In [3]:
```
import math
import random
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
```

Section 1: Generating the data

In [4]:
```
random.seed(10)
random.random()
```

Out[4]: 0.5714025946899135

In [5]:
```
MAX_X_POSITION = 700
MAX_Y_POSITION = 500
MAX_PARTICLE_RADIUS = 5
```

1) Write a function that generates a random particle

In [40]:
```
def generate_particle(max_x=MAX_X_POSITION, max_y=MAX_Y_POSITION, max_r=MAX_PARTICLE_RADIUS):
    x = random.uniform(0,max_x)
    y = random.uniform(0,max_y)
    r = random.uniform(0,max_r)
    return (x,y,r)
```

2) Generate 1,000 random particles using `random.seed(10)`

In [47]:
```
random.seed(10)
particles = [ generate_particle(MAX_X_POSITION, MAX_Y_POSITION, MAX_PARTICLE_RADIUS) for i in range(1000)] #
len(particles)
```

Out[47]:
```
1000
```

Activity 3: Store your particles in a CSV file

In [54]:
```
# store your particles in `particles.csv`
import csv
fields = ['x','y','r']

with open('particles.csv','w') as f:
    writer = csv.writer(f)
    writer.writerow(fields)
    writer.writerows(particles)
```

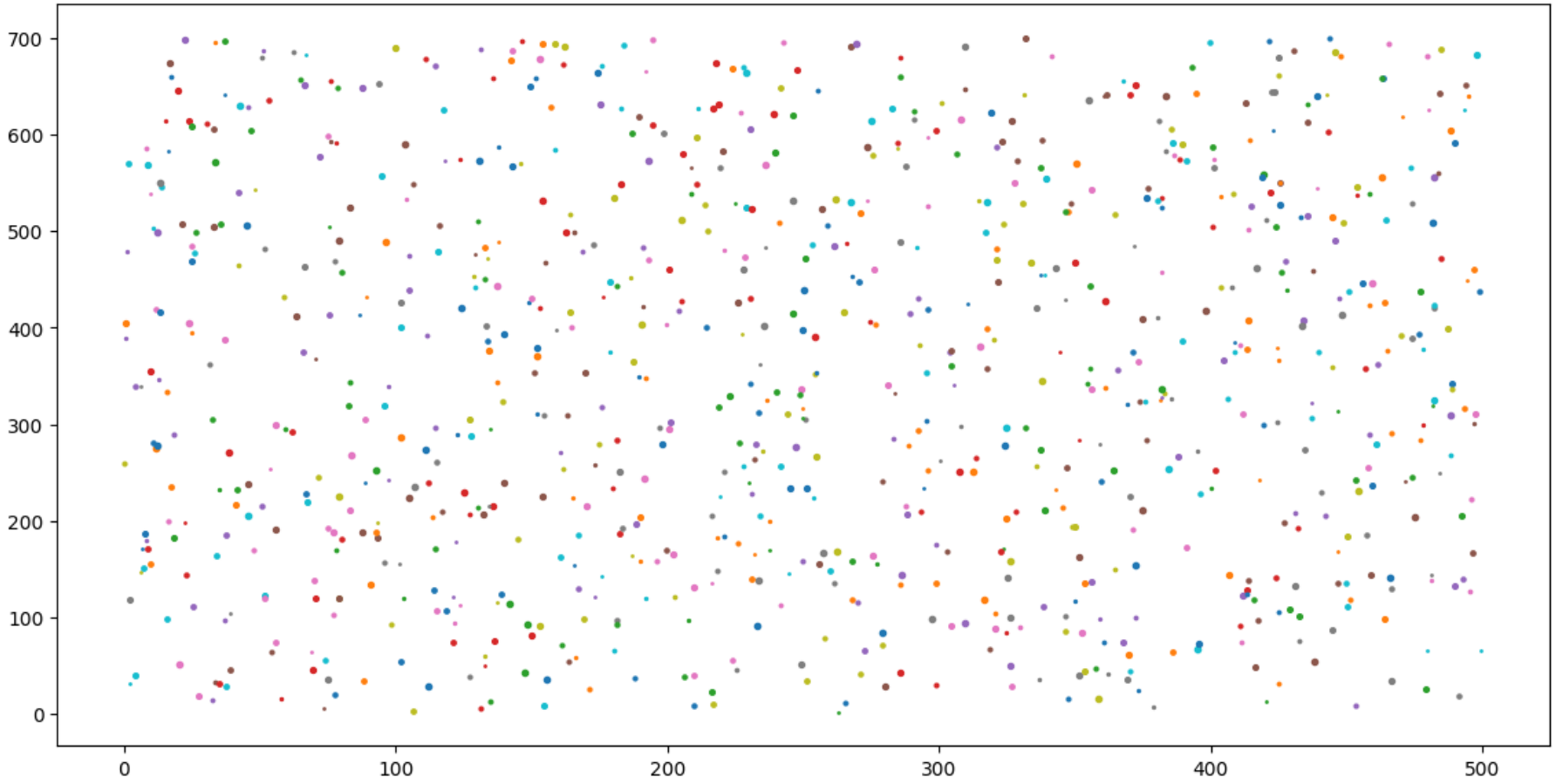
Here you can see the first rows of the expected `particles.csv` :

