대는 없지만 아군은 있을 때, 4. 상대도 있고 아군도 있을 때]로 나눌 수 있다.

각 상황에 맞추면, 1. 상대와 아군이 모두 나오지 않았을 경우

이 경우는 주로 유저가 게임 내에서 가장 먼저 선택을 해야 하는 경우이다. 즉, 상대팀까지 총 10명 중 가장 먼저 챔피언 선택을 해야 하는데, 이 때는 챔피언을 추천하는 것이 의미가 없어진다. 만약 모든 챔피언을 상대하는 승률의 평균이 가장 높은 챔피언을 추천해준다면, 해당 챔피언을 상대로 승률이 매우 높은 챔피언이 존재할 수 있고, 상대방이 그 챔피언을 선택해버린다면 유저의 선택이 최악의 선택이 될 수 있다. 따라서 이 경우 추천을 해주기 보다는, 유저가 가장 자신 있어 하고 잘하는 챔피언을 선택하는 것이 올바른 방식이라고 판단했다.

다음으로 2. 상대는 선택되었지만 아군이 선택되지 않았을 경우

이 경우는 유저와 같은 라인의 상대방은 선택했지만 아군에서 유저의 라인과 조합적으로 잘 맞는 라인의 챔피언이 아직 선택되지 않은 경우이다. 이 때는 상대방의 챔피언을 상대로 가장 승률이 높은 챔피언을 추천해준다.

3. 아군은 선택되었지만 상대가 선택되지 않았을 경우

이 경우는 아군에서 유저와 조합이 잘 맞는 라인의 챔피언이 선택되었지만, 유저와 같은 라인의 상대방이 아직 선택하지 않은 경우이다. 이 때는 아군의 챔피언과의 승률을 참조하여 가장 승률이 높은 챔피언을 추천해준다.

4. 아군과 상대방 모두 선택되었을 경우

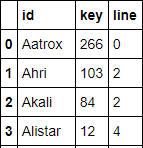
이 경우는 유저의 선택 순서가 후순위일 경우 발생하는 경우이다. 이 때는 두 가지 모두 고려해야 하기 때문에, 상대방 챔피언을 만났을 때의 승률 \* 아군 챔피언과 함께 했을 때의 승률로 계산하여 가장 승률이 높은 챔피언을 추천해준다. 두 가지 승률을 곱하는 이유는 각각의 두 가지 상황이 동시에 일어나서 승리할 확률이기 때문이다.

1. GUI 제작
2. GUI 제작의 경우 프로그램 사용자가 한눈에 알아보기 쉽게 제작하도록 노력하였다. 실제 게임 속 챔피언 선택 창과 가장 비슷한 형태로 구현하여 사용자가 따로 설명이 없어도 같은 방식으로 동작하는 것을 알 수 있도록 하였다.
3. 웹사이트 중 밴픽을 연습할 수 있도록 만들어진 페이지들이 있는데, 실제로 소규모 롤 대회를 준비하는 사람들이나, 대회에서 진행된 경기의 밴픽을 연구하기 위해 사용되고 있다. 이러한 페이지들 로부터 영감을 얻어 형태를 비슷하게 구현하였다.
4. 파이썬 모듈인 tkinter를 사용하여 인터페이스를 구현하고, 버튼을 생성하여 유저가 실제 게임내에서 밴픽을 진행하며 금지당한 챔피언과 선택된 챔피언들을 입력할 수 있다. 유저의 차례가 되었을 때 본인의 라인을 선택하고 추천받기를 누르면 앞에서 설명한 방식으로 계산된 승률이 높은 것부터 5개가 차례대로 출력되게 하였다. 유저는 추천된 챔피언 중 자신있어 하거나 원하는 챔피언을 선택하면 된다.
5. **프로젝트 결과**
   1. API로부터 받은 데이터
6. 유저 데이터



이 중 summoner ID(소환사 ID)를 통해 유저별 매치 데이터를 받아온다.

