**agentframework:**

import random

random.seed(0)

n\_agents = 10

# Initialise agents

agents = []

class Agent():

pass

def \_\_init\_\_(self):

self.x = random.randint(0, 99)

self.y = random.randint(0, 99)

def \_\_str\_\_(self):

return self.\_\_class\_\_.\_\_name\_\_ + "(x=" + str(self.x) \

+ ", y=" + str(self.y) + ")"

def \_\_repr\_\_(self):

return str(self)

def move(self, x\_min, y\_min, x\_max, y\_max):

rn =random.random()

if rn < 0.5:

self.x = self.x + 1

else:

self.x = self.x - 1

#y-coordinate

rn = random.random()

#print("rn", rn)

if rn < 0.5:

self.y = self.y + 1

else:

self.y = self.y - 1

# Apply movement constraints.

if self.x < x\_min:

self.x = x\_min

if self.y < y\_min:

self.y = y\_min

if self.x > x\_max:

self.x = x\_max

if self.y > y\_max:

self.y = y\_max

agents = Agent()

print(agents)

**io:**

import csv

def read\_data():

# Read input data

f = open('C:/Users/xiaoyu/programming/data/input/in.txt', newline='')

data = []

n\_rows = 0

n\_cols = None

for line in csv.reader(f, quoting=csv.QUOTE\_NONNUMERIC):

row = []

for value in line:

row.append(value)

# print(value)

if n\_cols is None:

n\_cols = len(row)

assert(n\_cols == len(row))

data.append(row)

n\_rows += 1

f.close()

return data, n\_rows, n\_cols

return None #this line is never executed

**model:**

import random

import matplotlib

from matplotlib import pyplot as plt

import operator

import time

import my\_modules.agentframework as af

import my\_modules.io as io

# read\_data function in io

environment, n\_rows, n\_cols = io.read\_data()

# set the random seed

random.seed(0)

# set parameters

n\_agents = 10

# initialise agents

agents = []

for i in range(n\_agents):

# create an agent

agents.append(af.Agent())

print(agents[i])

print(agents)

def get\_distance(x0, y0, x1, y1):

x = x1 - x0

y = y1 - y0

#return 0.5 \*\* ( x\*x + y\*y )

return ( x\*x + y\*y ) \*\* 0.5

print(get\_distance(0,0,3,4)) # test result should be 5

max\_distance = 0

for i in range(len(agents)):

a = agents[i]

for j in range(len(agents)):

b = agents[j]

distance = get\_distance(a.x, a.y, b.x, b.y)

print("distance between", a, b, distance)

max\_distance = max(max\_distance, distance)

print("max\_distance", max\_distance)

# Variables for constraining movement.

# The minimum x coordinate.

x\_min = 0

# The minimum y coordinate.

y\_min = 0

# The maximum an agents x coordinate is allowed to be.

x\_max = n\_cols - 1

# The maximum an agents y coordinate is allowed to be.

y\_max = n\_rows - 1

# Move agents

n\_iterations = 1000

for n\_iterations in range(n\_iterations):

for i in range(n\_agents):

# Change agents(i) coordinates randomly

agents[i].move(x\_min, y\_min, x\_max, y\_max)

# Plot

plt.imshow(environment)

for i in range(n\_agents):

plt.scatter(agents[i].x, agents[i].y, color='black')

# Plot the coordinate with the largest x red

lx = max(agents, key=operator.attrgetter('x'))

plt.scatter(lx.x, lx.y, color='red')

# Plot the coordinate with the smallest x blue

sx = min(agents, key=operator.attrgetter('x'))

plt.scatter(sx.x, sx.y, color='blue')

# Plot the coordinate with the largest y yellow

ly = max(agents, key=operator.attrgetter('y'))

plt.scatter(ly.x, ly.y, color='yellow')

# Plot the coordinate with the smallest y green

sy = min(agents, key=operator.attrgetter('y'))

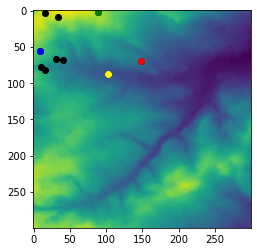
plt.scatter(sy.x, sy.y, color='green')

plt.show()

start = time.perf\_counter()

end = time.perf\_counter()

print("Time taken to calculate maximum distance", end - start, "second")



