Exercise 11.1

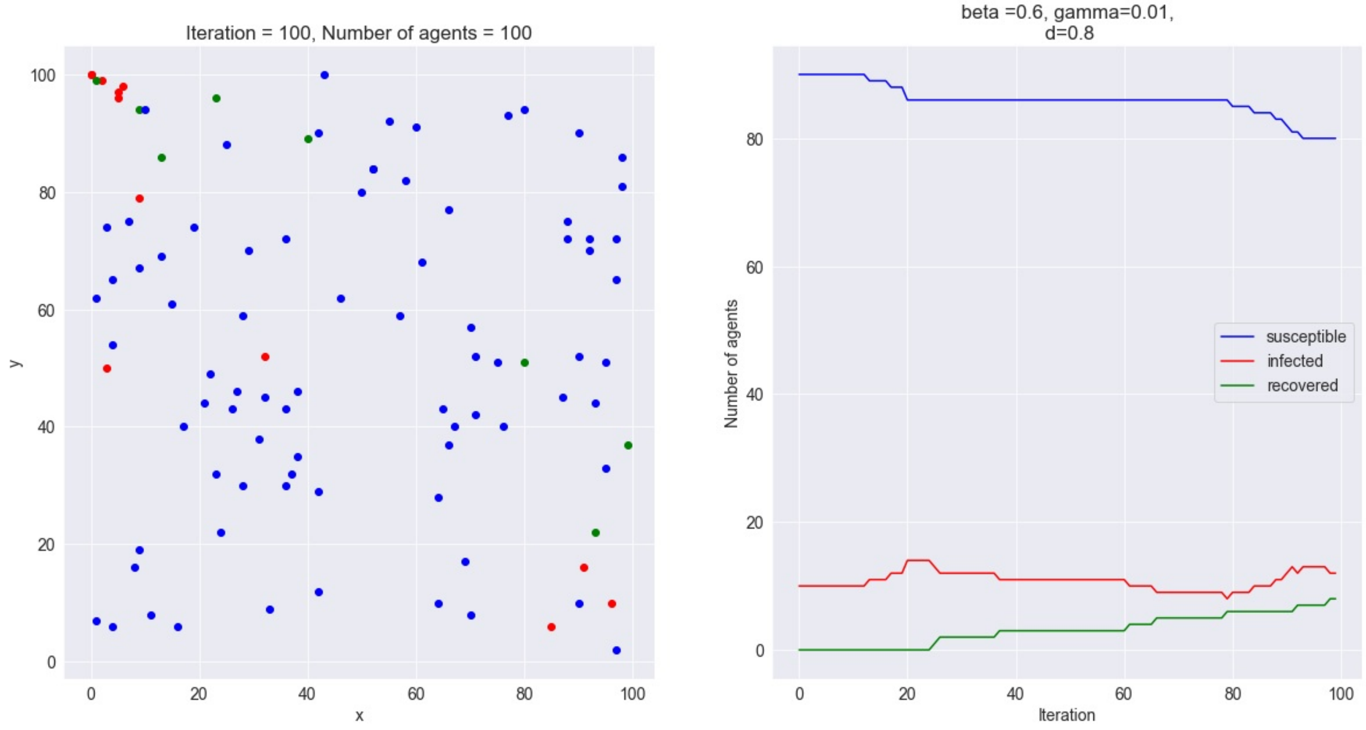
In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pylab
import seaborn as sns
import random
import math
```

In [2]:

```
no_of_individuals = 100
no_of_iterations = 100
diffusion_rate = 0.8   # diffusion probability
initial_infection_rate = 0.1
beta = 0.6   #Infection Probability
gamma = 0.01  # Recovery Probability
grid_x = 100  # max coordinate in x
grid_y = 100  #max coordinate in y

# status 1 = susceptible
# status 2 = infected
# status 3 = immune (recovered)

individual_position = []
individual_status = []
final_status = np.zeros((3, no_of_iterations))


for i in range(no_of_individuals):
    individual_position.append([random.randint(0,grid_x), random.randint(0,grid_y)])
    individual_status.append(1)

initially_infected_individuals = random.sample(range(no_of_individuals), round(no_of_individuals*initial_infectic
for j in initially_infected_individuals:
    individual_status[j] = 2

for m in range(no_of_iterations):
    for n in range(no_of_individuals):
        rand1 = np.random.rand()
        if rand1 < diffusion_rate:
            motion = np.random.choice([1, 2, 3, 4])
            if motion == 1:
                if individual_position[n][0] < grid_x:
                    individual_position[n][0] = individual_position[n][0] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 2:
                if individual_position[n][0] > 0:
                    individual_position[n][0] = individual_position[n][0] - 1
                else:
                    individual_position[n][0] = grid_x

            elif motion == 3:
                if individual_position[n][1] < grid_y:
                    individual_position[n][1] = individual_position[n][1] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 4:
                if individual_position[n][1] > 0:
                    individual_position[n][1] = individual_position[n][1] - 1
                else:
                    individual_position[n][0] = grid_y

    for i in range(no_of_individuals):
        if individual_status[i] == 2:
            infected_individual_position = individual_position[i]
            rand2 = np.random.rand()
            if rand2 < beta:
                for j in range(no_of_individuals):
                    if individual_position[j] == infected_individual_position and individual_status[j] == 1:
                        individual_status[j] = 2
            rand3 = np.random.rand()
            if rand3 < gamma:
                individual_status[i] = 3


    for i in range(no_of_individuals):
        if individual_status[i] == 1:
            final_status[0][m] = final_status[0][m] + 1
        if individual_status[i] == 2:
            final_status[1][m] = final_status[1][m] + 1
        if individual_status[i] == 3:
            final_status[2][m] = final_status[2][m] + 1



sns.set_style("darkgrid")
fig = plt.figure(figsize=(20,10))
fig.tight_layout()
plt.rcParams.update({'font.size': 14})

ax1 = fig.add_subplot(121)
ax1.clear()
# ax1.set_ylim(0, 10)
# ax1.set_xlim(0, 10)
for i in range(no_of_individuals):
    if individual_status[i] == 1:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "b")
    elif individual_status[i] == 2:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "r")
    elif individual_status[i] == 3:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "g")
ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title(f'Iteration = {no_of_iterations}, Number of agents = {no_of_individuals}')

ax1 = fig.add_subplot(122)
ax1.plot(range(no_of_iterations), final_status[0][:], "b", label = "susceptible")
ax1.plot(range(no_of_iterations), final_status[1][:], "r", label = "infected")
ax1.plot(range(no_of_iterations), final_status[2][:], "g", label = "recovered")
ax1.set_xlabel('Iteration')
ax1.set_ylabel('Number of agents')
plt.legend(ncol = 1, loc = "center right")
ax1.set_title(f" beta ={beta}, gamma={gamma}, \n d={diffusion_rate}")

fig.show()
```

```
<ipython-input-2-e54fcf30fc37>:108: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_i
nline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