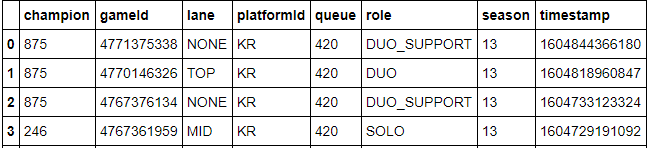
1. 챔피언 데이터



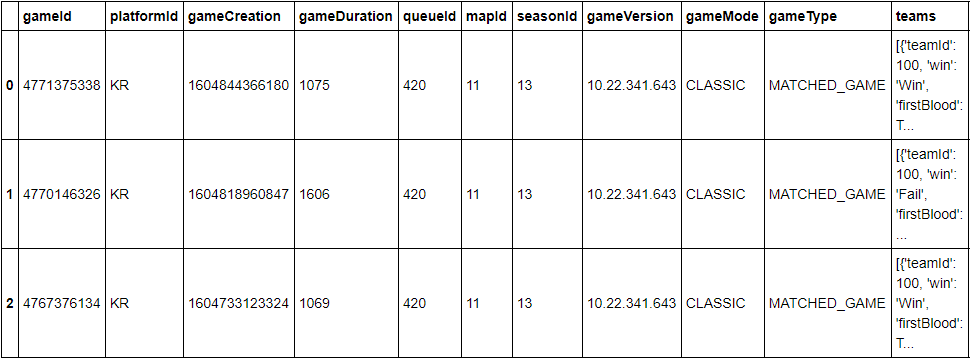
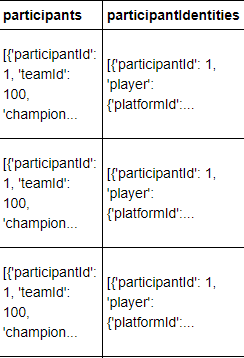
각 챔피언마다 이름과 key값이 존재한다.

line의 경우 해당 챔피언이 가장 많이 가는 라인은 입력해주었다.

1. 유저별 매치 데이터



유저 데이터에서 얻은 소환사 ID로 한 유저가 플레이한 매치들을 받아온다. 이 데이터에서 gameID를 통해 전체 정보가 있는 매치 데이터를 받아온다.

1. 전체 매치 데이터

한 매치에 대해 모든 정보들이 들어가 있다. 10명의 플레이어 각각에 대한 정보, 각각이 플레이한 챔피언 정보 등을 가지고 있고, 이 중 플레이어가 고른 챔피언들과 승패에 대한 정보들을 사용한다.

* 1. 데이터 전처리

1. 높은 티어의 유저들을 기준으로 받아온 데이터이다 보니 전체 매치 데이터에는 중복이 발생한다. 매치 1개가 여러 번 계산되면 승률 부분에서 정확한 측정이 되지 않으므로 중복을 제거해준다.
2. 전체 매치 데이터에서 특정 챔피언을 기준으로 나머지 챔피언들을 적팀으로 만났을 때의 경기수를 구하여 해당 챔피언이 등장한 모든 경기수를 나누면 승률이 나온다.

또한, 그 특정 챔피언이 같은 팀의 다른 챔피언과 함께 매치를 진행했을 경우의 경기수를 구하여 승률을 계산한다.



key = 266인 챔피언에 대해 행의 챔피언을 상대했을 시의 승률

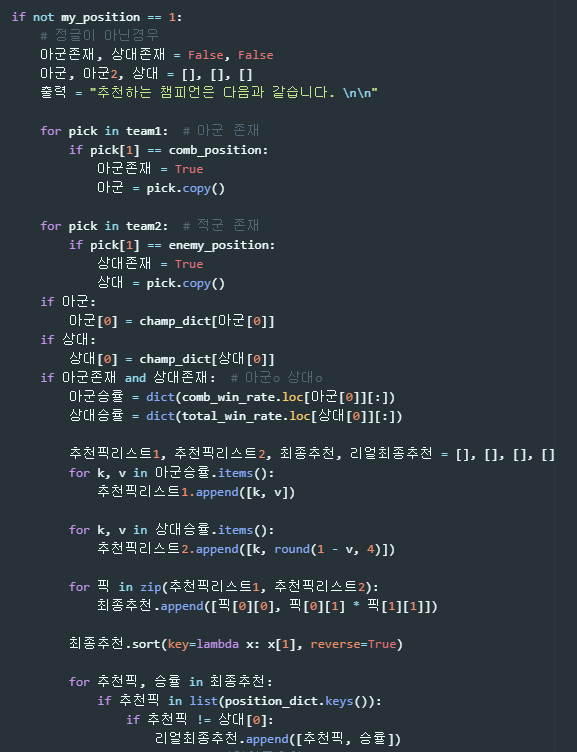
(행 번호는 특정 챔피언 key값이 아님.)

