1 MDP class

In this task you will first formulate an arbitrary MDP. To simplify the process, we have created the **MDP** class for you. To initialize the MDP, create an MDP object from the respective class with the following information:

```
mdp_object = MDP(
    states=[[state_variable_1],[state_variable_2],[...]],
    actions = [action_1, action_2, ...],
    r=-1 # default reward
)
```

MDP will then used your input states and actions to create an empty matrix of P and R. The matrix has the following structure:

```
# P[new_state, current_state, action] = probability
mdp_object.P.shape = (num_state, num_state, num_action)
# R[state, action] = reward, default to -1
mdp_object.R.shape = (num_state, num_action)
```

The following is a list of internal variables of this class that you can access:

```
MDP().a              # list of actions
MDP().s              # list of state variables
MDP().num_s          # number of state
MDP().num_s_vars     # number of state variables
MDP().num_a          # number of action
MDP().P              # transition matrix
MDP().R              # reward matrix
```

The following functions are supported in this class:

```
# Add new route to current MDP object
add_route(current_state, action, new_state, p=1.0)

# Add new reward to current MDP routes
add_reward(state, action, reward)

# Get state indexing in MDP().P and MDP().R
get_index(state)

# Get state in index form from index
get_state(index)

# Get state in real state variable value from from index
get_real_state_value(index)

# Get basic MDP information
# Output: num_a, num_s, R, P
get_mdp()
```

The class MDP will help you to index and manage routes and rewards within the MDP easier. Assuming that you have an MDP with 2 state variables $s_1 = \{restaurant, supermarket\}, s_2 = \{vacant, full\}$ and 4 different actions to take $a = \{left, right, forward, backward\}$, simply create and index your MDP as follows:

```
mdp_object = MDP(
    states=[["restaurant","supermarket"],["vacant","full"]],
    actions=["left","right","forward","backward"]
)
# to say that the probability to go to
# s_new=[supermarket,vacant] from s_cur=[restaurant,full]
# when taking action a=forward is 0.8, simply call
mdp_object.add_route(
    ["restaurant","full"],
    "forward",
    ["supermarket","vacant"]
) = 0.8
```

Let's have a simple example to show you how this class works. Assuming that we want to describe the MDP in figure 1, the following code describes the respective MDP using our MDP class:

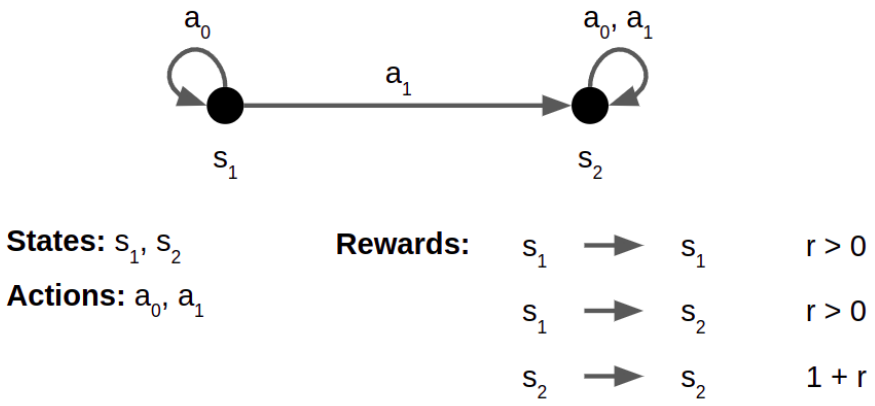

Figure 1: Simple two-state MDP

```
class TwoStateMDP(MDP):
    def __init__(self):
        self.states = ["s1", "s2"]
        self.actions = ["a0", "a1"]
        self.gam = 0.9

        # call the parent class
        # notice that the state is a list of state variables
        super().__init__(
            states=[self.states], actions=self.actions)
        self.populate_data()
```

```
    def populate_data(self):
        # add all routes from s1
        self.add_route(["s1"],"a0",["s1"])
        self.add_route(["s1"],"a1",["s2"])
        # add all routes from s2
        self.add_route(["s2"],"a0",["s2"])
        self.add_route(["s2"],"a1",["s2"])

        # let's populate the reward, assuming r>0 is 0.5
        for a in self.a:
            self.add_reward(["s1"],a,0.5)
            self.add_reward(["s2"],a,1.5)
```

Let's test some internal functions of class MDP:

```
twoStateMDP = TwoStateMDP()
print(twoStateMDP.get_index(["s1"]))
print(twoStateMDP.get_state(0))
print(twoStateMDP.get_real_state_value(0))

# Output
>>> 0
>>> [0]
>>> ['s1']
```