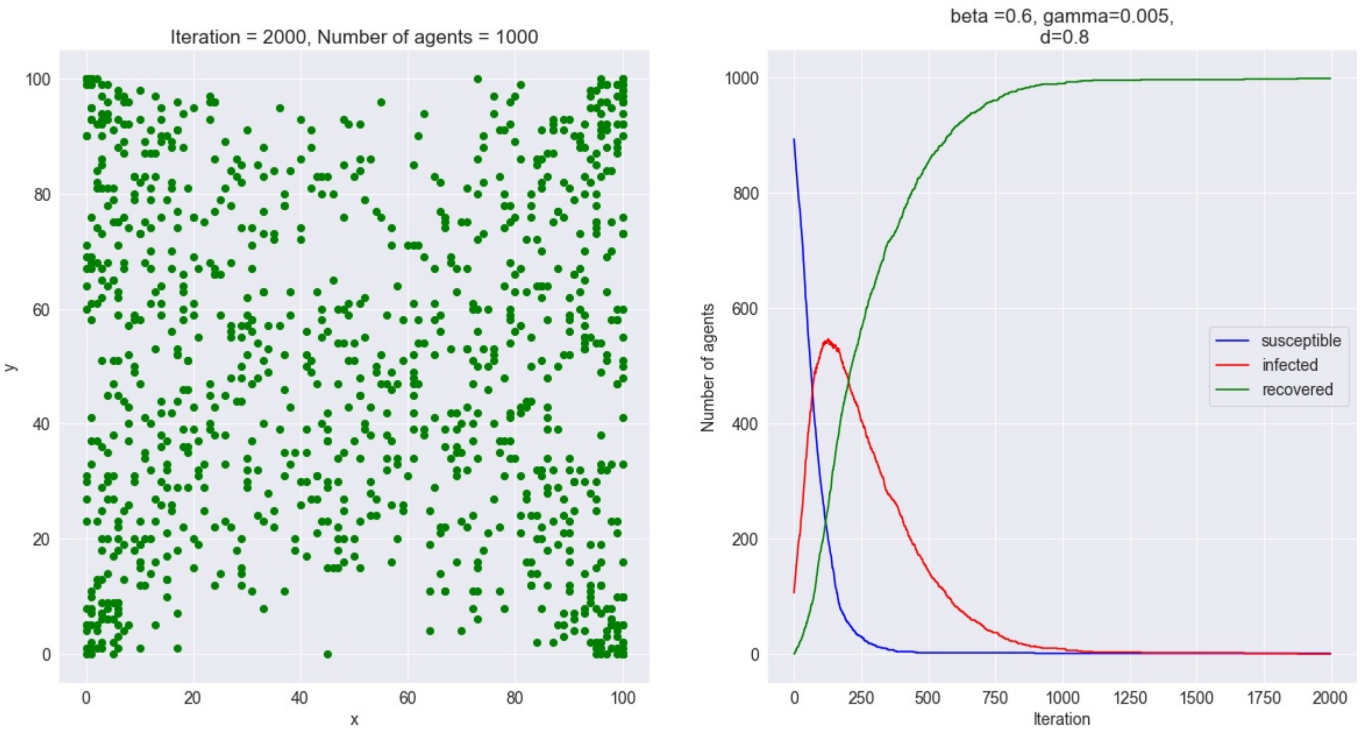
In [3]:

```
no_of_individuals = 1000
no_of_iterations = 2000
diffusion_rate = 0.8   # diffusion probability
initial_infection_rate = 0.1
beta = 0.6   #Infection Probability
gamma = 0.005  # Recovery Probability
grid_x = 100  # max coordinate in x
grid_y = 100  #max coordinate in y

# status 1 = susceptible
# status 2 = infected
# status 3 = immune (recovered)

individual_position = []
```

```
individual_status = []
final_status = np.zeros((3, no_of_iterations))


for i in range(no_of_individuals):
    individual_position.append([random.randint(0,grid_x), random.randint(0,grid_y)])
    individual_status.append(1)

initially_infected_individuals = random.sample(range(no_of_individuals), round(no_of_individuals*initial_infectio
for j in initially_infected_individuals:
    individual_status[j] = 2

for m in range(no_of_iterations):
    for n in range(no_of_individuals):
        rand1 = np.random.rand()
        if rand1 < diffusion_rate:
            motion = np.random.choice([1, 2, 3, 4])
            if motion == 1:
                if individual_position[n][0] < grid_x:
                    individual_position[n][0] = individual_position[n][0] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 2:
                if individual_position[n][0] > 0:
                    individual_position[n][0] = individual_position[n][0] - 1
                else:
                    individual_position[n][0] = grid_x

            elif motion == 3:
                if individual_position[n][1] < grid_y:
                    individual_position[n][1] = individual_position[n][1] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 4:
                if individual_position[n][1] > 0:
                    individual_position[n][1] = individual_position[n][1] - 1
                else:
                    individual_position[n][0] = grid_y

    for i in range(no_of_individuals):
        if individual_status[i] == 2:
            infected_individual_position = individual_position[i]
            rand2 = np.random.rand()
            if rand2 < beta:
                for j in range(no_of_individuals):
                    if individual_position[j] == infected_individual_position and individual_status[j] == 1:
                        individual_status[j] = 2
            rand3 = np.random.rand()
            if rand3 < gamma:
                individual_status[i] = 3


    for i in range(no_of_individuals):
        if individual_status[i] == 1:
            final_status[0][m] = final_status[0][m] + 1
        if individual_status[i] == 2:
            final_status[1][m] = final_status[1][m] + 1
        if individual_status[i] == 3:
            final_status[2][m] = final_status[2][m] + 1



sns.set_style("darkgrid")
fig = plt.figure(figsize=(20,10))
fig.tight_layout()
plt.rcParams.update({'font.size': 14})

ax1 = fig.add_subplot(121)
ax1.clear()
# ax1.set_ylim(0, 10)
# ax1.set_xlim(0, 10)
for i in range(no_of_individuals):
    if individual_status[i] == 1:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "b")
    elif individual_status[i] == 2:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "r")
    elif individual_status[i] == 3:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "g")
ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title(f'Iteration = {no_of_iterations}, Number of agents = {no_of_individuals}')

ax1 = fig.add_subplot(122)
ax1.plot(range(no_of_iterations), final_status[0][:], "b", label = "susceptible")
ax1.plot(range(no_of_iterations), final_status[1][:], "r", label = "infected")
ax1.plot(range(no_of_iterations), final_status[2][:], "g", label = "recovered")
ax1.set_xlabel('Iteration')
```

```
ax1.set_ylabel('Number of agents')
plt.legend(ncol = 1, loc = "center right")
ax1.set_title(f" beta ={beta}, gamma={gamma}, \n d={diffusion_rate}")

fig.show()
```