특정 챔피언에 대해 아군 조합 챔피언과의 승률도 이와 같은 형식으로 저장된다.

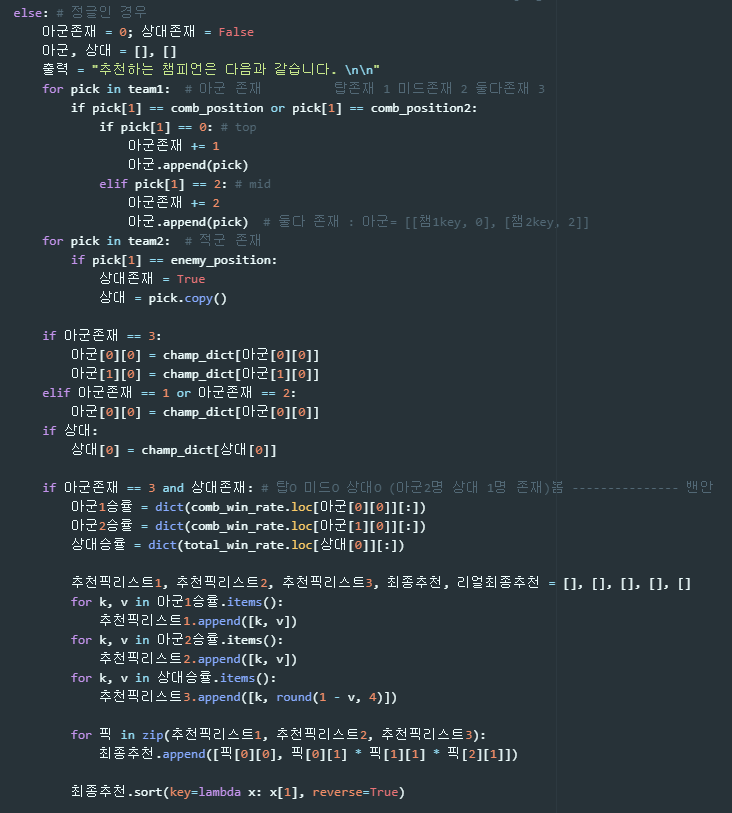
* 1. 챔피언 추천 과정



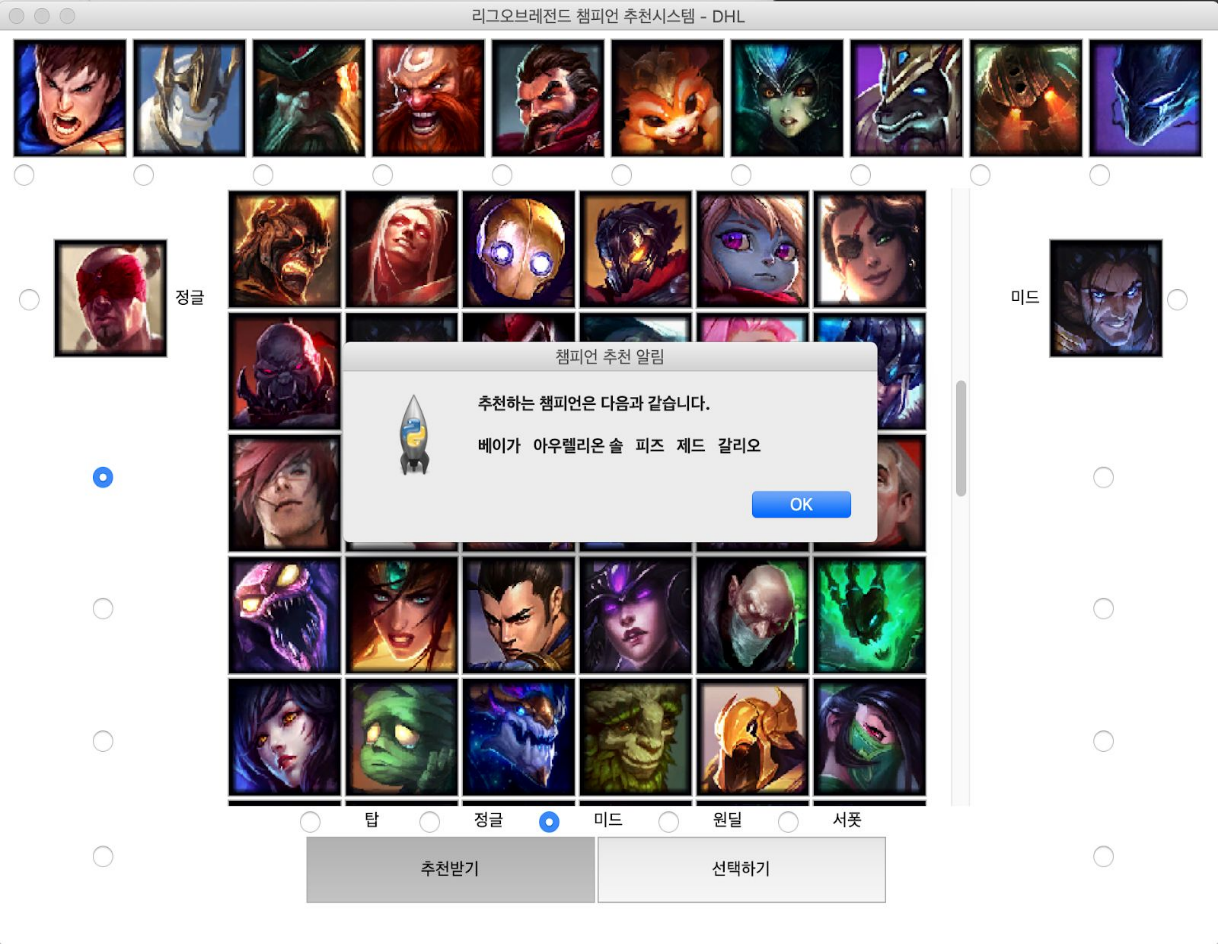
유저의 라인에 따라 comb\_position 즉, 조합이 가장 좋은 라인을 지정하여 이후에 조합 라인이 있는지 확인하여준다.



유저의 라인이 정글이 아닌 경우에는 아군의 조합으로 참조하는 라인이 각자 1개씩이다. 각 상황에 맞춰서 추천해줄 챔피언의 승률을 계산한다. 챔피언 선택에 앞서 밴된 챔피언의 경우 추천되는 챔피언 리스트에서 빠지도록 한다.



유저의 라인이 정글인 경우, 아군의 조합으로 탑 라인과 미드 라인의 챔피언이 있는지 확인한다. 이후 각 상황에 맞춰서 추천해줄 챔피언의 승률을 계산한다. 다른 라인의 경우와 마찬가지로 밴된 챔피언은 추천 챔피언 리스트에서 삭제하여 출력한다.

* 1. GUI

1. **프로젝트 고찰**
2. 챔피언 추천과정

