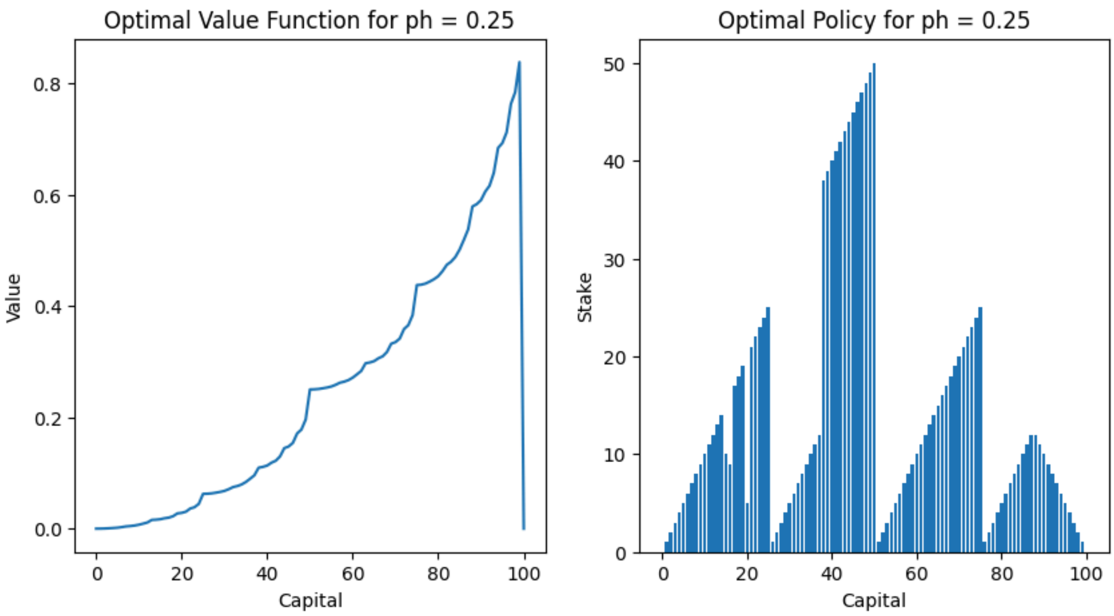
NAME: RUPESH DHIRWANI

CLASS: D16AD

ROLL NO: 10

```
import numpy as np
import matplotlib.pyplot as plt

def value_iteration(ph, theta=1e-6, max_iter=1000, gamma=1.0, goal=100):
    V = np.zeros(goal + 1)

    def one_step_lookahead(s, V, ph):
        actions = np.arange(1, min(s, goal - s) + 1)
        action_returns = np.zeros(actions.shape)
        for a in actions:
            action_returns[a - 1] = ph * (1 if s + a >= goal else V[s + a]) + (1 - ph) * V[s - a]
        return action_returns.max()

    for _ in range(max_iter):
        delta = 0
        for s in range(1, goal):
            v = V[s]
            V[s] = one_step_lookahead(s, V, ph)
            delta = max(delta, abs(v - V[s]))
        if delta < theta:
            break

    policy = np.zeros(goal + 1)
    for s in range(1, goal):
        actions = np.arange(1, min(s, goal - s) + 1)
        action_returns = np.zeros(actions.shape)
        for a in actions:
            action_returns[a - 1] = ph * (1 if s + a >= goal else V[s + a]) + (1 - ph) * V[s - a]
        policy[s] = actions[np.argmax(action_returns)]

    return V, policy

def plot_results(V, policy, ph):
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.plot(V)
    plt.title("Optimal Value Function for ph = {}".format(ph))
    plt.xlabel("Capital")
    plt.ylabel("Value")

    plt.subplot(1, 2, 2)
    plt.bar(range(len(policy)), policy)
    plt.title("Optimal Policy for ph = {}".format(ph))
    plt.xlabel("Capital")
    plt.ylabel("Stake")
    plt.show()

ph_values = [0.25, 0.55]

for ph in ph_values:
    V, policy = value_iteration(ph)
    plot_results(V, policy, ph)
    print("Optimized Policy:")
    print(policy)
    print("")

    print("Optimized Value Function:")
    print(V)
    print("")
```




```
Optimized Policy:
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 10.  9. 17.
 18. 19.  5. 21. 22. 23. 24. 25.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.
 11. 12. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50.  1.  2.  3.
  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21.
 22. 23. 24. 25.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 12. 11.
 10.  9.  8.  7.  6.  5.  4.  3.  2.  1.  0.]

Optimized Value Function:
[0.00000000e+00 7.28459563e-05 2.91426259e-04 6.95257448e-04
 1.16577863e-03 1.77125235e-03 2.78105366e-03 4.03661077e-03
 4.66311452e-03 5.60140889e-03 7.08500942e-03 9.04085029e-03
 1.11242325e-02 1.56796345e-02 1.61464431e-02 1.69534393e-02
 1.86524581e-02 1.98260567e-02 2.24056377e-02 2.73847259e-02
```

```
 6.45857902e-02 6.59973359e-02 6.78137571e-02 7.08431744e-02
 7.46098323e-02 7.64893436e-02 7.93042283e-02 8.37550596e-02
 8.96226583e-02 9.58726976e-02 1.09538927e-01 1.10939343e-01
 1.13360318e-01 1.18457374e-01 1.21978171e-01 1.29716994e-01
 1.44654195e-01 1.47520238e-01 1.53983628e-01 1.70990647e-01
 1.77987721e-01 1.95990791e-01 2.50000000e-01 2.50218570e-01
 2.50874334e-01 2.52085790e-01 2.53497336e-01 2.55313757e-01
 2.58343174e-01 2.62109832e-01 2.63989344e-01 2.66804228e-01
 2.71255060e-01 2.77122658e-01 2.83372698e-01 2.97038927e-01
 2.98439343e-01 3.00860318e-01 3.05957374e-01 3.09478171e-01
 3.17216994e-01 3.32154195e-01 3.35020238e-01 3.41483628e-01
 3.58490647e-01 3.65487721e-01 3.83490791e-01 4.37500000e-01
 4.38155750e-01 4.40123002e-01 4.43757381e-01 4.47992008e-01
 4.53441295e-01 4.62529523e-01 4.73829507e-01 4.79468031e-01
 4.87912745e-01 5.01265179e-01 5.18867985e-01 5.37618093e-01
 5.78616813e-01 5.82818036e-01 5.90080971e-01 6.05372130e-01
 6.15934559e-01 6.39150989e-01 6.83962610e-01 6.92560728e-01
 7.11950919e-01 7.62971957e-01 7.83963189e-01 8.37972392e-01
 0.00000000e+00]
```