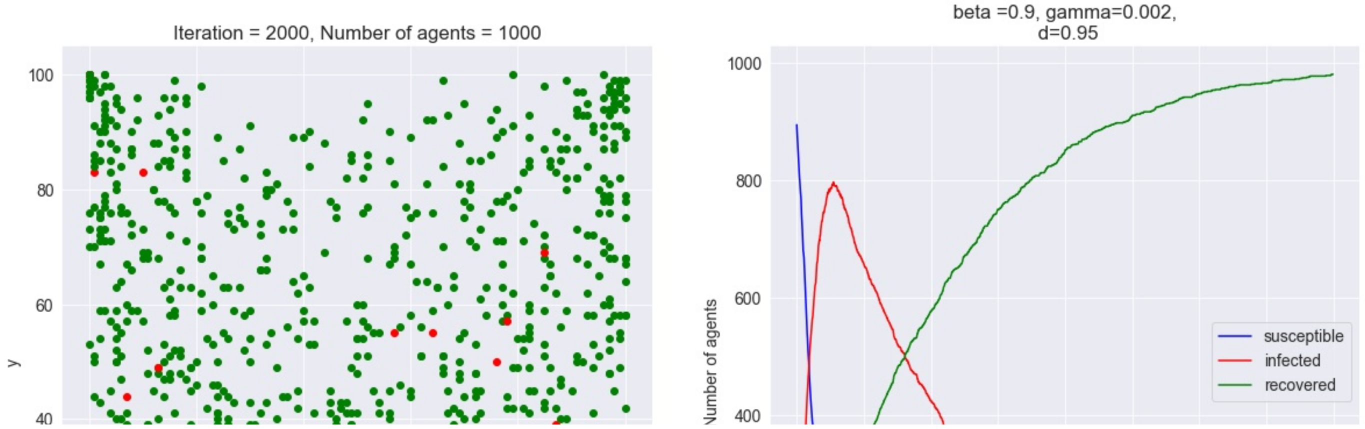
```
<ipython-input-3-9476ed2905c1>:108: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_i
nline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```

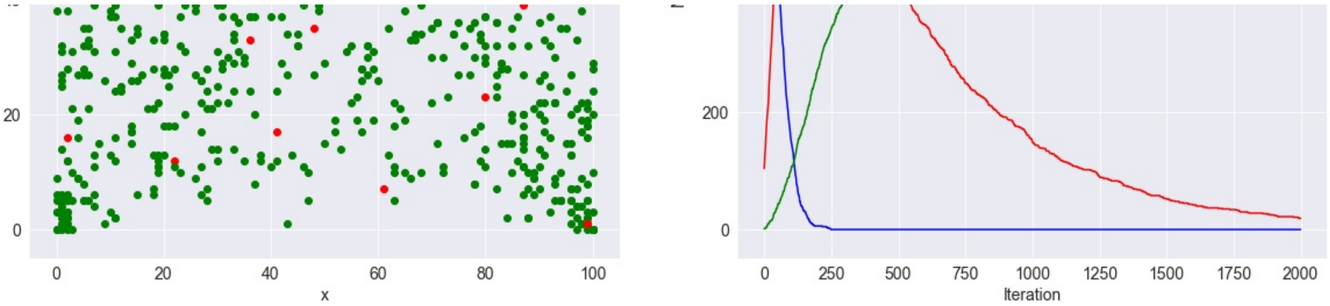


In [4]:

```
no_of_individuals = 1000
no_of_iterations = 2000
diffusion_rate = 0.95   # diffusion probability
initial_infection_rate = 0.1
beta = 0.9   #Infection Probability
gamma = 0.002  # Recovery Probability
grid_x = 100  # max coordinate in x
grid_y = 100  #max coordinate in y

# status 1 = susceptible
# status 2 = infected
# status 3 = immune (recovered)

individual_position = []
individual_status = []
final_status = np.zeros((3, no_of_iterations))


for i in range(no_of_individuals):
    individual_position.append([random.randint(0,grid_x), random.randint(0,grid_y)])
    individual_status.append(1)

initially_infected_individuals = random.sample(range(no_of_individuals), round(no_of_individuals*initial_infectio
for j in initially_infected_individuals:
    individual_status[j] = 2

for m in range(no_of_iterations):
    for n in range(no_of_individuals):
        rand1 = np.random.rand()
        if rand1 < diffusion_rate:
            motion = np.random.choice([1, 2, 3, 4])
            if motion == 1:
                if individual_position[n][0] < grid_x:
                    individual_position[n][0] = individual_position[n][0] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 2:
                if individual_position[n][0] > 0:
                    individual_position[n][0] = individual_position[n][0] - 1
                else:
                    individual_position[n][0] = grid_x
            elif motion == 3:
                if individual_position[n][1] < grid_y:
                    individual_position[n][1] = individual_position[n][1] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 4:
                if individual_position[n][1] > 0:
                    individual_position[n][1] = individual_position[n][1] - 1
                else:
                    individual_position[n][0] = grid_y

    for i in range(no_of_individuals):
        if individual_status[i] == 2:
            infected_individual_position = individual_position[i]
            rand2 = np.random.rand()
            if rand2 < beta:
                for j in range(no_of_individuals):
                    if individual_position[j] == infected_individual_position and individual_status[j] == 1:
                        individual_status[j] = 2
            rand3 = np.random.rand()
            if rand3 < gamma:
                individual_status[i] = 3


    for i in range(no_of_individuals):
        if individual_status[i] == 1:
            final_status[0][m] = final_status[0][m] + 1
        if individual_status[i] == 2:
            final_status[1][m] = final_status[1][m] + 1
        if individual_status[i] == 3:
            final_status[2][m] = final_status[2][m] + 1



sns.set_style("darkgrid")
fig = plt.figure(figsize=(20,10))
fig.tight_layout()
plt.rcParams.update({'font.size': 14})

ax1 = fig.add_subplot(121)
ax1.clear()
# ax1.set_ylim(0, 10)
# ax1.set_xlim(0, 10)
for i in range(no_of_individuals):
    if individual_status[i] == 1:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "b")
    elif individual_status[i] == 2:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "r")
    elif individual_status[i] == 3:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "g")
ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title(f'Iteration = {no_of_iterations}, Number of agents = {no_of_individuals}')

ax1 = fig.add_subplot(122)
ax1.plot(range(no_of_iterations), final_status[0][:], "b", label = "susceptible")
ax1.plot(range(no_of_iterations), final_status[1][:], "r", label = "infected")
ax1.plot(range(no_of_iterations), final_status[2][:], "g", label = "recovered")
ax1.set_xlabel('Iteration')
ax1.set_ylabel('Number of agents')
plt.legend(ncol = 1, loc = "center right")
ax1.set_title(f" beta ={beta}, gamma={gamma}, \n d={diffusion_rate}")

fig.show()
```

```
<ipython-input-4-5e10c0a5e74d>:108: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_i
nline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```

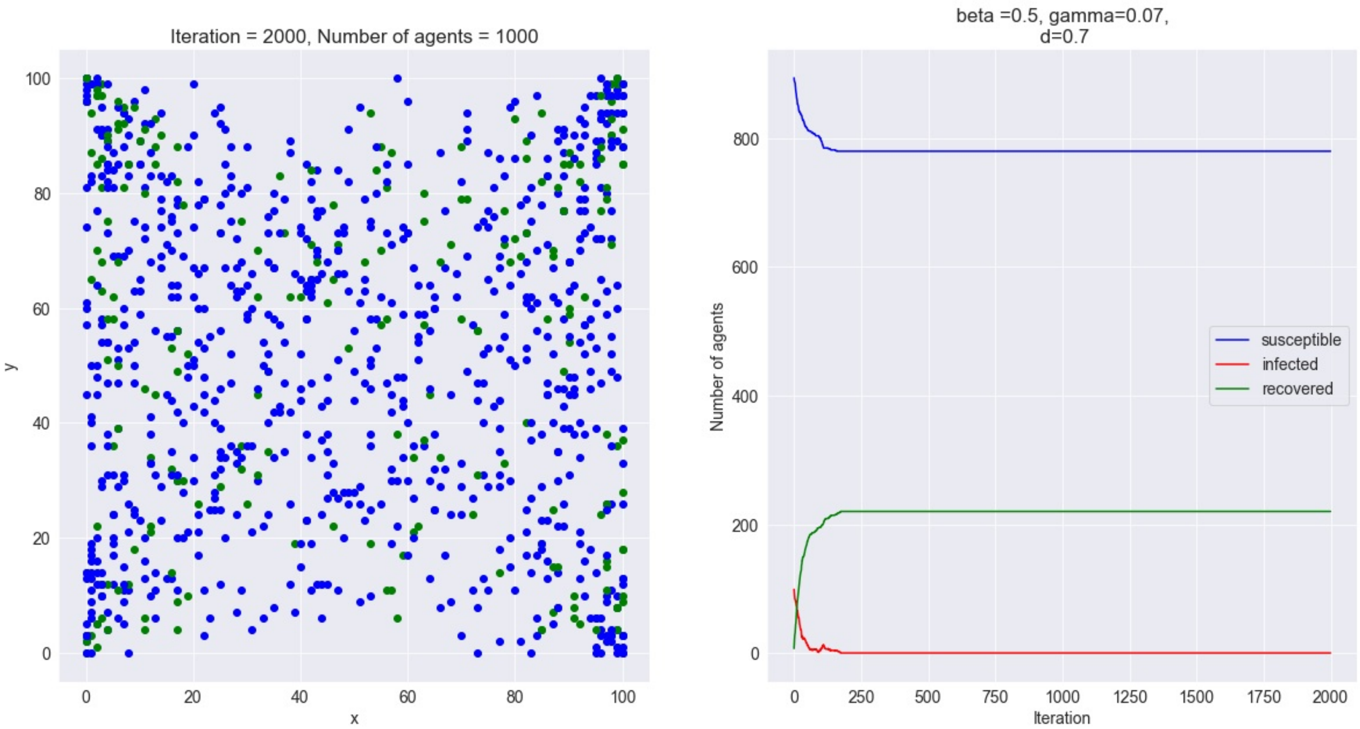




In [5]:

```
no_of_individuals = 1000
no_of_iterations = 2000
diffusion_rate = 0.7   # diffusion probability
initial_infection_rate = 0.1
beta = 0.5   #Infection Probability
gamma = 0.07  # Recovery Probability
grid_x = 100  # max coordinate in x
grid_y = 100  #max coordinate in y

# status 1 = susceptible
# status 2 = infected
# status 3 = immune (recovered)

individual_position = []
individual_status = []
final_status = np.zeros((3, no_of_iterations))


for i in range(no_of_individuals):
    individual_position.append([random.randint(0,grid_x), random.randint(0,grid_y)])
    individual_status.append(1)

initially_infected_individuals = random.sample(range(no_of_individuals), round(no_of_individuals*initial_infectio
for j in initially_infected_individuals:
    individual_status[j] = 2

for m in range(no_of_iterations):
    for n in range(no_of_individuals):
        rand1 = np.random.rand()
        if rand1 < diffusion_rate:
            motion = np.random.choice([1, 2, 3, 4])
            if motion == 1:
                if individual_position[n][0] < grid_x:
                    individual_position[n][0] = individual_position[n][0] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 2:
                if individual_position[n][0] > 0:
                    individual_position[n][0] = individual_position[n][0] - 1
                else:
                    individual_position[n][0] = grid_x

            elif motion == 3:
                if individual_position[n][1] < grid_y:
                    individual_position[n][1] = individual_position[n][1] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 4:
                if individual_position[n][1] > 0:
                    individual_position[n][1] = individual_position[n][1] - 1
                else:
                    individual_position[n][0] = grid_y

    for i in range(no_of_individuals):
        if individual_status[i] == 2:
            infected_individual_position = individual_position[i]
            rand2 = np.random.rand()
            if rand2 < beta:
                for j in range(no_of_individuals):
                    if individual_position[j] == infected_individual_position and individual_status[j] == 1:
                        individual_status[j] = 2
            rand3 = np.random.rand()
            if rand3 < gamma:
                individual_status[i] = 3


    for i in range(no_of_individuals):
        if individual_status[i] == 1:
            final_status[0][m] = final_status[0][m] + 1
        if individual_status[i] == 2:
            final_status[1][m] = final_status[1][m] + 1
        if individual_status[i] == 3:
            final_status[2][m] = final_status[2][m] + 1



sns.set_style("darkgrid")
fig = plt.figure(figsize=(20,10))
fig.tight_layout()
plt.rcParams.update({'font.size': 14})

ax1 = fig.add_subplot(121)
ax1.clear()
# ax1.set_ylim(0, 10)
# ax1.set_xlim(0, 10)
for i in range(no_of_individuals):
    if individual_status[i] == 1:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "b")
    elif individual_status[i] == 2:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "r")
    elif individual_status[i] == 3:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "g")
ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title(f'Iteration = {no_of_iterations}, Number of agents = {no_of_individuals}')

ax1 = fig.add_subplot(122)
ax1.plot(range(no_of_iterations), final_status[0][:], "b", label = "susceptible")
ax1.plot(range(no_of_iterations), final_status[1][:], "r", label = "infected")
ax1.plot(range(no_of_iterations), final_status[2][:], "g", label = "recovered")
ax1.set_xlabel('Iteration')
ax1.set_ylabel('Number of agents')
plt.legend(ncol = 1, loc = "center right")
ax1.set_title(f" beta ={beta}, gamma={gamma}, \n d={diffusion_rate}")

fig.show()
```

```
<ipython-input-5-af2bfc2bd229>:108: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_i
nline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```

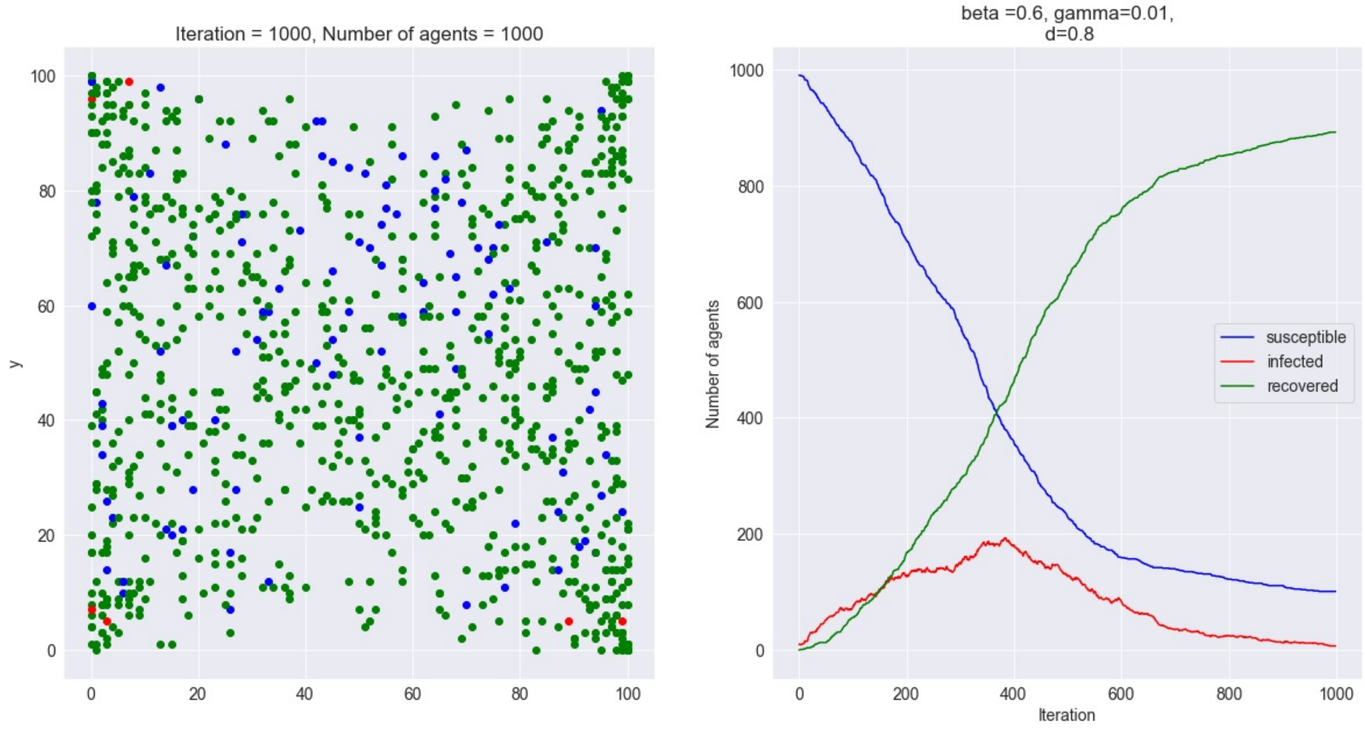


In [6]:

```
no_of_individuals = 1000
no_of_iterations = 1000
diffusion_rate = 0.8   # diffusion probability
initial_infection_rate = 0.01
beta = 0.6   #Infection Probability
gamma = 0.01  # Recovery Probability
grid_x = 100  # max coordinate in x
grid_y = 100  #max coordinate in y

# status 1 = susceptible
# status 2 = infected
# status 3 = immune (recovered)

individual_position = []
individual_status = []
final_status = np.zeros((3, no_of_iterations))


for i in range(no_of_individuals):
    individual_position.append([random.randint(0,grid_x), random.randint(0,grid_y)])
    individual_status.append(1)

initially_infected_individuals = random.sample(range(no_of_individuals), round(no_of_individuals*initial_infectio
for j in initially_infected_individuals:
    individual_status[j] = 2

for m in range(no_of_iterations):
    for n in range(no_of_individuals):
        rand1 = np.random.rand()
        if rand1 < diffusion_rate:
            motion = np.random.choice([1, 2, 3, 4])
            if motion == 1:
                if individual_position[n][0] < grid_x:
                    individual_position[n][0] = individual_position[n][0] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 2:
                if individual_position[n][0] > 0:
                    individual_position[n][0] = individual_position[n][0] - 1
                else:
                    individual_position[n][0] = grid_x

            elif motion == 3:
                if individual_position[n][1] < grid_y:
                    individual_position[n][1] = individual_position[n][1] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 4:
                if individual_position[n][1] > 0:
                    individual_position[n][1] = individual_position[n][1] - 1
                else:
                    individual_position[n][0] = grid_y

    for i in range(no_of_individuals):
        if individual_status[i] == 2:
            infected_individual_position = individual_position[i]
            rand2 = np.random.rand()
            if rand2 < beta:
                for j in range(no_of_individuals):
                    if individual_position[j] == infected_individual_position and individual_status[j] == 1:
                        individual_status[j] = 2
            rand3 = np.random.rand()
            if rand3 < gamma:
                individual_status[i] = 3


    for i in range(no_of_individuals):
        if individual_status[i] == 1:
            final_status[0][m] = final_status[0][m] + 1
        if individual_status[i] == 2:
            final_status[1][m] = final_status[1][m] + 1
        if individual_status[i] == 3:
            final_status[2][m] = final_status[2][m] + 1



sns.set_style("darkgrid")
fig = plt.figure(figsize=(20,10))
fig.tight_layout()
plt.rcParams.update({'font.size': 14})

ax1 = fig.add_subplot(121)
ax1.clear()
# ax1.set_ylim(0, 10)
# ax1.set_xlim(0, 10)
for i in range(no_of_individuals):
    if individual_status[i] == 1:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "b")
    elif individual_status[i] == 2:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "r")
    elif individual_status[i] == 3:
        plt.scatter(individual_position[i][0],individual_position[i][1], color = "g")
# ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title(f'Iteration = {no_of_iterations}, Number of agents = {no_of_individuals}')

ax1 = fig.add_subplot(122)
```

```
ax1.plot(range(no_of_iterations), final_status[0][:], "b", label = "susceptible")
ax1.plot(range(no_of_iterations), final_status[1][:], "r", label = "infected")
ax1.plot(range(no_of_iterations), final_status[2][:], "g", label = "recovered")
ax1.set_xlabel('Iteration')
ax1.set_ylabel('Number of agents')
plt.legend(ncol = 1, loc = "center right")
ax1.set_title(f" beta ={beta}, gamma={gamma}, \n d={diffusion_rate}")

fig.show()
```

```
<ipython-input-6-8f74b89cf45f>:108: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_i
nline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



Exercise 11.2 (part -a)

In [8]:

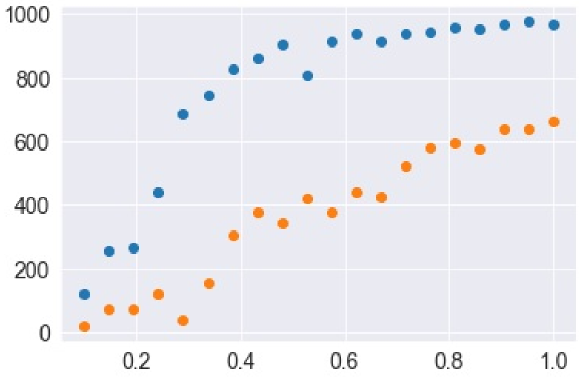
```
# for gamma = 0.01
import numpy as np
import random
import matplotlib.pyplot as plt

no_of_individuals = 1000
no_of_iterations = 1000
diffusion_rate = 0.8   # diffusion probability
initial_infection_rate = 0.01
beta = np.linspace(0.1, 1, 20)   #Infection Probability
gamma = 0.01  # Recovery Probability
grid_x = 100  # max coordinate in x
grid_y = 100  #max coordinate in y

# status 1 = susceptible
# status 2 = infected
# status 3 = immune (recovered)
repetition = 3
recovered_individuals_1 = []
for b in beta:
    total_recovered = 0
    for rep in range(repetition):
#         print(rep)
        individual_position = []
        individual_status = []

        for i in range(no_of_individuals):
            individual_position.append([random.randint(0,grid_x), random.randint(0,grid_y)])
            individual_status.append(1)

        initially_infected_individuals = random.sample(range(no_of_individuals), round(no_of_individuals*initial_
        for j in initially_infected_individuals:
            individual_status[j] = 2
        for m in range(no_of_iterations):
            for n in range(no_of_individuals):
                rand1 = np.random.rand()
                if rand1 < diffusion_rate:
                    motion = np.random.choice([1, 2, 3, 4])
                    if motion == 1:
                        if individual_position[n][0] < grid_x:
                            individual_position[n][0] = individual_position[n][0] + 1
                        else:
                            individual_position[n][0] = 0

                    elif motion == 2:
                        if individual_position[n][0] > 0:
                            individual_position[n][0] = individual_position[n][0] - 1
                        else:
                            individual_position[n][0] = grid_x

                    elif motion == 3:
                        if individual_position[n][1] < grid_y:
                            individual_position[n][1] = individual_position[n][1] + 1
                        else:
                            individual_position[n][0] = 0

                    elif motion == 4:
                        if individual_position[n][1] > 0:
                            individual_position[n][1] = individual_position[n][1] - 1
                        else:
                            individual_position[n][0] = grid_y

            for i in range(no_of_individuals):
                if individual_status[i] == 2:
                    infected_individual_position = individual_position[i]
                    rand2 = np.random.rand()
                    if rand2 < b:
                        for j in range(no_of_individuals):
                            if individual_position[j] == infected_individual_position and individual_status[j] ==
                                individual_status[j] = 2
                    rand3 = np.random.rand()
                    if rand3 < gamma:
                        individual_status[i] = 3

        recovered_individuals = 0
        for i in range(no_of_individuals):
            if individual_status[i] == 3:
                recovered_individuals = recovered_individuals + 1
#         print('recovered', recovered_individuals)
        total_recovered = total_recovered + recovered_individuals
#         print('total recovered', total_recovered)
    recovered_individuals_1.append(total_recovered/repetition)
#     print(recovered_individuals_1)
```

In [9]:

```
# for gamma = 0.02

no_of_individuals = 1000
no_of_iterations = 1000
diffusion_rate = 0.8   # diffusion probability
initial_infection_rate = 0.01
beta = np.linspace(0.1, 1, 20)   #Infection Probability
gamma = 0.02  # Recovery Probability
grid_x = 100  # max coordinate in x
grid_y = 100  #max coordinate in y
final_status = np.zeros((3, no_of_iterations))

# status 1 = susceptible
# status 2 = infected
# status 3 = immune (recovered)

repetition = 3
recovered_individuals_2 = []
for b in beta:
    total_recovered = 0
    for rep in range(repetition):
#         print(rep)
        individual_position = []
        individual_status = []

        for i in range(no_of_individuals):
            individual_position.append([random.randint(0,grid_x), random.randint(0,grid_y)])
            individual_status.append(1)

        initially_infected_individuals = random.sample(range(no_of_individuals), round(no_of_individuals*initial_
        for j in initially_infected_individuals:
            individual_status[j] = 2

        for m in range(no_of_iterations):
            for n in range(no_of_individuals):
                rand1 = np.random.rand()
                if rand1 < diffusion_rate:
                    motion = np.random.choice([1, 2, 3, 4])
                    if motion == 1:
                        if individual_position[n][0] < grid_x:
                            individual_position[n][0] = individual_position[n][0] + 1
                        else:
                            individual_position[n][0] = 0

                    elif motion == 2:
                        if individual_position[n][0] > 0:
                            individual_position[n][0] = individual_position[n][0] - 1
                        else:
                            individual_position[n][0] = grid_x

                    elif motion == 3:
                        if individual_position[n][1] < grid_y:
                            individual_position[n][1] = individual_position[n][1] + 1
                        else:
                            individual_position[n][0] = 0

                    elif motion == 4:
                        if individual_position[n][1] > 0:
                            individual_position[n][1] = individual_position[n][1] - 1
                        else:
                            individual_position[n][0] = grid_y

            for i in range(no_of_individuals):
                if individual_status[i] == 2:
                    infected_individual_position = individual_position[i]
                    rand2 = np.random.rand()
                    if rand2 < b:
                        for j in range(no_of_individuals):
                            if individual_position[j] == infected_individual_position and individual_status[j] ==
                                individual_status[j] = 2
                    rand3 = np.random.rand()
                    if rand3 < gamma:
                        individual_status[i] = 3

        recovered_individuals = 0
        for i in range(no_of_individuals):
            if individual_status[i] == 3:
                recovered_individuals = recovered_individuals + 1
#         print('recovered', recovered_individuals)
        total_recovered = total_recovered + recovered_individuals
#         print('total recovered', total_recovered)
    recovered_individuals_2.append(total_recovered/repetition)
#     print(recovered_individuals_1)
```

In [10]:

```
plt.scatter(beta, recovered_individuals_1)
plt.scatter(beta, recovered_individuals_2)
plt.show()
```

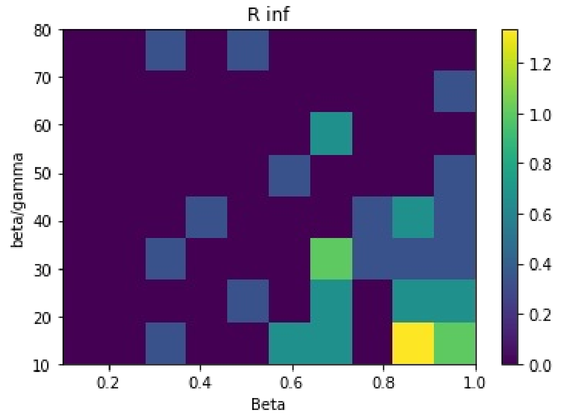


Exercise 11.2 (part -b)

In [3]:

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

no_of_individuals = 1000
no_of_iterations = 1000
diffusion_rate = 0.8   # diffusion probability
initial_infection_rate = 0.01
beta = np.linspace(0.1, 1, 10)    #Infection Probability
# gamma = 0.01  # Recovery Probability
grid_x = 100  # max coordinate in x
```

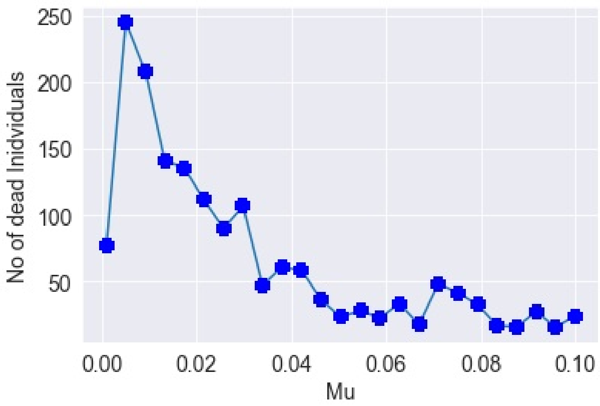
```
grid_y = 100  #max coordinate in y
repetition = 3
# status 1 = susceptible
# status 2 = infected
# status 3 = immune (recovered)

# final_status = np.zeros((3, no_of_iterations))
BG = [80,70,60,50,40,30,20,10]

beta_list = []
for bg in range(len(BG)):
    bglist = []
    for b in beta:
        total_recovered = 0
        for rep in range(repetition):
            gamma = 1/(BG[bg]/b)
            individual_position = []
            individual_status = []
            final_status = np.zeros((3, no_of_iterations))

            for i in range(no_of_individuals):
                individual_position.append([random.randint(0,grid_x), random.randint(0,grid_y)])
                individual_status.append(1)

            initially_infected_individuals = random.sample(range(no_of_individuals), round(no_of_individuals*init
            for j in initially_infected_individuals:
                individual_status[j] = 2

            for m in range(no_of_iterations):
                for n in range(no_of_individuals):
                    rand1 = np.random.rand()
                    if rand1 < diffusion_rate:
                        motion = np.random.choice([1, 2, 3, 4])
                        if motion == 1:
                            if individual_position[n][0] < grid_x:
                                individual_position[n][0] = individual_position[n][0] + 1
                        else:
                            individual_position[n][0] = 0

                    elif motion == 2:
                        if individual_position[n][0] > 0:
                            individual_position[n][0] = individual_position[n][0] - 1
                        else:
                            individual_position[n][0] = grid_x

                    elif motion == 3:
                        if individual_position[n][1] < grid_y:
                            individual_position[n][1] = individual_position[n][1] + 1
                        else:
                            individual_position[n][0] = 0

                    elif motion == 4:
                        if individual_position[n][1] > 0:
                            individual_position[n][1] = individual_position[n][1] - 1
                        else:
                            individual_position[n][0] = grid_y

            for i in range(no_of_individuals):
                if individual_status[i] == 2:
                    infected_individual_position = individual_position[i]
                    rand2 = np.random.rand()
                    if rand2 < b:
                        for j in range(no_of_individuals):
                            if individual_position[j] == infected_individual_position and individual_status[j] ==
                                individual_status[j] = 2
                    rand3 = np.random.rand()
                    if rand3 < gamma:
                        individual_status[i] = 3


            for i in range(no_of_individuals):
                if individual_status[i] == 1:
                    final_status[0][m] = final_status[0][m] + 1
                if individual_status[i] == 2:
                    final_status[1][m] = final_status[1][m] + 1
                if individual_status[i] == 3:
                    final_status[2][m] = final_status[2][m] + 1
            total_recovered = final_status[2][no_of_iterations-1]
        bglist.append(total_recovered/repetition)
    beta_list.append(bglist)


plt.imshow(beta_list,aspect='auto', extent = [min(beta), max(beta), min(BG), max(BG)])
plt.xlabel("Beta")
plt.ylabel("beta/gamma")
plt.title("R inf")
plt.colorbar()
plt.show()
```



Exercise 11.3

In [13]:

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


no_of_individuals = 1000
no_of_iterations = 1000
diffusion_rate = 0.8   # diffusion probability
initial_infection_rate = 0.01
beta = 0.6  #Infection Probability
gamma = 0.01  # Recovery Probability
grid_x = 100  # max coordinate in x
grid_y = 100  #max coordinate in y
Mu = np.linspace(0.001, 0.1, 25)
# status 1 = susceptible
# status 2 = infected
# status 3 = immune (recovered)
repetition = 3
no_of_dead = []


for d in range(len(Mu)):
    total_dead = 0
    for rep in range(repetition):
        mu = Mu[d]
#         print('mu', mu)
#         print('rep', rep)
        individual_position = []
        individual_status = []
#         StatusList = np.zeros((3,numberOfLoops))
        for i in range(no_of_individuals):
            individual_position.append([random.randint(0,grid_x), random.randint(0,grid_y)])
            individual_status.append(1)

        initially_infected_individuals = random.sample(range(no_of_individuals), round(no_of_individuals*initial_
        for j in initially_infected_individuals:
            individual_status[j] = 2

        for m in range(no_of_iterations):
            for n in range(no_of_individuals):
                rand1 = np.random.rand()
                if rand1 < diffusion_rate:
                    motion = np.random.choice([1, 2, 3, 4])
                    if motion == 1:
                        if individual_position[n][0] < grid_x:
                            individual_position[n][0] = individual_position[n][0] + 1
                        else:
                            individual_position[n][0] = 0

                    elif motion == 2:
                        if individual_position[n][0] > 0:
                            individual_position[n][0] = individual_position[n][0] - 1
                        else:
                            individual_position[n][0] = grid_x

                    elif motion == 3:
                        if individual_position[n][1] < grid_y:
                            individual_position[n][1] = individual_position[n][1] + 1
                        else:
                            individual_position[n][0] = 0

                    elif motion == 4:
                        if individual_position[n][1] > 0:
                            individual_position[n][1] = individual_position[n][1] - 1
                        else:
                            individual_position[n][0] = grid_y

            for i in range(no_of_individuals):
                if individual_status[i] == 2:
                    infected_individual_position = individual_position[i]
                    rand2 = np.random.rand()
                    if rand2 < beta:
                        for j in range(no_of_individuals):
                            if individual_position[j] == infected_individual_position and individual_status[j] ==
                                individual_status[j] = 2
                    rand3 = np.random.rand()
                    if rand3 < gamma:
                        individual_status[i] = 3
                    rand4 = np.random.rand()
                    if rand4 < mu:
                        individual_status[i] = 4


        dead_individuals = 0
        for i in range(no_of_individuals):
            if individual_status[i] == 4:
                dead_individuals = dead_individuals + 1
#         print('dead', dead_individuals)
        total_dead = total_dead + dead_individuals
#         print('total dead', total_dead)
    no_of_dead.append(total_dead/repetition)
#     print(no_of_dead)


plt.plot(Mu,no_of_dead,marker = "+",markeredgecolor = 'blue',markeredgewidth = 10)
plt.xlabel('Mu')
plt.ylabel('No of dead Inidviduals')
plt.show()
```