- 우리가 계획한 대로 챔피언 추천이 가능하다. 유저의 순서에 맞춰 이전에 선택되거나 밴된 챔피언들을 모두 참조하여 남아있는 챔피언 중 유저의 라인에 맞는 챔피언의 승률을 계산하여 가장 높은 것부터 출력된다.

- 하지만, 구현한 추천 방식 자체가 모든 상황에 대해서 계산하는 것이 아닌 유저의 현재 상황과 이전 상황에 대해서만 계산하다 보니 유저의 챔피언 선택 이후의 상황에 대해 계산할 수 없다는 점이 아쉽다.

1. GUI 부분

- tkinter 모듈을 처음 사용하다 보니 인터페이스 구현에 있어 미숙한 점이 많았다. 중앙에 챔피언을 선택할 수 있는 부분에서 스크롤이 제대로 작동하지 않은 점, 챔피언 선택시 전체 창 비율이 맞지 않는 점들이 아쉽다.

- 추가적으로 가장 구현하고 싶었던 점이 챔피언 검색 기능이다. 실제 게임과 동시에 프로그램이 작동하여야 하기 때문에 챔피언을 위치에 집어넣어주는 것이 빠르게 이루어져야 하는데, 이 과정에서 챔피언을 일일이 찾아서 눌러주어야 하기가 힘들다. 챔피언 검색 기능이 추가된다면, 이 시간을 줄여 더 실용적인 프로그램이 될 것이다.