Reinforce Learning 6885 Final Project

Mount the Google Drive onto the Colab as the storage location.

Following the instructions returned from the below cell. You will click a web link and select the google account you want to mount, then copy the authorication code to the blank, press enter.

```
# This must be run within a Google Colab environment
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
```

Append the directory location where you upload the start code folder to the sys.path

```
import sys
# sys.path.append('/content/gdrive/My Drive/RL/.')
sys.path.append('/content/gdrive/My Drive/RL/Final_Project.')
```

```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

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

Cd your path to where Final_Project is stored

```
cd /content/gdrive/My Drive/RL/Final_Project
```

```
/content/gdrive/My Drive/RL/Final_Project
```

```
ls
```

```
"Chongzhi's Strategy.py"   Final_Project.ipynb  "Yinsen's Strategy.py"
```

```
Epsilon_RPSLS.py        pictures/           "Zifan's Strategy.py"
Epsilon_Strategy.py     __pycache__/
```

Part1 ROCK! PAPER! SCISSORS!

1. ϵ-Greedy Strategy Environment for ROCK! PAPER! SCISSORS!

In Reinforcement Learning, we are always faced with the dilemma of exploration and exploitation. ϵ-Greedy is a trade-off between them.

```
from Epsilon_Strategy import Epsilon_Strategy, Epsilon_Strategy_RPSLS
```

```
epsilon_greedy_strategy = Epsilon_Strategy()
```

2. Zifan's Strategy Environment for ROCK! PAPER! SCISSORS!

3. Chongzhi's Strategy Environment for ROCK! PAPER! SCISSORS!

4. Yinsen's Strategy Environment for ROCK! PAPER! SCISSORS!

5. Player V.S. Environment

```
def Game_Result(play1, play2):
    """
    Default:  play1 is made by user, play2 is made by computer,
    Record plays in opponent and self history list respectively,
    Return the result in string for print
    """
    global user_win_times, computer_win_times

    if (play1 == "R" and play2 == "S") or (play1 == "P" and play2 == "R")
        user_win_times += 1
        winner = 'User'

    elif (play1 == "R" and play2 == "P") or (play1 == "P" and play2 == "S
        computer_win_times += 1
        winner = 'Computer'

    else:
        return "Tie"

    result_template = "{0} is winner, user win rate is {1:.2f}%"
    return result_template.format(winner, 100 * user_win_times / (user_wi


import sys

RPS_list = ["R", "P", "S"]
opponent_history = []
self_history = []
user_win_times = 0
computer_win_times = 0

Computer_player = Epsilon_Strategy()
while True:
    print("\nEnter your play, 'R' or 'P' or 'S', or enter 'E' to end")
    user_play = input()
    if user_play in RPS_list:
        computer_play = Computer_player.Next_Move()
        result = Game_Result(user_play, computer_play)
        opponent_history.append(user_play)
```

```
            self_history.append(computer_play)
            Computer_player.Learn(opponent_history, self_history)
            print('User_play: {0}, computer_play: {1}, {2}'.format(user_play,
        elif user_play in ['exit','EXIT','e','E','quit','QUIT','q','Q']:
            sys.exit()
        else:
            print('Input is invalid!!!')


        Enter your play, 'R' or 'P' or 'S', or enter 'E' to end
        E
        An exception has occurred, use %tb to see the full traceback.

        SystemExit

        SEARCH STACK OVERFLOW
        /usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2890:
          warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

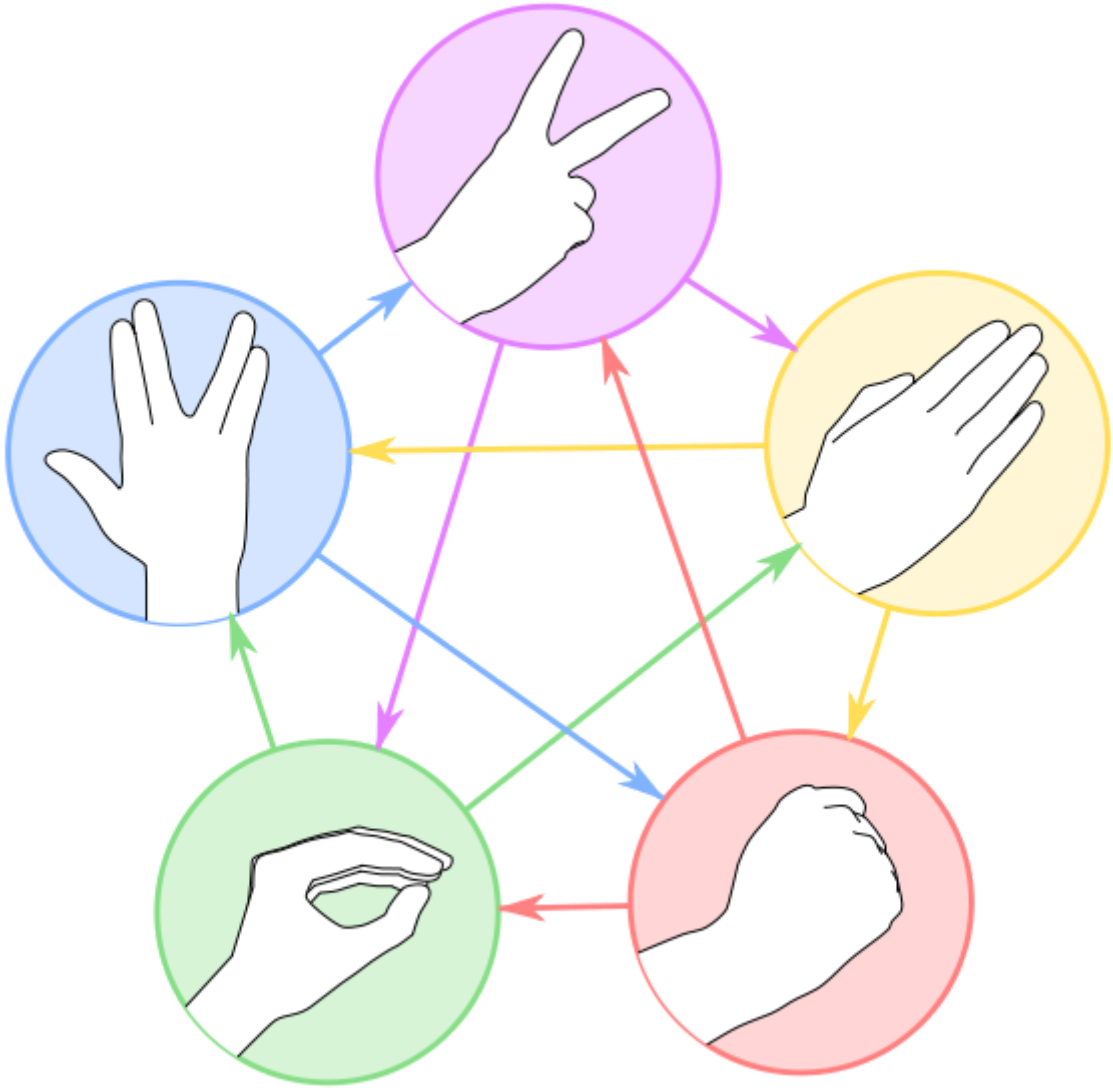
6. Environment V.S. Environment

```
def Competetion_Result(play1, play2):
    """
    Record plays in opponent and self history list respectively,
    Return the result in string for print
    """
    global Strategy1_win_times, Strategy2_win_times

    if (play1 == "R" and play2 == "S") or (play1 == "P" and play2 == "R")
        Strategy1_win_times += 1
        winner = 'Strategy1'

    elif (play1 == "R" and play2 == "P") or (play1 == "P" and play2 == "S
        Strategy2_win_times += 1
        winner = 'Strategy2'

    else:
        return "Tie"

    result_template = "{0} is winner, now scores are {1}:{2}"
    return result_template.format(winner, Strategy1_win_times, Strategy2_


import sys

RPS_list = ["R", "P", "S"]
Strategy1_history = []
Strategy2_history = []
Strategy1_win_times = 0
Strategy2_win_times = 0
```

```
#-------------------------------------------------------------------------
# Assign episodes to ends. Change this!!!
Max_episodes = 1000
episodes = 1
# Assign two different player, from different strategy. Change this!!!
Strategy1 = Epsilon_Strategy(epsilon=0.5,min_eps=0.2,episodes_to_min_eps=
Strategy2 = Epsilon_Strategy(epsilon=0.2,min_eps=0.01,episodes_to_min_eps
#-------------------------------------------------------------------------
while episodes <= Max_episodes:
    Strategy1_play = Strategy1.Next_Move()
    Strategy2_play = Strategy2.Next_Move()
    result = Competetion_Result(Strategy1_play, Strategy2_play)
    Strategy1_history.append(Strategy1_play)
    Strategy2_history.append(Strategy2_play)
    Strategy1.Learn(Strategy1_history, Strategy2_history)
    Strategy2.Learn(Strategy1_history, Strategy2_history)
    if result != 'Tie':
      if episodes == Max_episodes:
        print('{0}th round, Strategy1: {1}, Strategy2: {2}, {3}'.format(e
      episodes += 1

if Strategy1_win_times > Strategy2_win_times:
  print("Final winner is Strategy1")
else:
  print("Final winner is Strategy2")
```

```
1000th round, Strategy1: S, Strategy2: P, Strategy1 is winner, now scores are
Final winner is Strategy2
```