**model:**

import random

import matplotlib

from matplotlib import pyplot as plt

import operator

import time

import my\_modules.agentframework as af

import my\_modules.io as io

# read\_data function in io

environment, n\_rows, n\_cols = io.read\_data()

# set the random seed

random.seed(0)

# set parameters

n\_agents = 10

# initialise agents

agents = []

for i in range(n\_agents):

# create an agent

agents.append(af.Agent())

print(agents[i])

print(agents)

def get\_distance(x0, y0, x1, y1):

x = x1 - x0

y = y1 - y0

#return 0.5 \*\* ( x\*x + y\*y )

return ( x\*x + y\*y ) \*\* 0.5

print(get\_distance(0,0,3,4)) # test result should be 5

max\_distance = 0

for i in range(len(agents)):

a = agents[i]

for j in range(len(agents)):

b = agents[j]

distance = get\_distance(a.x, a.y, b.x, b.y)

print("distance between", a, b, distance)

max\_distance = max(max\_distance, distance)

print("max\_distance", max\_distance)

# Variables for constraining movement.

# The minimum x coordinate.

x\_min = 0

# The minimum y coordinate.

y\_min = 0

# The maximum an agents x coordinate is allowed to be.

x\_max = n\_cols - 1

# The maximum an agents y coordinate is allowed to be.

y\_max = n\_rows - 1

# Move agents

n\_iterations = 1000

for n\_iterations in range(n\_iterations):

for i in range(n\_agents):

# Change agents(i) coordinates randomly

agents[i].move(x\_min, y\_min, x\_max, y\_max)

# Plot

plt.imshow(environment)

plt.ylim(y\_min, y\_max)

plt.xlim(x\_min, x\_max)

for i in range(n\_agents):

plt.scatter(agents[i].x, agents[i].y, color='black')

# Plot the coordinate with the largest x red

lx = max(agents, key=operator.attrgetter('x'))

plt.scatter(lx.x, lx.y, color='red')

# Plot the coordinate with the smallest x blue

sx = min(agents, key=operator.attrgetter('x'))

plt.scatter(sx.x, sx.y, color='blue')

# Plot the coordinate with the largest y yellow

ly = max(agents, key=operator.attrgetter('y'))

plt.scatter(ly.x, ly.y, color='yellow')

# Plot the coordinate with the smallest y green

sy = min(agents, key=operator.attrgetter('y'))

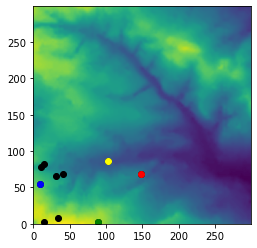
plt.scatter(sy.x, sy.y, color='green')

plt.show()

start = time.perf\_counter()

end = time.perf\_counter()

print("Time taken to calculate maximum distance", end - start, "second")



**agentframework:**

import random

random.seed(0)

n\_agents = 10

# Initialise agents

agents = []

class Agent():

pass

def \_\_init\_\_(self, i, n\_cols, n\_rows):

self.i = i

tnc = int(n\_cols / 3)

self.x = random.randint(tnc - 1, (2 \* tnc) - 1)

tnr = int(n\_rows / 3)

self.y = random.randint(tnr - 1, (2 \* tnr) - 1)

def \_\_str\_\_(self):

return self.\_\_class\_\_.\_\_name\_\_ + "(x=" + str(self.x) \

+ ", y=" + str(self.y) + ")"

def \_\_repr\_\_(self):

return str(self)

def move(self, x\_min, y\_min, x\_max, y\_max):

rn =random.random()

if rn < 0.5:

self.x = self.x + 1

else:

self.x = self.x - 1

#y-coordinate

rn = random.random()

#print("rn", rn)

if rn < 0.5:

self.y = self.y + 1

else:

self.y = self.y - 1

# Apply movement constraints.

if self.x < x\_min:

self.x = x\_min

if self.y < y\_min:

self.y = y\_min

if self.x > x\_max:

self.x = x\_max

if self.y > y\_max:

self.y = y\_max

**io:**

import csv

def read\_data():

