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

def printPrettyMatrix(objStr, matrix):
    objStr+="\t\t["
    for row in matrix:
        objStr+="["
        for col in row:
             objStr+=f" {col:.02f},"
        objStr= objStr[:-1]+"],"
        objStr+="\n \t\t"
    objStr=objStr[:-5]+"]"
    return objStr

def showPlot(PlotSS, PlotII, PlotRR, node, ts):
    plt.title(f"Node {node} Infctd:{np.max(PlotII)}")
    S_i = plt.plot(PlotSS[node,:], label=f'S_{node}({ts})') # just plot node 0's
susceptible over time
    I_i = plt.plot(PlotII[node,:], label=f'I_{node}({ts})') # just plot node 0's
infection over time
    R_i = plt.plot(PlotRR[node,:], label=f'R_{node}({ts})') # just plot node 0's
recovery over time
    plt.legend()
    plt.show()

def SIR(beta, gamma, A, I0, timesteps, show_plot=False, node=1) :
    '''
    Notes:
      - assumes that everything is dimensioned correctly
      - all S/I/R are fractions of node's population
      - R0 = 0
      - S0 = 1-I0
    '''
    n = A.shape[0]

    # initialize:
    SS = np.zeros([n,timesteps]) # each column in a list of S_i's; there is one
column per timestep
    II = np.zeros_like(SS) # copy the shape of SS to create II
    RR = np.zeros_like(SS) # nobody starts out with immunity
    II[:,0] = I0 # set the initial infection
    SS[:,0] = np.ones(n)-II[:,0] # if you're not infected, you're susceptible

    for t in range(1,timesteps):
        bigS = np.diag(SS[:,t-1])
        new_infections = beta * bigS @ A @ II[:,t-1]
        heals = gamma*II[:,t-1]

        SS[:,t] = SS[:,t-1] - new_infections
        II[:,t] = II[:,t-1] + new_infections - heals
```

```
        RR[:,t] = RR[:,t-1] + heals

    if show_plot:
        showPlot(SS, II, RR, node, timesteps)

    return SS,II,RR

############################################################
# Homework 05: Question 1
############################################################
print("\nQuestion 1:")
beta = 0.8
gamma = 0.4
Q1_og_infection = [0.1, 0.0, 0.0, 0.0, 0.0]
Q1_graph_adj = np.array([[0.8, 0.0, 0.0, 0.05, 0.15],
                         [1.0, 0.0, 0.0, 0.0, 0.0],
                         [0.3, 0.2, 0.5, 0.0, 0.0],
                         [0.0, 0.0, 0.05, 0.95, 0.0],
                         [0.0, 0.0, 0.0, 0.2, 0.8]])
q1_timesteps = 1

Q1_SS, Q1_II, Q1_RR = SIR(beta, gamma, Q1_graph_adj, Q1_og_infection,
q1_timesteps+1, show_plot=False)
print("\tS_i(t):\t")
print(printPrettyMatrix("", Q1_SS))
print("\tI_i(t):\t")
print(printPrettyMatrix("",  Q1_II))
print("\tR_i(t):\t")
print(printPrettyMatrix("", Q1_RR))

############################################################
# Homework 05: Question 2
############################################################
print("\nQuestion 2:")
Q2_og_infection = [0.1, 0.0, 0.0, 0.0, 0.0]
Q2_graph_adj = np.array([[0.8, 0.0, 0.0, 0.05, 0.15],
                         [1.0, 0.0, 0.0, 0.0, 0.0],
                         [0.3, 0.2, 0.5, 0.0, 0.0],
                         [0.0, 0.0, 0.05, 0.95, 0.0],
                         [0.0, 0.0, 0.0, 0.2, 0.8]])
q2_timesteps = 0

while  True:
    Q2_SS, Q2_II, Q2_RR = SIR(beta, gamma, Q2_graph_adj, Q2_og_infection,
q2_timesteps+1, show_plot=False)
    if np.all(Q2_II[:, q2_timesteps] != 0.0):
        print(f"all nodes are infected at time step {q2_timesteps} \t{Q2_II[:,
q2_timesteps]}")
        break
```

```
    q2_timesteps += 1

print("\tS_i(t):\t")
print(printPrettyMatrix("", Q2_SS))
print("\tI_i(t):\t")
print(printPrettyMatrix("",  Q2_II))
print("\tR_i(t):\t")
print(printPrettyMatrix("", Q2_RR))


############################################################
# Homework 05: Question 3
############################################################
print("\nQuestion 3:")
print("\tSuppose that the connection between node 2 and node 1 is removed. If\n",
      "\tnothing else changes in the network, would this change the spread of\n",
      "\tthe epidemic? Can you predict how many people in node 2 would
eventually\n",
      "\tget sick?\n")
Q3_og_infection = [0.1, 0.0, 0.0, 0.0, 0.0]
Q3_graph_adj = np.array([[0.8, 0.0, 0.0, 0.05, 0.15],
                         [0.0, 0.0, 0.0, 0.0, 0.0],
                         [0.3, 0.2, 0.5, 0.0, 0.0],
                         [0.0, 0.0, 0.05, 0.95, 0.0],
                         [0.0, 0.0, 0.0, 0.2, 0.8]])
q3_timesteps = 0

