In [546]:
```
from racecar.SDRaceCar import SDRaceCar
import numpy as np
import matplotlib.pyplot as plt
```

In [547]:
```
env = SDRaceCar(render_env=False, track='Circle')
#env.render()
state = env.reset()
```

In [548]:
```
def mse (referencex , generatedx , referencey , generatedy):
    """
    MSE error
    :param referencex: xreference trajectory
    :param generatedx : xgenerated trajectory
    :param referencey: yreference trajectory
    :param generatedy : ygenerated trajectory
    :return: MSE error
    """
    sum = 0
    for i in range(0, len(referencex)):
        sum = sum + (referencex[i] - generatedx[i]) ** 2 + (reference
y[i] - generatedy[i]) ** 2
    return np.sqrt(sum/len(referencex))
```

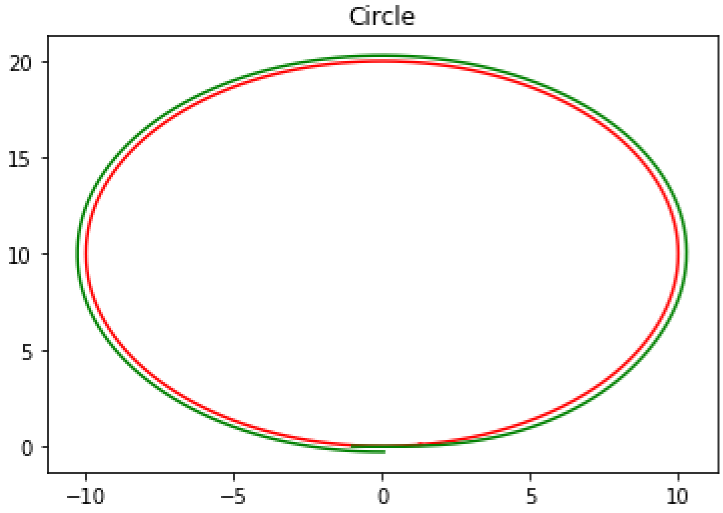
In [549]:
```
def taninverse(x1,x2,y1,y2):
    if(y2 > y1 and x2 < x1):
        angle = np.arctan((y2 - y1)/(x2 - x1)) + np.pi
    if(y2 < y1 and x2 < x1):
        angle = np.pi + np.arctan((y2 - y1)/(x2 - x1))
    if(y2 < y1 and x2 > x1):
        angle = np.arctan((y2 - y1)/(x2 - x1))
    if(y2 > y1 and x2 > x1):
                    #obtuse
        angle = np.arctan((y2 - y1)/(x2 - x1))
    return angle
```

In [552]:

```
def racecar(k_p, k_d, input_signal = "Circle"):
    """
    Racecar steps
    :param k_p : control position
    :param k_d : control velocity
    :param mass: Mass
    """
    env = SDRaceCar(render_env=True, track=input_signal)
    l_r = env.l_r
    l_f = env.l_f
    mass = env.m

    x = []
    y = []
    xref = []
    yref = []
    previous_ind = 0
    steps = 0
    done = False
    return_states = env.reset()
    pos_x = return_states[0];
    pos_y = return_states[1];
    psi   = return_states[2];
    v_x   = return_states[3];
    v_y   = return_states[4];
    omega = return_states[5];
    h     = return_states[6];

    x.append(pos_x)
    y.append(pos_y)
    xref.append(h[0])
    yref.append(h[1])

    while not done:



        del_x , del_y = h[0] - pos_x, h[1] - pos_y
        v = np.sqrt(v_x*v_x + v_y*v_y)
        theta = np.arctan2(del_y, del_x)
        w_angle = theta - psi
        #print(theta,w_angle)
        if w_angle < -np.pi:
            w_angle += 2*np.pi
        elif w_angle > np.pi:
            w_angle -= 2*np.pi
        w_angle = w_angle * 2/np.pi;

        e = np.sqrt(del_x*del_x + del_y*del_y)

        v_ref = np.array([np.sqrt((del_x*del_x + del_y*del_y) / (np.c
os(w_angle))**2)])
        v_e = v_ref - v
        thrust = k_p*e + k_d*v_e

        thrust = np.clip(thrust, 0, 20).item()
        thrust = (thrust/10) - 1
        env.step([w_angle,thrust])


        return_states = env.get_observation();
        pos_x = return_states[0];
        pos_y = return_states[1];
        psi   = return_states[2];
        v_x   = return_states[3];
        v_y   = return_states[4];
        omega = return_states[5];
        h     = return_states[6];


        #pos_ref =  env.track[:,current_ind];
        x.append(pos_x)
        y.append(pos_y)
        xref.append(h[0])
        yref.append(h[1])


        steps+= 1
        current_ind = env.closest_track_ind
        # CONDITION TO CHECK lap-completion
        if current_ind - previous_ind<=-500:
            done =True
        previous_ind = current_ind

    print("The MSE error is", mse(xref,x, yref, y))
    plt.plot(xref, yref, color='r')
    plt.plot(x,y, color = 'g')
    plt.title(input_signal)


    return steps
```

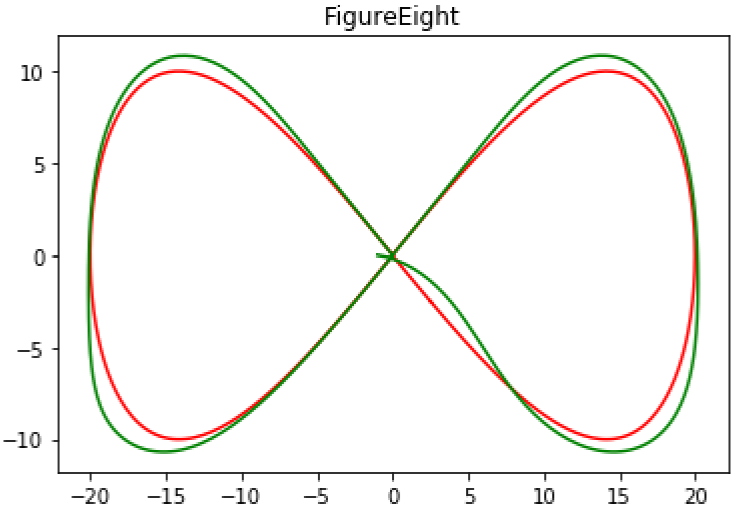
In [563]:
```
step = racecar(2.8, 1.3, input_signal = 'Circle')
print("Steps taken in circle is",step)
```

```
The MSE error is 1.3378962380494839
Steps taken in circle is 338
```



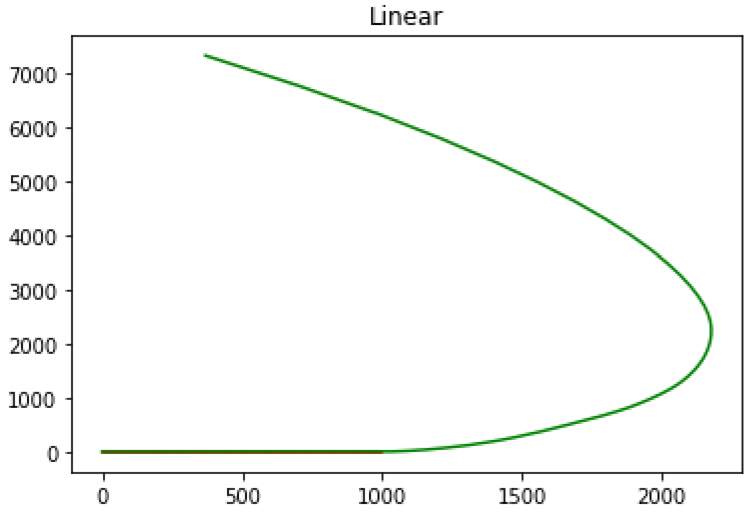
In [565]:
```
step = racecar(3, 2.8, input_signal = 'FigureEight')
print("Steps taken in FigureEight is",step)
```

```
The MSE error is 2.7257063486832975
Steps taken in FigureEight is 443
```



```
In [556]: step = racecar(3, 2.8, input_signal = 'Linear')
          print("Steps taken in Linear is",step)
```

```
The MSE error is 1177.8901565383424
Steps taken in Linear is 830
```

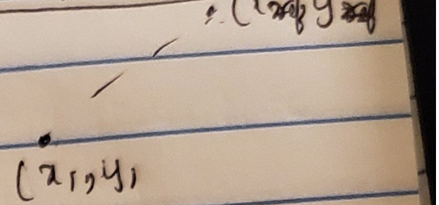


The linear does not come close to the trajectory since the initial reset position H is different from the -1,0 set as the initial position of the bot.

Racecar Question

~~We use a~~ The model implmented here is that of a PD controller. In which the ~~Torque~~ Thrust applied is equal to

$$\boxed{\text{Thrust} = K_p\, e + K_d\, v\_e}\,, \quad \boxed{K_p = 3,\ K_d = 3.1}$$



We compute the error by taking

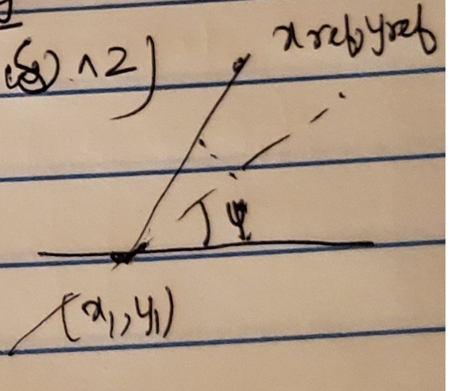
$$e = \sqrt{(x_{ref} - x_1)^2 + (y_{ref} - y_1)^2}$$

x_{ref} and y_{ref} are determined by the h[0] and h[1]

The V_{ref} is computed by $\sqrt{\dfrac{\Delta x^2 + \Delta y^2}{(np.cos(\delta)^2)}}$

The wheel angle is computed by

$$\delta = \tan^{-1}\left(\frac{y_{ref} - y_1}{x_{ref} - x_1}\right) - \psi$$



v_e is computed by $\boxed{V_e = V_{ref} - V_{observed}}$