# Read input data

f = open('C:/Users/xiaoyu/programming/data/input/in.txt', newline='')

data = []

n\_rows = 0

n\_cols = None

for line in csv.reader(f, quoting=csv.QUOTE\_NONNUMERIC):

row = []

for value in line:

row.append(value)

# print(value)

if n\_cols is None:

n\_cols = len(row)

assert(n\_cols == len(row))

data.append(row)

n\_rows += 1

f.close()

return data, n\_rows, n\_cols

return None #this line is never executed

**model:**

import random

import matplotlib

from matplotlib import pyplot as plt

import operator

import time

import my\_modules.agentframework as af

import my\_modules.io as io

# read\_data function in io

environment, n\_rows, n\_cols = io.read\_data()

# set the random seed

random.seed(0)

# set parameters

n\_agents = 10

# initialise agents

agents = []

for i in range(n\_agents):

# create an agent

agents.append(af.Agent(i, n\_cols, n\_rows))

print(agents[i])

print(agents)

def get\_distance(x0, y0, x1, y1):

x = x1 - x0

y = y1 - y0

#return 0.5 \*\* ( x\*x + y\*y )

return ( x\*x + y\*y ) \*\* 0.5

print(get\_distance(0,0,3,4)) # test result should be 5

max\_distance = 0

for i in range(len(agents)):

a = agents[i]

for j in range(len(agents)):

b = agents[j]

distance = get\_distance(a.x, a.y, b.x, b.y)

print("distance between", a, b, distance)

max\_distance = max(max\_distance, distance)

print("max\_distance", max\_distance)

# Variables for constraining movement.

# The minimum x coordinate.

x\_min = 0

# The minimum y coordinate.

y\_min = 0

# The maximum an agents x coordinate is allowed to be.

x\_max = n\_cols - 1

# The maximum an agents y coordinate is allowed to be.

y\_max = n\_rows - 1

# Move agents

n\_iterations = 1000

for n\_iterations in range(n\_iterations):

for i in range(n\_agents):

# Change agents(i) coordinates randomly

agents[i].move(x\_min, y\_min, x\_max, y\_max)

# Plot

plt.imshow(environment)

plt.ylim(y\_min, y\_max)

plt.xlim(x\_min, x\_max)

for i in range(n\_agents):

plt.scatter(agents[i].x, agents[i].y, color='black')

# Plot the coordinate with the largest x red

lx = max(agents, key=operator.attrgetter('x'))

plt.scatter(lx.x, lx.y, color='red')

# Plot the coordinate with the smallest x blue

sx = min(agents, key=operator.attrgetter('x'))

plt.scatter(sx.x, sx.y, color='blue')

# Plot the coordinate with the largest y yellow

ly = max(agents, key=operator.attrgetter('y'))

plt.scatter(ly.x, ly.y, color='yellow')

# Plot the coordinate with the smallest y green

sy = min(agents, key=operator.attrgetter('y'))

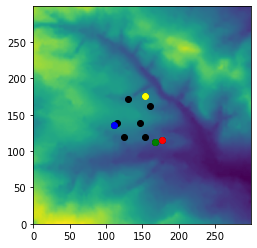
plt.scatter(sy.x, sy.y, color='green')

plt.show()

start = time.perf\_counter()

end = time.perf\_counter()

print("Time taken to calculate maximum distance", end - start, "second")



**agentframework:**

import random

import matplotlib

from matplotlib import pyplot as plt

import operator

import time

import my\_modules.agentframework as af

import my\_modules.io as io

# read\_data function in io

environment, n\_rows, n\_cols = io.read\_data()

# set the random seed

random.seed(0)

# set parameters

n\_agents = 10

# initialise agents

agents = []

for i in range(n\_agents):

# create an agent

agents.append(af.Agent(i, n\_cols, n\_rows))

print(agents[i])

print(agents)

def get\_distance(x0, y0, x1, y1):

x = x1 - x0

y = y1 - y0

#return 0.5 \*\* ( x\*x + y\*y )

return ( x\*x + y\*y ) \*\* 0.5

print(get\_distance(0,0,3,4)) # test result should be 5

max\_distance = 0

for i in range(len(agents)):

a = agents[i]

for j in range(len(agents)):

b = agents[j]

distance = get\_distance(a.x, a.y, b.x, b.y)

print("distance between", a, b, distance)

max\_distance = max(max\_distance, distance)

print("max\_distance", max\_distance)

