

Description

(함수 이름은 Bold 처리 하였다.)

def dist(point 1, point 2) , return 값은 두 점사이의 거리

입력받은 point1과 point2를 이용하여 두 점의 사이의 거리를 구하겠다. 두 점은 x,y 좌표로 이루어져 있다. (아닐 수도 있는데, 확실한건 한 점당 두 개의 요소를 포함한다,) 두 점 사이의 거리는 피타고拉斯의 정리를 이용하여 구한다.

def rect(r, theta) , return 값은 x,y좌표

입력받은 r과 theta를 이용하여 x,y좌표로 바꾸어준다. 즉, 극좌표계를 xy좌표계로 바꿔주는 함수.

theta는 degree(도)를 값을 가진다. x,y 좌표계를 바꾸어줄 때는 degree 값을 radian 값으로 바꿔서 계산한다.

def polar(x,y), return 값은 r,theta 값

입력받은 x,y좌표를 r과 theta로 나타내어 준다. 즉, xy좌표계를 극좌표계로 바꾸어주는 함수. 여기서리턴하는 theta는 degree(도) 단위를 가진다.

def angle_mod_360(angle)

입력받은 angle의 값을 -180 ~ 180도 사이의 값으로 표현해주도록 바꾸어주는 함수. 계산 과정은 대충 밑의 코드를 보면 충분히 이해 가능하다.

def get_waypoints_ordered_in_driving_direction(parms), return 값은 waypoint. (즉 list 값을 리턴)

parms란 입력변수는 deepracer가 내부적으로 가지고 있는 변수들을 함수? 리스트? 로 만들어 놓은 것 같다. 여기에는 x,y좌표, waypoints, 트랙의 넓이, 조향 각도, 조향 머리가 향하는 방향, 거꾸로 가고 있는지 를 포함하고 있다. 일단, 우리는 입력받은 parms 중에서 is_reversed(거꾸로 가고 있는지)와 waypoints(리스트 타입)을 사용할 것이다. 만약 차량이 거꾸로 가고 있다면 아마도 ? 내 생각임 : waypoint의 순서를 바꾸어 준후 1234 면 4321 이런 식으로 그리고 list로 묶어서 리턴해준다.

제대로 가고 있다면 그냥 parms 에 있는 waypoint들을 그대로 리턴 해주면 된다.

즉, 이 함수는 차량의 진행방향에 따라 알맞은 waypoint들을 리턴해주는 함수 인듯 싶다.

def up_sample(waypoints,factor)

제공받은 waypoint들 사이에 추가적은 waypoint를 더하는 함수라고 설명되어있음. 아마도 waypoint들을

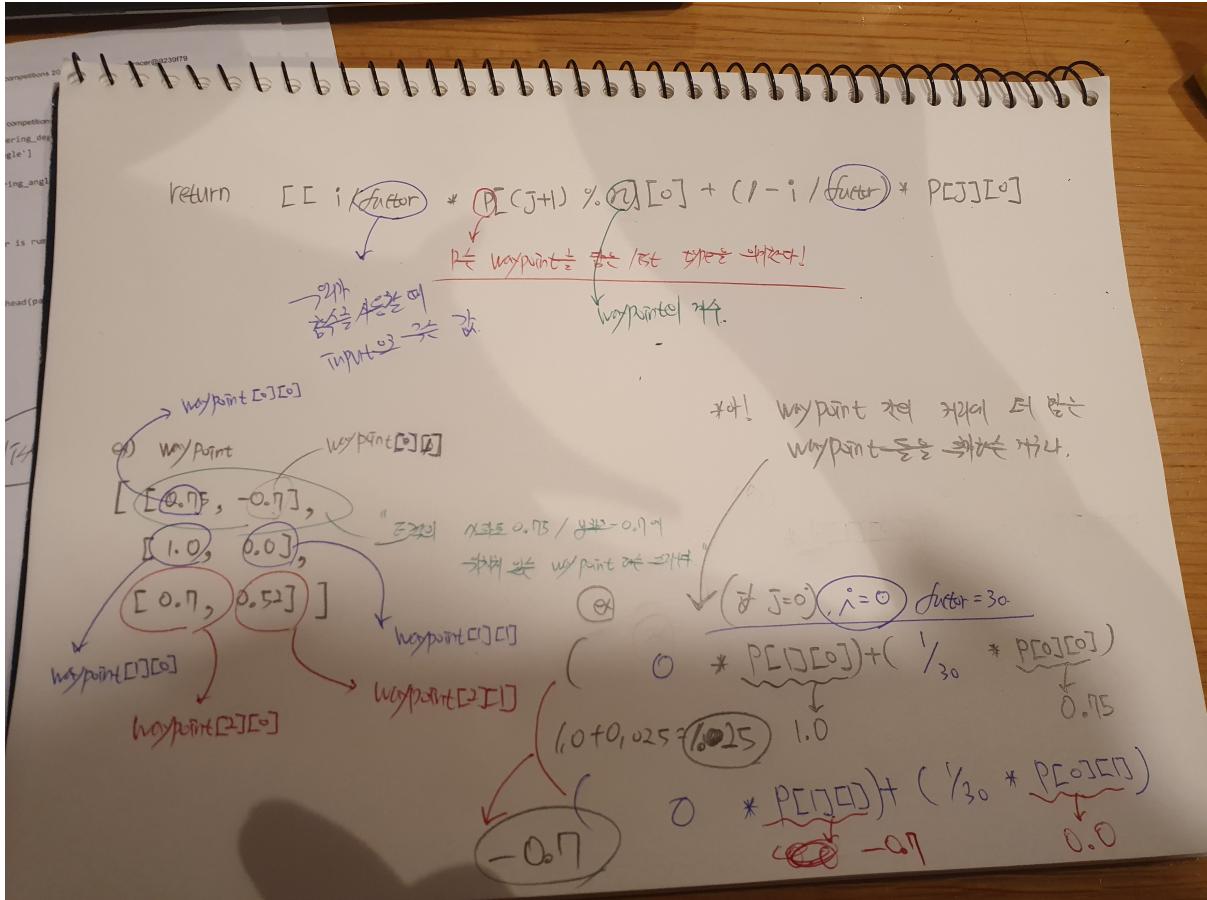
더 세분화해서 쪼개는 작업을 하는 코드인 듯 싶다.

`p = waypoints // 그럼 p도 list 타입이 되는 것 같음`

`n = len(p) // waypoint의 개수를 n이라고 하겠다는 뜻같음. waypoint 자체는 list타입이니까`

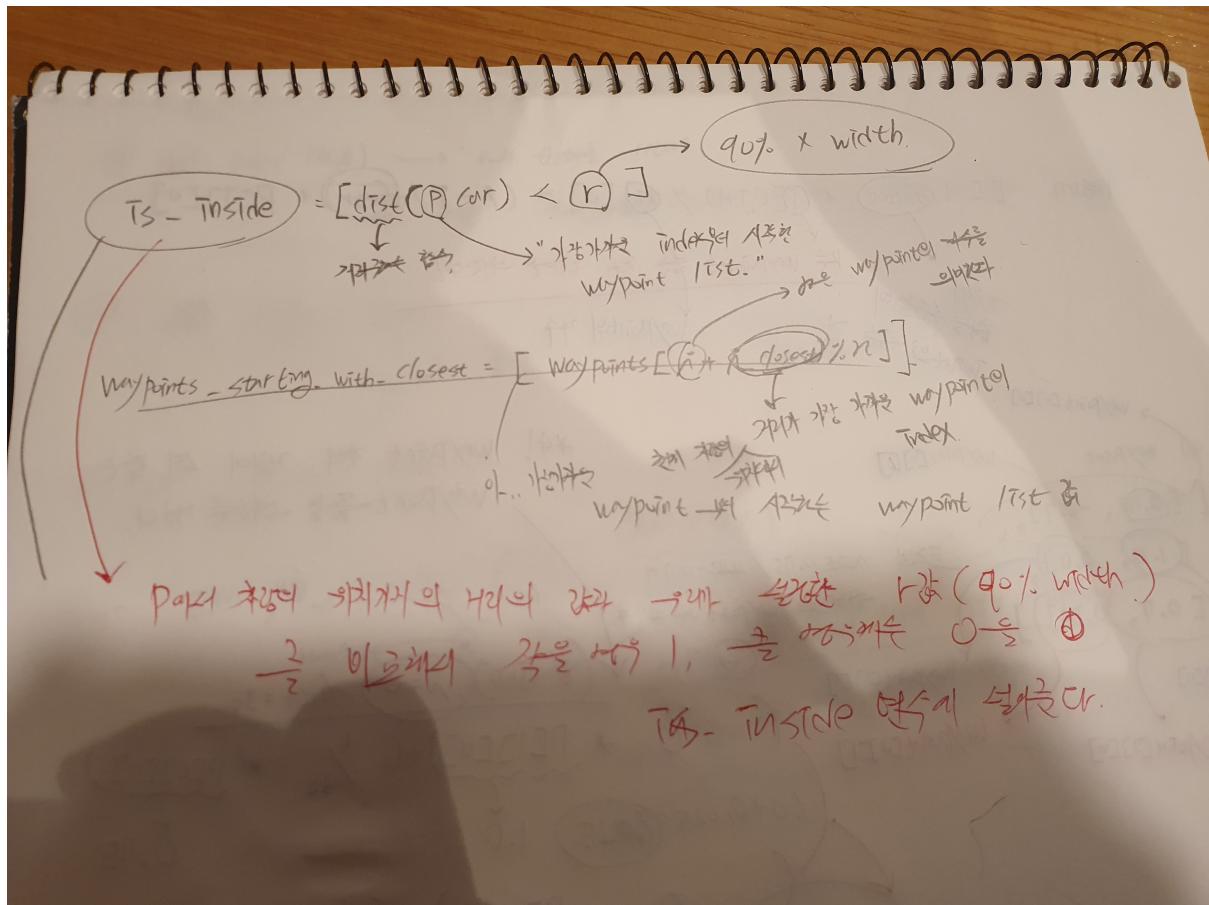
`factor`는 함수를 사용할 때 마다, 그 때 그때 인수로 넣어주면 되는 값이다.

(ex. `up_sample(waypoints,30)`)



```
def get_target_point(parms)
```

deepracer에 내장되어 있는 내부 parameter를 집단적으로 가지고 있는 parms를 input 값으로 받는다. 이는 곧 필요할 때 마다, parms에 있는 값을 사용하겠다는 뜻이 된다. 처음으로 waypoints 변수에 `up_sample` 함수를 사용하여 세분화 된 waypoint를 얻는다. 그리고 car변수에 현재 deepracer의 위치 좌표를 담아준다. distances라는 우리가 세분화한 waypoint와 deepracer간의 거리를 담는 리스트변수를 만들어 준다. 그리고min_dist라는 변수에는 앞에서 선언한 변수 distances 중 가장 작은 값을 담아준다. 그 후, `i_closest`라는 변수에는 distances 변수 중 가장 작은 값의 index를 담아준다. n변수에는 세분화된 waypoints의 개수를 담아준다.r변수는 트랙 넓이의 90% 값을 담아준다. 아마 우리가 만들 가상의 원의 반지름의 값인듯 싶다.

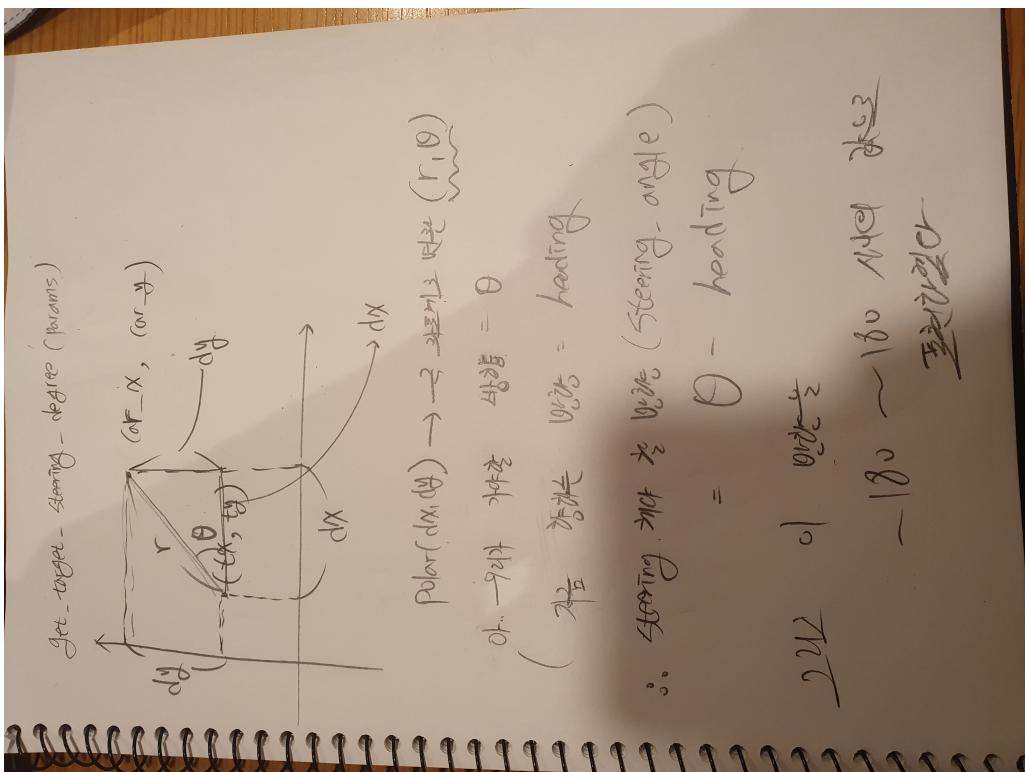


i_fisrt_outside 에는 is_inside 변수에 있는 값들 중 0인 값의 인덱스, 즉 처음으로 waypoint 와 차량의 거리가 반지름의 크기보다 커지는 그 때의 index값을 i_fisrt_outside변수에 담아준다. 그래서 만약 i_first_outside가 0보다 작다면, 다시 말해서 반지름 r보다 큰 값을 가지는 거리의 값이 없는 경우에는 waypoints[i_closest]를 return 해준다.

그렇지 않을 경우에는 waypoints_starting_with_closest[i_first_outside]값을 return 해준다.

def get_target_steering_degree(params)

위의 값과 동일하게 params를 input으로 받아준다. tx,ty변수에는 위에서 정의한 get_target_point함수를 통해 얻은 waypoint 값들 중 하나(어떤 값이 들어갈지는 get_target_point 함수에 정의되어 있다.)로 각각 tx와 ty에 담아주도록 한다. car_x에는 차량의 x좌표, car_y에는 차량의 y좌표. heading에는 deepracer의 머리가 바라보고 있는 방향의 값을 담아준다



def score_steer_to_point_ahead

best_steering_angle 변수 값에는 바로 위에 있는 함수 값을 통해 얻은 리턴 값을 넣어주도록 한다.

그 다음 steering angle 변수 값에는 deepracer 자체적으로 가지게 될 변수 parms 중 steering_angle 값을 넣어준다. error라는 변수를 설정해서 현재 우리의 차량의 조향각과 우리가 지향해야 할 조향각의 차 이를 구한 후 60으로 나누어준다. score 는 1.0에서 error를 빼준만큼 부여한다. 즉, error가 클수록 reward를 적게 주고, 우리가 가야할 조향 방향과 현재 차량의 조향방향 간에 차이가 적을 수록 더 큰 reward를 부여하도록 하겠다. max함수를 통하여 만약 reward가 0.01보다 작을 경우는 0.01를 reward로 받도록 만든다. 이는 수렴성을 고려하여 그렇게 한 듯 싶다.

(참고로 여기서 나오는 수렴성의 개념은 굉장히 중요하다. DeepRacing 주행에서 더 이상 주행 성적이 요동치지 않고 꾸준한 결과 나오는 그런 학습 데이터가 쌓였다고 생각하면 된다. 사실, 나도 강화학습에 대해서는 제대로 이해하지 못하고 있어, 나의 개념이 틀렸을 수도 있다. 나는 이렇게 생각했다 라는 것 정도로 참고하면 좋을 것 같다. 관심이 있으면 강화학습 강의를 들어보는 것을 추천한다.)

강화학습 관련 수업 링크 :

RL Course by David Silver - Lecture 1: Introduction to Reinforcement Learning
 Reinforcement Learning Course by David Silver# Lecture 1: Introduction to Reinforcement Learning #Slides and more info about the course: <http://goo.gl/vUiyjq>
 ▶ <https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ>



(알파고를 만든 Google DeepMind사의 silver 교수의 강의이다. 나는 4강까지 들었으며, 들던 중 DeepRacing에 적용하기는 너무 어려운 개념인 듯 싶어 필요할 것 같은 내용들만 선별해서 학습했다. 관심이 있다면 보는 것도 좋다.)