In [55]:
```
!head particles.csv
```

```
x,y,r
399.98181628293946,214.4445273375573,2.890456505672352
144.2687624976512,406.66062567866004,4.117944362667227
457.43077373082303,80.11477825940983,2.603346798199623
229.4409681354652,124.99833834320017,4.764084545729558
697.5898947776186,22.278191225216514,4.300805186431455
422.23342767772954,190.80299295955894,1.4180910895335757
472.4753929944692,228.41557552915282,3.4293074273729736
463.2924240366458,66.4890723561064,3.8391890697199527
687.6892743078336,484.6940802024594,3.066634102733545
```

4) Plot your particles in a matplotlib plane

In [76]:
```
fig, ax = plt.subplots(figsize=(14, 7))
# your code here...
for x,y,r in particles:
    ax.scatter(y,x,s=int(r*2))
```



Section 2: Spatial analysis

5) Write the function `calculate_particle_position`

In [75]:
```
from enum import Enum

class ParticlePosition(Enum):
    PARTIALLY_CONTAINED = 1
    COMPLETELY_CONTAINED = 2
    OUTSIDE = 3
```

In [78]:
```
min(5,4)
```

Out[78]: 4

In [244]:
```
def calculate_particle_position(p, A):
    px, py, pr = p
    ax, ay, ar = A
    d = math.sqrt((px-ax)**2 + (py-ay)**2)
    d_min = min(pr,ar)
    d_max = max(pr,ar)
    if d > pr+ar:
        return ParticlePosition.OUTSIDE
    elif d_max >= d + d_min:
        return ParticlePosition.COMPLETELY_CONTAINED
    else:
        return ParticlePosition.PARTIALLY_CONTAINED
```

If you want to test your solution, your function should work in the following cases:

In [245]:
```
# Should return: ParticlePosition.COMPLETELY_CONTAINED
calculate_particle_position((4,5,1), (6, 5, 3))
```

Out[245]: <ParticlePosition.COMPLETELY_CONTAINED: 2>

In [246]:
```
# Should return: ParticlePosition.PARTIALLY_CONTAINED
calculate_particle_position((8,5,2), (6, 5, 3))
```