# Variables for constraining movement.

# The minimum x coordinate.

x\_min = 0

# The minimum y coordinate.

y\_min = 0

# The maximum an agents x coordinate is allowed to be.

x\_max = n\_cols - 1

# The maximum an agents y coordinate is allowed to be.

y\_max = n\_rows - 1

# Move agents

n\_iterations = 1000

for n\_iterations in range(n\_iterations):

for i in range(n\_agents):

# Change agents(i) coordinates randomly

agents[i].move(x\_min, y\_min, x\_max, y\_max)

# Plot

plt.imshow(environment)

plt.ylim(y\_min, y\_max)

plt.xlim(x\_min, x\_max)

for i in range(n\_agents):

plt.scatter(agents[i].x, agents[i].y, color='black')

# Plot the coordinate with the largest x red

lx = max(agents, key=operator.attrgetter('x'))

plt.scatter(lx.x, lx.y, color='red')

# Plot the coordinate with the smallest x blue

sx = min(agents, key=operator.attrgetter('x'))

plt.scatter(sx.x, sx.y, color='blue')

# Plot the coordinate with the largest y yellow

ly = max(agents, key=operator.attrgetter('y'))

plt.scatter(ly.x, ly.y, color='yellow')

# Plot the coordinate with the smallest y green

sy = min(agents, key=operator.attrgetter('y'))

plt.scatter(sy.x, sy.y, color='green')

plt.show()

start = time.perf\_counter()

end = time.perf\_counter()

print("Time taken to calculate maximum distance", end - start, "second")

**io:**

import csv

def read\_data():

# Read input data

f = open('C:/Users/xiaoyu/programming/data/input/in.txt', newline='')

data = []

n\_rows = 0

n\_cols = None

for line in csv.reader(f, quoting=csv.QUOTE\_NONNUMERIC):

row = []

for value in line:

row.append(value)

# print(value)

if n\_cols is None:

n\_cols = len(row)

assert(n\_cols == len(row))

data.append(row)

n\_rows += 1

f.close()

return data, n\_rows, n\_cols

return None #this line is never executed

**model:**

import random

import matplotlib

from matplotlib import pyplot as plt

import operator

import time

import my\_modules.agentframework as af

import my\_modules.io as io

# read\_data function in io

environment, n\_rows, n\_cols = io.read\_data()

# set the random seed

random.seed(0)

# set parameters

n\_agents = 10

# initialise agents

agents = []

for i in range(n\_agents):

# create an agent

agents.append(af.Agent(i, n\_cols, n\_rows))

print(agents[i])

print(agents)

def get\_distance(x0, y0, x1, y1):

x = x1 - x0

y = y1 - y0

#return 0.5 \*\* ( x\*x + y\*y )

return ( x\*x + y\*y ) \*\* 0.5

print(get\_distance(0,0,3,4)) # test result should be 5

max\_distance = 0

for i in range(len(agents)):

a = agents[i]

for j in range(len(agents)):

b = agents[j]

distance = get\_distance(a.x, a.y, b.x, b.y)

print("distance between", a, b, distance)

max\_distance = max(max\_distance, distance)

print("max\_distance", max\_distance)

# Variables for constraining movement.

# The minimum x coordinate.

x\_min = 0

# The minimum y coordinate.

y\_min = 0

# The maximum an agents x coordinate is allowed to be.

x\_max = n\_cols - 1

# The maximum an agents y coordinate is allowed to be.

y\_max = n\_rows - 1

# Move agents

n\_iterations = 1000

for n\_iterations in range(n\_iterations):

for i in range(n\_agents):

# Change agents(i) coordinates randomly

agents[i].move(x\_min, y\_min, x\_max, y\_max)

# Plot

plt.imshow(environment)

plt.ylim(y\_min, y\_max)

plt.xlim(x\_min, x\_max)

for i in range(n\_agents):

plt.scatter(agents[i].x, agents[i].y, color='black')

# Plot the coordinate with the largest x red

lx = max(agents, key=operator.attrgetter('x'))

plt.scatter(lx.x, lx.y, color='red')

# Plot the coordinate with the smallest x blue

sx = min(agents, key=operator.attrgetter('x'))

plt.scatter(sx.x, sx.y, color='blue')

# Plot the coordinate with the largest y yellow

ly = max(agents, key=operator.attrgetter('y'))

plt.scatter(ly.x, ly.y, color='yellow')

# Plot the coordinate with the smallest y green

sy = min(agents, key=operator.attrgetter('y'))

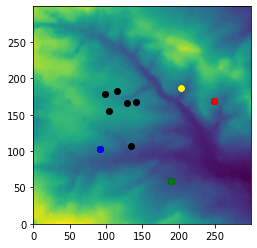
plt.scatter(sy.x, sy.y, color='green')

plt.show()

start = time.perf\_counter()

end = time.perf\_counter()

print("Time taken to calculate maximum distance", end - start, "second")



**agentframework:**

import random

random.seed(0)

n\_agents = 10

# Initialise agents

agents = []

class Agent():

pass

def \_\_init\_\_(self, i, environment, n\_cols, n\_rows):

self.i = i

self.environment = environment

tnc = int(n\_cols / 3)

self.x = random.randint(tnc - 1, (2 \* tnc) - 1)

tnr = int(n\_rows / 3)

self.y = random.randint(tnc - 1, (2 \* tnr) - 1)

self.store = 0

def \_\_str\_\_(self):

return self.\_\_class\_\_.\_\_name\_\_ + "(x=" + str(self.x) \

+ ", y=" + str(self.y) + ")"

def \_\_repr\_\_(self):

return str(self)

def move(self, x\_min, y\_min, x\_max, y\_max):

rn =random.random()

if rn < 0.5:

self.x = self.x + 1

else:

self.x = self.x - 1

#y-coordinate

rn = random.random()

#print("rn", rn)

if rn < 0.5:

self.y = self.y + 1

else:

self.y = self.y - 1

# Apply movement constraints.

if self.x < x\_min:

self.x = x\_min

if self.y < y\_min:

self.y = y\_min

if self.x > x\_max:

self.x = x\_max

if self.y > y\_max:

self.y = y\_max

def eat(self):

if self.environment[self.y][self.x] >= 10:

self.environment[self.y][self.x] -= 10

self.store += 10

**io:**

import csv

def read\_data():

# Read input data

f = open('C:/Users/xiaoyu/programming/data/input/in.txt', newline='')

data = []

n\_rows = 0

n\_cols = None

for line in csv.reader(f, quoting=csv.QUOTE\_NONNUMERIC):

row = []

for value in line:

row.append(value)

# print(value)

if n\_cols is None:

n\_cols = len(row)

assert(n\_cols == len(row))

data.append(row)

n\_rows += 1

f.close()

return data, n\_rows, n\_cols

return None #this line is never executed

model:

import random

import matplotlib

from matplotlib import pyplot as plt

import operator

import time

import my\_modules.agentframework as af

import my\_modules.io as io

# read\_data function in io

environment, n\_rows, n\_cols = io.read\_data()

# set the random seed

random.seed(0)

# set parameters

n\_agents = 10

# initialise agents

agents = []

for i in range(n\_agents):

# create an agent

agents.append(af.Agent(i, environment, n\_cols, n\_rows))

print(agents[i])

print(agents)

def get\_distance(x0, y0, x1, y1):

x = x1 - x0

y = y1 - y0

#return 0.5 \*\* ( x\*x + y\*y )

return ( x\*x + y\*y ) \*\* 0.5

print(get\_distance(0,0,3,4)) # test result should be 5

max\_distance = 0

for i in range(len(agents)):

a = agents[i]

for j in range(len(agents)):

b = agents[j]

distance = get\_distance(a.x, a.y, b.x, b.y)

print("distance between", a, b, distance)

max\_distance = max(max\_distance, distance)

print("max\_distance", max\_distance)

# Variables for constraining movement.

# The minimum x coordinate.

x\_min = 0

# The minimum y coordinate.

y\_min = 0

# The maximum an agents x coordinate is allowed to be.

x\_max = n\_cols - 1

# The maximum an agents y coordinate is allowed to be.

y\_max = n\_rows - 1

# Move agents

n\_iterations = 1000

for n\_iterations in range(n\_iterations):

for i in range(n\_agents):

# Change agents(i) coordinates randomly

agents[i].move(x\_min, y\_min, x\_max, y\_max)

# Plot

plt.imshow(environment)

plt.ylim(y\_max / 3, y\_max \* 2 / 3)

plt.xlim(x\_max / 3, x\_max \* 2 / 3)

for i in range(n\_agents):

plt.scatter(agents[i].x, agents[i].y, color='black')

# Plot the coordinate with the largest x red

lx = max(agents, key=operator.attrgetter('x'))

plt.scatter(lx.x, lx.y, color='red')

# Plot the coordinate with the smallest x blue

sx = min(agents, key=operator.attrgetter('x'))

plt.scatter(sx.x, sx.y, color='blue')

# Plot the coordinate with the largest y yellow

ly = max(agents, key=operator.attrgetter('y'))

plt.scatter(ly.x, ly.y, color='yellow')

# Plot the coordinate with the smallest y green

sy = min(agents, key=operator.attrgetter('y'))

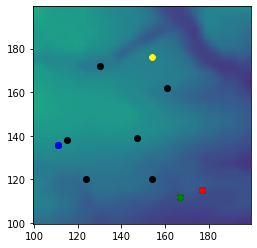
plt.scatter(sy.x, sy.y, color='green')

plt.show()

start = time.perf\_counter()

end = time.perf\_counter()

print("Time taken to calculate maximum distance", end - start, "second")



**model:**

import random

import matplotlib

from matplotlib import pyplot as plt

import operator

import time

import my\_modules.agentframework as af

import my\_modules.io as io

# read\_data function in io

environment, n\_rows, n\_cols = io.read\_data()

# set the random seed

random.seed(0)

# set parameters

n\_agents = 10

# initialise agents

agents = []

for i in range(n\_agents):

# create an agent

agents.append(af.Agent(i, environment, n\_cols, n\_rows))

print(agents[i])

print(agents)

def get\_distance(x0, y0, x1, y1):

x = x1 - x0

y = y1 - y0

#return 0.5 \*\* ( x\*x + y\*y )

return ( x\*x + y\*y ) \*\* 0.5

print(get\_distance(0,0,3,4)) # test result should be 5

max\_distance = 0

for i in range(len(agents)):

a = agents[i]

for j in range(len(agents)):

b = agents[j]

distance = get\_distance(a.x, a.y, b.x, b.y)

print("distance between", a, b, distance)

max\_distance = max(max\_distance, distance)

print("max\_distance", max\_distance)

# Variables for constraining movement.

# The minimum x coordinate.

x\_min = 0

# The minimum y coordinate.

y\_min = 0

# The maximum an agents x coordinate is allowed to be.

x\_max = n\_cols - 1

# The maximum an agents y coordinate is allowed to be.

y\_max = n\_rows - 1

# Move agents

n\_iterations = 100

for n\_iterations in range(n\_iterations):

for i in range(n\_agents):

# Change agents(i) coordinates randomly

agents[i].move(x\_min, y\_min, x\_max, y\_max)

# Plot

plt.imshow(environment)

plt.ylim(y\_max / 3, y\_max \* 2 / 3)

plt.xlim(x\_max / 3, x\_max \* 2 / 3)

for i in range(n\_agents):

plt.scatter(agents[i].x, agents[i].y, color='black')

# Plot the coordinate with the largest x red

lx = max(agents, key=operator.attrgetter('x'))

plt.scatter(lx.x, lx.y, color='red')

# Plot the coordinate with the smallest x blue

sx = min(agents, key=operator.attrgetter('x'))

plt.scatter(sx.x, sx.y, color='blue')

# Plot the coordinate with the largest y yellow

ly = max(agents, key=operator.attrgetter('y'))

plt.scatter(ly.x, ly.y, color='yellow')

# Plot the coordinate with the smallest y green

sy = min(agents, key=operator.attrgetter('y'))

plt.scatter(sy.x, sy.y, color='green')

plt.show()

start = time.perf\_counter()

end = time.perf\_counter()

print("Time taken to calculate maximum distance", end - start, "second")

