```
from cartpole import CartPoleEnv
import math
import numpy as np

env = CartPoleEnv()
env.reset()

def discretize(val,bounds,n_states):
    discrete_val = 0
    if val <= bounds[0]:
        discrete_val = 0
    elif val >= bounds[1]:
        discrete_val = n_states-1
    else:
        discrete_val = int(round((n_states-1)*((val-bounds[0])/(bounds[1]-bounds[0]))))
    return discrete_val

def discretize_state(vals,s_bounds,n_s):
    discrete_vals = []
    for i in range(len(n_s)):
        discrete_vals.append(discretize(vals[i],s_bounds[i],n_s[i]))
    return np.array(discrete_vals,dtype=np.int)

# położenie, prędkość, kąt, prędkość kątowa
n_s = np.array([10,10,10,10])

#tablica zwierająca granice przedziałów
s_bounds = np.array(list(zip(env.observation_space.low,env.observation_space.high)))
s_bounds[1] = (-1.0,1.0)
s_bounds[3] = (-1.0,1.0)
#konieczna konwersja typu
s_bounds = np.dtype('float64').type(s_bounds)

episodes=1000
gamma=0.9
alpha=0.6

V = np.zeros(n_s)


for i in range(episodes):

    print("episode=",i)

    obs = env.reset()
    s = discretize_state(obs,s_bounds,n_s)

    finished = False
    time_step=0


    #dlugie epizody przerywamy po 200 krokach
    while not finished and not time_step==200:
        #polityka stochastyczna - prawdopodobienstwo 0.5 dla obu akcji (w lewo, w prawo)
        action = np.random.randint(0,2)
        obs, reward, finished, info = env.step(action)
        state_new = discretize_state(obs,s_bounds,n_s)
        V[s] = V[s] + alpha * (reward + gamma*V[state_new] - V[s])
        s = state_new

        time_step+=1



        #DO UZUPEŁNIENIA
print(V)
```

```
episode= 984
episode= 985
episode= 986
episode= 987
episode= 988
episode= 989
episode= 990
episode= 991
episode= 992
episode= 993
episode= 994
episode= 995
episode= 996
episode= 997
episode= 998
episode= 999
[[[[10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   ...
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]]

  [[10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   ...
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]]

  [[10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   ...
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]]

  ...

  [[10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   ...
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]]

  [[10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   ...
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]
   [10. 10. 10. ... 10. 10. 10.]]

  [[10. 10. 10. ... 10. 10. 10.]
```