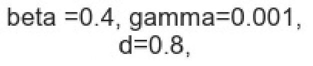


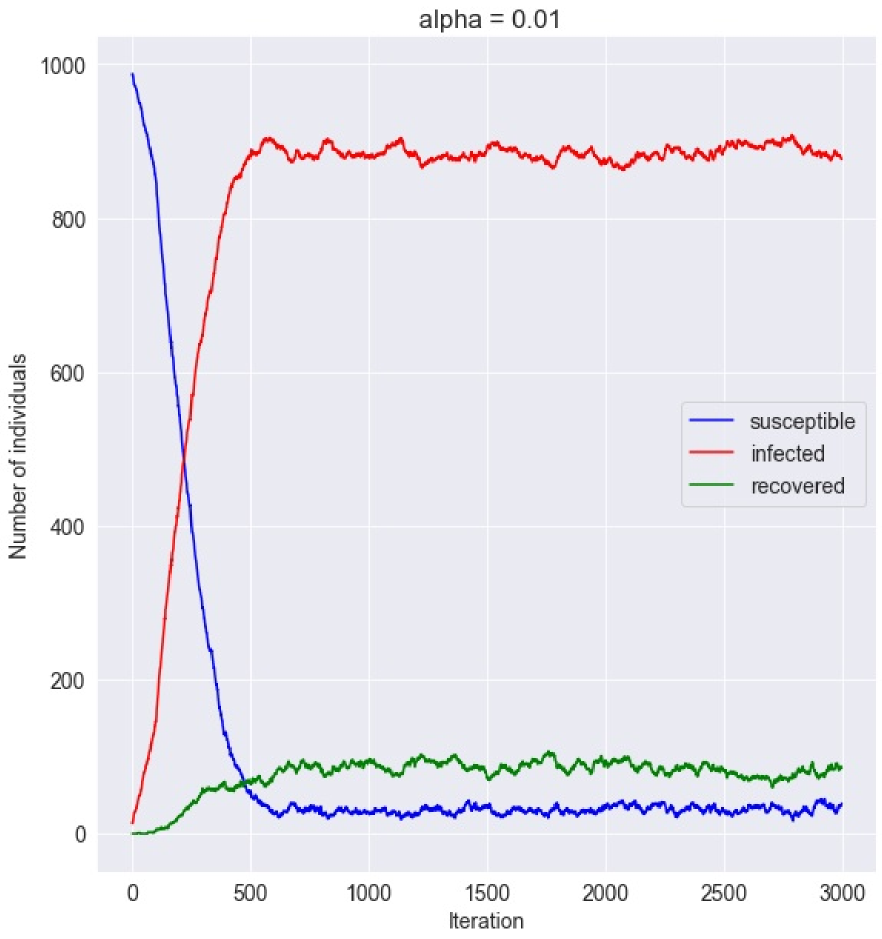
Exercise 11.4

In [14]:

```
import numpy as np
import matplotlib.pyplot as plt
import pylab
import seaborn as sns
import random
import math

no_of_individuals = 1000
no_of_iterations = 3000
diffusion_rate = 0.8   # diffusion probability
initial_infection_rate = 0.01
beta = 0.4   #Infection Probability
gamma = 0.001  # Recovery Probability
grid_x = 100  # max coordinate in x
grid_y = 100  #max coordinate in y
alpha = 0.01
# status 1 = susceptible
# status 2 = infected
# status 3 = immune (recovered)

individual_position = []
individual_status = []
final_status = np.zeros((3, no_of_iterations))


for i in range(no_of_individuals):
    individual_position.append([random.randint(0,grid_x), random.randint(0,grid_y)])
    individual_status.append(1)

initially_infected_individuals = random.sample(range(no_of_individuals), round(no_of_individuals*initial_infectic
for j in initially_infected_individuals:
    individual_status[j] = 2

for m in range(no_of_iterations):
    for n in range(no_of_individuals):
        rand1 = np.random.rand()
        if rand1 < diffusion_rate:
            motion = np.random.choice([1, 2, 3, 4])
            if motion == 1:
                if individual_position[n][0] < grid_x:
                    individual_position[n][0] = individual_position[n][0] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 2:
                if individual_position[n][0] > 0:
                    individual_position[n][0] = individual_position[n][0] - 1
                else:
                    individual_position[n][0] = grid_x

            elif motion == 3:
                if individual_position[n][1] < grid_y:
                    individual_position[n][1] = individual_position[n][1] + 1
                else:
                    individual_position[n][0] = 0

            elif motion == 4:
                if individual_position[n][1] > 0:
                    individual_position[n][1] = individual_position[n][1] - 1
                else:
                    individual_position[n][0] = grid_y

    for i in range(no_of_individuals):
        if individual_status[i] == 2:
            infected_individual_position = individual_position[i]
            rand2 = np.random.rand()
            if rand2 < beta:
                for j in range(no_of_individuals):
                    if individual_position[j] == infected_individual_position and individual_status[j] == 1:
                        individual_status[j] = 2
            rand3 = np.random.rand()
            if rand3 < gamma:
                individual_status[i] = 3
        if individual_status[i] == 3:
            rand4 = np.random.rand()
            if rand4 < alpha:
                individual_status[i] = 1



    for i in range(no_of_individuals):
        if individual_status[i] == 1:
            final_status[0][m] = final_status[0][m] + 1
        if individual_status[i] == 2:
            final_status[1][m] = final_status[1][m] + 1
        if individual_status[i] == 3:
            final_status[2][m] = final_status[2][m] + 1



# sns.set_style("darkgrid")
fig = plt.figure(figsize=(20,10))
fig.tight_layout()
plt.rcParams.update({'font.size': 14})

ax1 = fig.add_subplot(121)
ax1.clear()
ax1.plot(range(no_of_iterations), final_status[0][:], "b", label = "susceptible")
ax1.plot(range(no_of_iterations), final_status[1][:], "r", label = "infected")
ax1.plot(range(no_of_iterations), final_status[2][:], "g", label = "recovered")
ax1.set_xlabel('Iteration')
ax1.set_ylabel('Number of individuals')
plt.legend(ncol = 1, loc = "center right")
ax1.set_title(f" beta ={beta}, gamma={gamma}, \n d={diffusion_rate}, \n alpha = {alpha}")

fig.show()
```

```
<ipython-input-14-ed4290cf6112>:106: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_
inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```





```
In [ ]:
```

```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js