while  True:
    Q3_SS, Q3_II, Q3_RR = SIR(beta, gamma, Q3_graph_adj, Q3_og_infection,
q3_timesteps+1, node=2, show_plot=False)
    if (Q3_II[:, q3_timesteps][1] != 0.0):
        print(f"Node 2 is infected are infected at time step {q3_timesteps}
\t{Q3_II[:, q3_timesteps]}")
        break
    if q3_timesteps % 5 == 0:
        print(f"Step:\t{q3_timesteps}")
        print(f"Node 2 infection %:{Q3_II[:, q3_timesteps][1]}")
        print(F"The rest of the network {Q3_II[:, q3_timesteps]}")
        # showPlot(Q3_SS, Q3_II, Q3_RR, 2, q3_timesteps)
    if q3_timesteps > 10:
        break
    q3_timesteps += 1

print(f"Last ten time steps of {q3_timesteps-1}")
print("\tS_i(t):\t")
print(printPrettyMatrix("",Q3_SS[:, q3_timesteps-10:q3_timesteps]))
print("\tI_i(t):\t")
print(printPrettyMatrix("",  Q3_II[:, q3_timesteps-10:q3_timesteps]))
print("\tR_i(t):\t")
print(printPrettyMatrix("", Q3_RR[:, q3_timesteps-10:q3_timesteps]))
```

```
############################################################
# Homework 05: Question 4
############################################################
print("\nQuestion 4:")
print("\tRepeat that same thought experiment, but this time let the initial\n",
      "\tinfection start on node 2 (so S_i(0)=0.1 and S_2(0)=0.99, but all other\n",
      "\tnodes have S_i(0)=1, I_i(0)=0 and R_i(0)=0.  Now can you predict how
many\n",
      "\tpeople in node 2 would eventually get sick?\n")
Q4_og_infection = [0.0, 0.01, 0.0, 0.0, 0.0]
Q4_graph_adj = np.array([[0.8, 0.0, 0.0, 0.05, 0.15],
                         [0.0, 0.0, 0.0, 0.0, 0.0],
                         [0.3, 0.2, 0.5, 0.0, 0.0],
                         [0.0, 0.0, 0.05, 0.95, 0.0],
                         [0.0, 0.0, 0.0, 0.2, 0.8]])
q4_timesteps = 0

while True:
    Q4_SS, Q4_II, Q4_RR = SIR(beta, gamma, Q4_graph_adj, Q4_og_infection,
q4_timesteps+1, node = 2, show_plot=False)
    if q4_timesteps % 25 == 0:
        print(f"Step:\t{q4_timesteps}")
        print(f"Node 2 infection %:{Q4_II[:, q4_timesteps][1]}")
        print(F"The rest of the network {Q4_II[:, q4_timesteps]}")
        showPlot(Q4_SS, Q4_II, Q4_RR, 2, q4_timesteps)
    if q4_timesteps > 50:
        break
    q4_timesteps += 1

print(f"Last ten time steps of {q4_timesteps-1}")
print("\tS_i(t):\t")
print(printPrettyMatrix("",Q4_SS[:, q4_timesteps-10:q4_timesteps]))
print("\tI_i(t):\t")
print(printPrettyMatrix("",  Q4_II[:, q4_timesteps-10:q4_timesteps]))
print("\tR_i(t):\t")
print(printPrettyMatrix("", Q4_RR[:, q4_timesteps-10:q4_timesteps]))


############################################################
# Homework 05: Question 5
############################################################
print("\nQuestion 5:")
print("\tInstead, suppose you remove the connection from node 2 to node 1\n",
      "\tand replace it with a self-loop on 2 with weight 1. If nothing else\n",
      "\tchanges in the network (and the infection starts at node 1), would \n",
      "\tthis change the spread of the epidemic? Can you predict how many\n",
      "\tpeople in node 2 would eventually get sick?\n")
Q5_og_infection = [0.0, 0.1, 0.0, 0.0, 0.0]
```

```
Q5_graph_adj = np.array([[0.8, 0.0, 0.0, 0.05, 0.15],
                         [0.0, 1.0, 0.0, 0.0, 0.0],
                         [0.3, 0.2, 0.5, 0.0, 0.0],
                         [0.0, 0.0, 0.05, 0.95, 0.0],
                         [0.0, 0.0, 0.0, 0.2, 0.8]])

q5_timesteps = 0

while True:
    Q5_SS, Q5_II, Q5_RR = SIR(beta, gamma, Q5_graph_adj, Q5_og_infection,
q5_timesteps+1, node = 2, show_plot=False)
    if q5_timesteps % 25 == 0:
        print(f"Step:\t{q5_timesteps}")
        print(f"Node 2 infection %:{Q5_II[:, q5_timesteps][1]}")
        print(F"The rest of the network {Q5_II[:, q5_timesteps]}")
        showPlot(Q5_SS, Q5_II, Q5_RR, 2, q5_timesteps)

    if q5_timesteps > 50:
        break
    q5_timesteps += 1

print(f"Last ten time steps of {q5_timesteps-1}")
print("\tS_i(t):\t")
print(printPrettyMatrix("",Q5_SS[:, q5_timesteps-10:q5_timesteps]))
print("\tI_i(t):\t")
print(printPrettyMatrix("",  Q5_II[:, q5_timesteps-10:q5_timesteps]))
print("\tR_i(t):\t")
print(printPrettyMatrix("", Q5_RR[:, q5_timesteps-10:q5_timesteps]))
```