Question 5

November 8, 2019

1 Locating a Point

We're trying to locate a point given a plane by zeroing in on one of the 4 possible quadrants.

For this purpose, I've created my own Python class, which takes care of the plane's position, its quadrants, and its bounding box, making it easier for us to draw the boxes when the required quadrant is found.

[1]:
```
from Plane import Plane
```

Now, we're defining the overall plane's boundaries. I've kept it limited to [0,10] on both axes.

[2]:
```
xs = [0,10]
ys = [0,10]
```

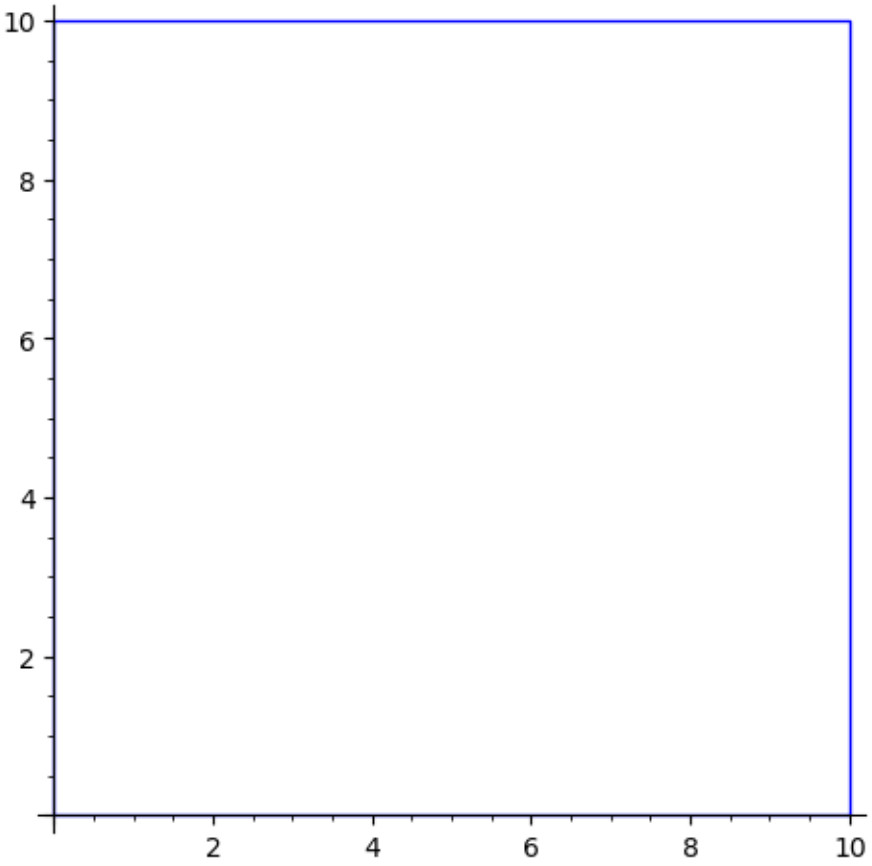
Creating the Plane object using the constructor method which takes in 2 lists as inputs, one with the x-range and the other with the y-range.

[3]:
```
plane = Plane(xs,ys)
print(plane.boundingBox())
```

```
[[0, 0], [0, 10], [10, 10], [10, 0]]
```

Now, we're plotting our sample space, before we start locating the point accurately.

[4]:
```
p = []
p.append(polygon(plane.boundingBox(),fill=False))
sum(p).show()
```



I've randomly chosen the point to be `[3,4]`, but it could indeed be anything within the range of the plane.

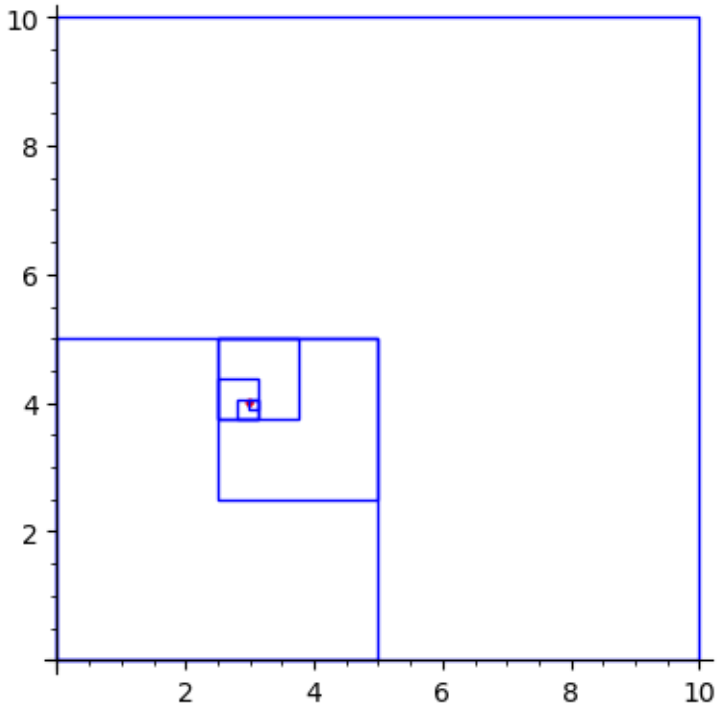
(Note that if you do put in a point outside this plane region, you'll get a NoneType object, which will stop the code from executing. So, the exceptions are handled in some way.)

[5]:
```
my_point = [3,4]
```

Now, we will zero in on our point 6 times. Change the `depth` from 6 to a larger value to make it zero in more closely.

The plot below shows the point in red, and all the boxes in blue.

[6]:
```
depth = 6
for i in range(depth):
    plane = plane.whichQuadrant(my_point)
    p.append(polygon(plane.boundingBox(),fill=False))
p.append(point(my_point,color='red'))
sum(p).show()
```


10
8
6
4
2
2
4
6
8
10

```
class Plane:
    def __init__(self, lstx, lsty):
        self.x_min = lstx[0]
        self.x_max = lstx[1]
        self.y_min = lsty[0]
        self.y_max = lsty[1]
        self.x_mid = (self.x_min+self.x_max)/2
        self.y_mid = (self.y_min+self.y_max)/2
        self.q1 = [[self.x_min,self.x_mid],[self.y_min,self.y_mid]]
        self.q2 = [[self.x_mid,self.x_max],[self.y_min,self.y_mid]]
        self.q3 = [[self.x_mid,self.x_max],[self.y_mid,self.y_max]]
        self.q4 = [[self.x_min,self.x_mid],[self.y_mid,self.y_max]]
    def whichQuadrant(self, point):
        x = point[0]
        y = point[1]
        if x>=self.x_min and x<self.x_mid:
            if y>=self.y_min and y<self.y_mid:
                return Plane(self.q1[0],self.q1[1])
            elif y>=self.y_mid and y<self.y_max:
                return Plane(self.q4[0],self.q4[1])
        elif x>=self.x_mid and x<self.x_max:
            if y>=self.y_min and y<self.y_mid:
                return Plane(self.q2[0],self.q2[1])
            elif y>=self.y_mid and y<self.y_max:
                return Plane(self.q3[0],self.q3[1])
        else:
            return None
    def __str__(self):
        return "x_min = "+str(self.x_min)+", x_max = "+str(self.x_max)+"\ny_min = "+
str(self.y_min)+", y_max = "+str(self.y_max)
    def boundingBox(self):
        return [[self.x_min,self.y_min],[self.x_min,self.y_max],[self.x_max,self.y_m
ax],[self.x_max,self.y_min]]
```