Part2 ROCK! PAPER! SCISSORS! LIZARD! SPOCK!



Game Rules

- **Scissors cuts Paper**
- **Paper covers Rock**
- **Rock crushes Lizard**
- **Lizard poisons Spock**
- **Spock smashes Scissors**
- **Scissors decapitates Lizard**
- **Lizard eats Paper**
- **Paper disproves Spock**
- **Spock vaporizes Rock**
- **Rock crushes Scissors**

> "I invented this game (with Karen Bryla) because it seems like when you know someone well enough, 75-80% of any Rock-Paper-Scissors games you play with

> that person end up in a tie. Well, here is a slight variation that reduces that probability."

> --Sam Kass1

1. ϵ-Greedy Strategy Environment for ROCK! PAPER! SCISSORS! LIZARD! SPOCK!

```
epsilon_greedy_strategy_RPSLS = Epsilon_Strategy_RPSLS()
```

2. Zifan's Strategy Environment for ROCK! PAPER! SCISSORS! LIZARD! SPOCK!

3. Chongzhi's Strategy Environment for ROCK! PAPER! SCISSORS! LIZARD! SPOCK!

4. Yinsen's Strategy Environment for ROCK! PAPER! SCISSORS! LIZARD! SPOCK!

5. Player V.S. Environment

```
def Game_Result_RPSLS(play1, play2):
    """
    Default:  play1 is made by user, play2 is made by computer,
    Record plays in opponent and self history list respectively,
    Return the result in string for print
    """
    global user_win_times_RPSLS, computer_win_times_RPSLS
```

```
        if play2 in RPSLS_counter_dictionary[play1]:
            user_win_times_RPSLS += 1
            winner = 'User'

        elif play1 in RPSLS_counter_dictionary[play2]:
            computer_win_times_RPSLS += 1
            winner = 'Computer'

        else:
            return "Tie"

        result_template = "{0} is winner, user win rate is {1:.2f}%"
        return result_template.format(winner, 100 * user_win_times_RPSLS / (u


    Computer_player = Epsilon_Strategy_RPSLS()
    RPSLS_list = ["Rock", "Paper", "Scissors", "Lizard", "Spock"]
    RPSLS_counter_dictionary = {
                                "Rock": ["Scissors", "Lizard"],  # Rock crush
                                "Paper": ["Rock", "Spock"],  # Paper covers R
                                "Scissors": ["Paper", "Lizard"],  # Scissors
                                "Lizard": ["Paper", "Spock"],  # Lizard eats
                                "Spock": ["Rock", "Scissors"]  # Spock vapori
                               }
    opponent_history_RPSLS = []
    self_history_RPSLS = []
    user_win_times_RPSLS = 0
    computer_win_times_RPSLS = 0
    while True:
        print("\nEnter your play, 'Rock' or 'Paper' or 'Scissors' or 'Lizard
        user_play = input()
        if user_play in RPSLS_list:
            computer_play = Computer_player.Next_Move()
            result = Game_Result_RPSLS(user_play, computer_play)
            opponent_history_RPSLS.append(user_play)
            self_history_RPSLS.append(computer_play)
            Computer_player.Learn(opponent_history_RPSLS, self_history_RPSLS)
            print('User_play: {0}, computer_play: {1}, {2}'.format(user_play,
        elif user_play in ['exit','EXIT','e','E','quit','QUIT','q','Q']:
            sys.exit()
        else:
            print('Input is invalid!!!')
```

```
Enter your play, 'Rock' or 'Paper' or 'Scissors' or 'Lizard or 'Spock
-
```

6. Environment V.S. Environment

```
def Competetion_Result_RPSLS(play1, play2):
    """
    Record plays in opponent and self history list respectively,
    Return the result in string for print
    """
    global Strategy1_win_times_RPSLS, Strategy2_win_times_RPSLS

    if play2 in RPSLS_counter_dictionary[play1]:
        Strategy1_win_times_RPSLS += 1
        winner = 'Strategy1'

    elif play1 in RPSLS_counter_dictionary[play2]:
        Strategy2_win_times_RPSLS += 1
        winner = 'Strategy2'

    else:
        return "Tie"

    result_template = "{0} is winner, now scores are {1}:{2}"
    return result_template.format(winner, Strategy1_win_times_RPSLS, Stra


import sys

RPSLS_list = ["Rock", "Paper", "Scissors", "Lizard", "Spock"]
RPSLS_counter_dictionary = {
            "Rock": ["Scissors", "Lizard"],  # Rock crushes Scissors, Roc
            "Paper": ["Rock", "Spock"],  # Paper covers Rock, Paper dispr
            "Scissors": ["Paper", "Lizard"],  # Scissors cuts Paper, Scis
            "Lizard": ["Paper", "Spock"],  # Lizard eats Paper, Lizard po
            "Spock": ["Rock", "Scissors"]  # Spock vaporizes Rock, Spock
        }
Strategy1_history_RPSLS = []
Strategy2_history_RPSLS = []
Strategy1_win_times_RPSLS = 0
Strategy2_win_times_RPSLS = 0
#-----------------------------------------------------------------------
# Assign episodes to ends. Change this!!!
Max_episodes = 1000
episodes = 1
# Assign two different player, from different strategy. Change this!!!
Strategy1_RPSLS = Epsilon_Strategy_RPSLS(epsilon=0.6,min_eps=0.2,episodes
Strategy2_RPSLS = Epsilon_Strategy_RPSLS(epsilon=0.1,min_eps=0.01,episode
#-----------------------------------------------------------------------
```

```
while episodes <= Max_episodes:
    Strategy1_play = Strategy1_RPSLS.Next_Move()
    Strategy2_play = Strategy2_RPSLS.Next_Move()
    result = Competetion_Result_RPSLS(Strategy1_play, Strategy2_play)
    Strategy1_history_RPSLS.append(Strategy1_play)
    Strategy2_history_RPSLS.append(Strategy2_play)
    Strategy1_RPSLS.Learn(opponent_history_RPSLS, self_history_RPSLS)
    Strategy2_RPSLS.Learn(opponent_history_RPSLS, self_history_RPSLS)
    if result != 'Tie':
      if episodes == Max_episodes:
        print('{0}th round, Strategy1: {1}, Strategy2: {2}, {3}'.format(e
      episodes += 1

if Strategy1_win_times_RPSLS > Strategy2_win_times_RPSLS:
  print("Final winner is Strategy1")
else:
  print("Final winner is Strategy2")
```

```
1000th round, Strategy1: Lizard, Strategy2: Scissors, Strategy2 is winner, now
Final winner is Strategy1
```