Out[246]: <ParticlePosition.PARTIALLY_CONTAINED: 1>

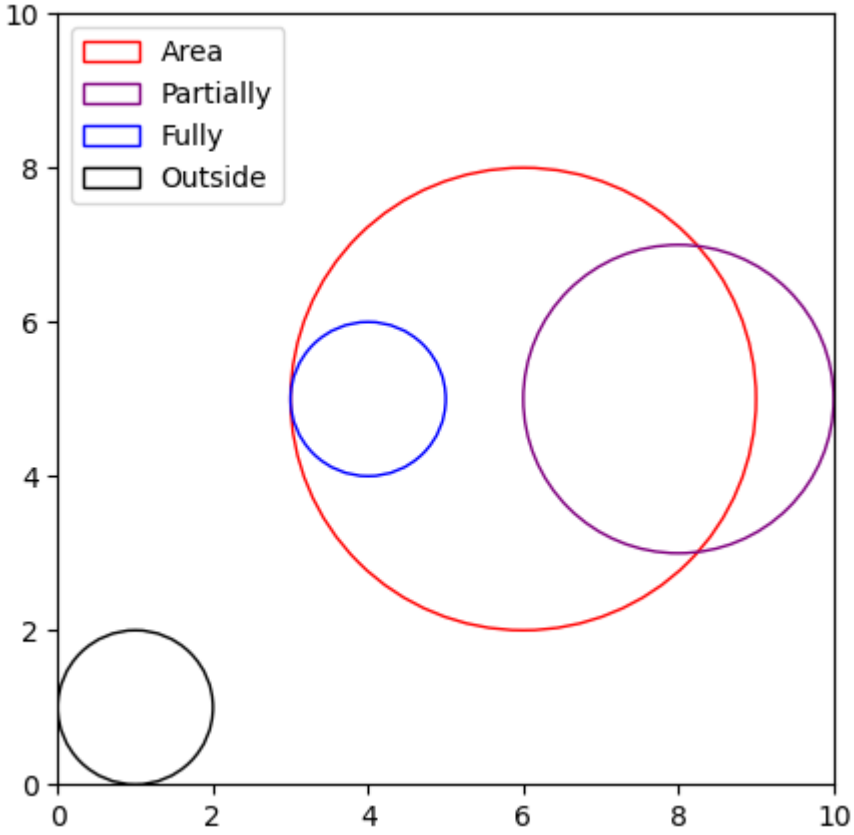
In [247]:
```
# Should return: ParticlePosition.OUTSIDE
calculate_particle_position((1,1,2), (6, 5, 3))
```

Out[247]:
```
<ParticlePosition.OUTSIDE: 3>
```

And you can visualize the above examples with matplotlib using:

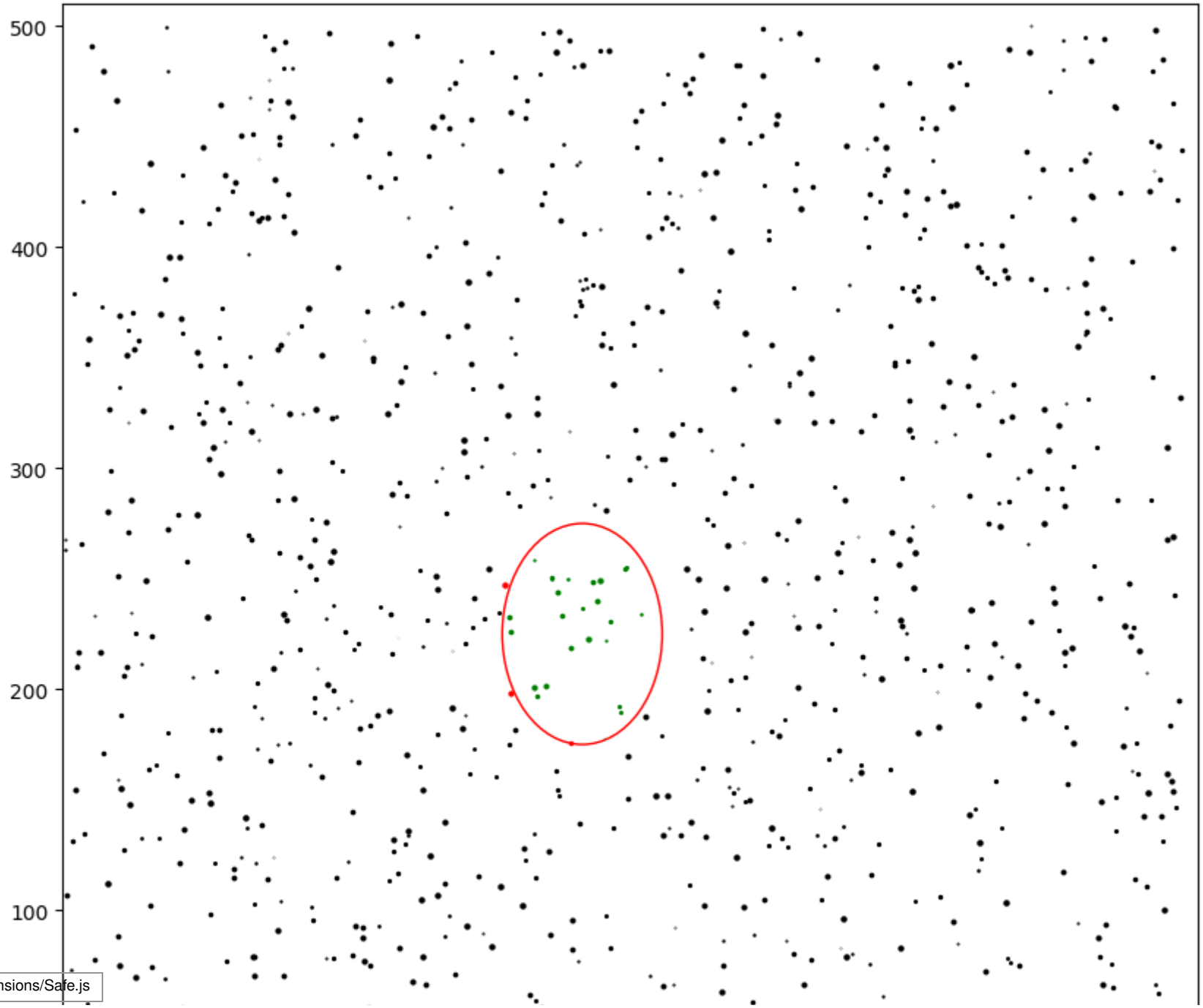
In [248]:
```
fig, ax = plt.subplots(figsize=(5, 5))

area = plt.Circle((6, 5), 3, color='r', fill=False, label='Area')
partial = plt.Circle((8, 5), 2, color='purple', fill=False, label='Partially')
fully = plt.Circle((4, 5), 1, color='blue', fill=False, label='Fully')
outside = plt.Circle((1, 1), 1, color='black', fill=False, label='Outside')

ax.add_patch(area)
ax.add_patch(partial)
ax.add_patch(fully)
ax.add_patch(outside)

ax.set_xlim((0, 10))
ax.set_ylim((0, 10))

ax.legend(loc='best')
```

Out[248]:
```
<matplotlib.legend.Legend at 0x7f451fb918d0>
```



In [249]:
```
A = (325, 225, 50)

ax, ay, ar = A
fig, axis = plt.subplots(figsize=(10, 10))

area = plt.Circle((ax, ay), ar, color='r', fill=False, label='Area')
axis.add_patch(area)

# Plot your particles here
for x,y,r in particles:
    res = calculate_particle_position((x,y,r), (ax, ay, ar))
    if res == ParticlePosition.OUTSIDE:
        axis.scatter(x,y,s=r,c='black')
    elif res == ParticlePosition.PARTIALLY_CONTAINED:
        axis.scatter(x,y,s=r,c='red')
    # your code here
    elif res==ParticlePosition.COMPLETELY_CONTAINED:
        axis.scatter(x,y,s=r,c='green')

axis.set_xlim((0, 710))
axis.set_ylim((0, 510))
```

Out[249]:
```
(0.0, 510.0)
```



In [ ]:

In [ ]:

In [ ]: