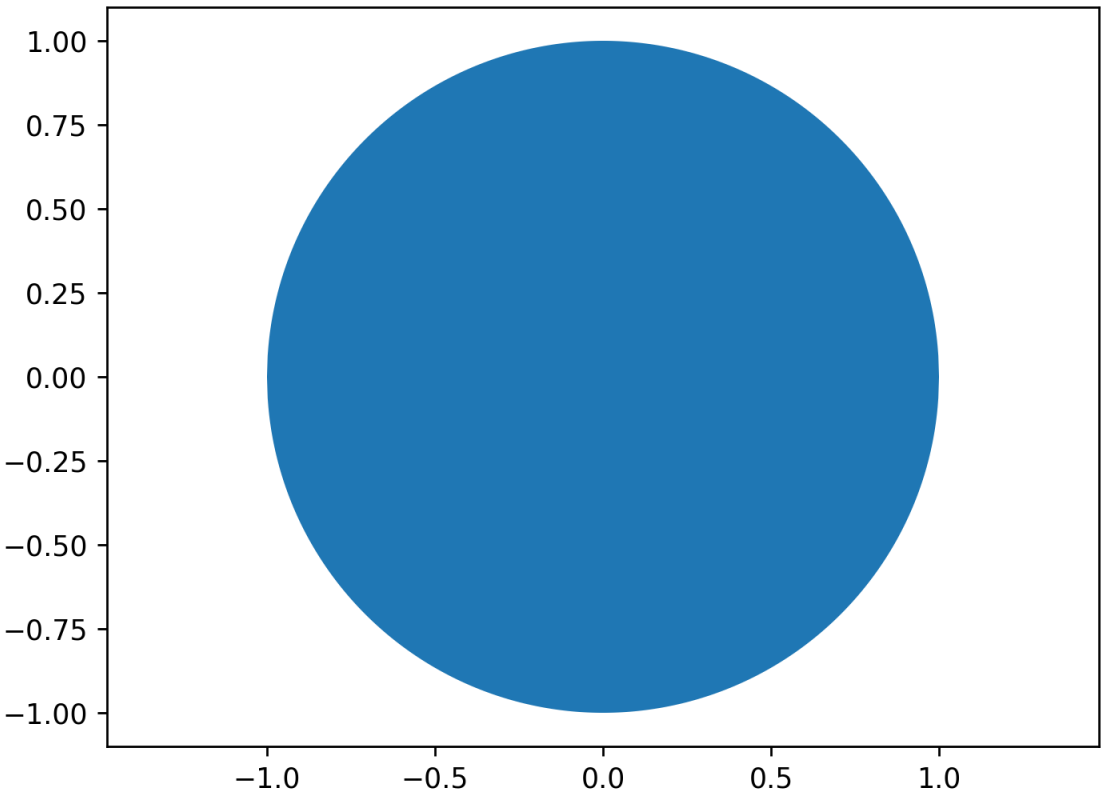
In [1]:
```
import matplotlib.pyplot as plt
import numpy as np

%matplotlib notebook
```

In [2]:
```
x_ = np.linspace(-1, 1, 1000)
semicircle = np.sqrt(1-x_**2)
plt.fill_between(x_, -semicircle, semicircle)
plt.axis('equal');
```



Gibbs sampling

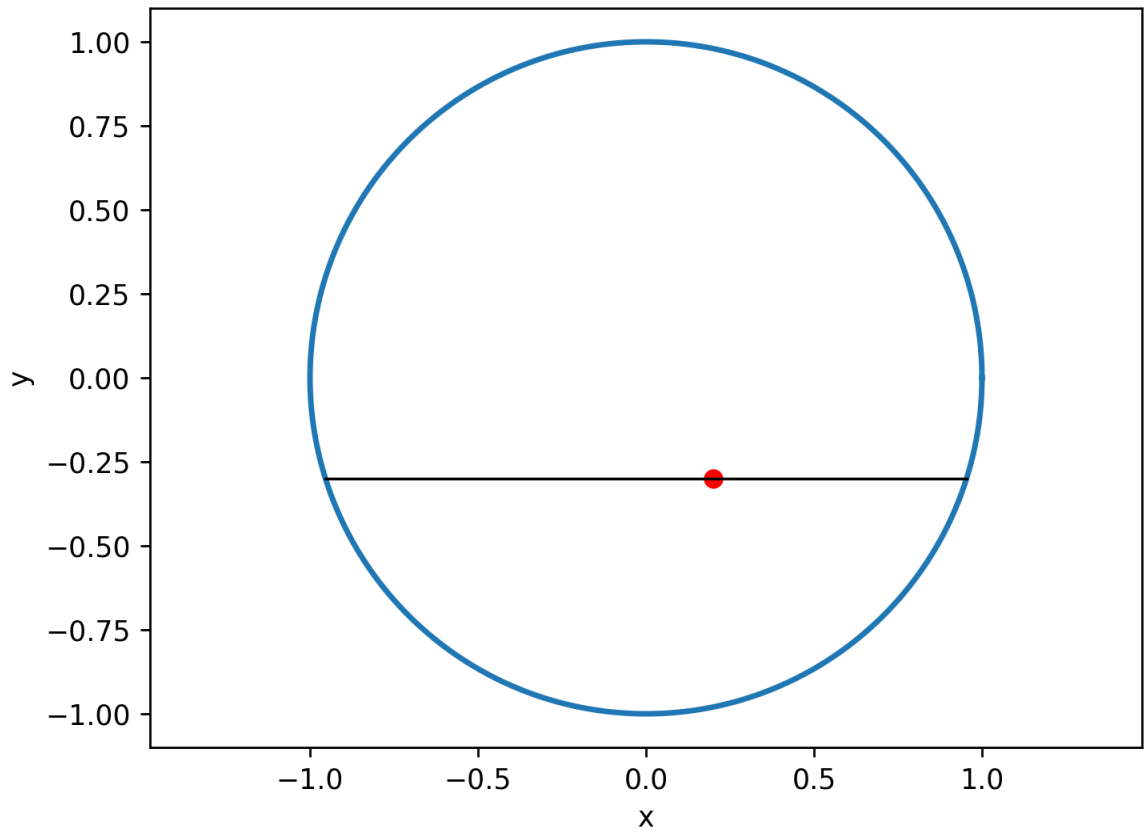
In [3]:
```
t = np.arange(0, 2 * np.pi, .001)

x_circ = np.cos(t)
y_circ = np.sin(t)

plt.figure()
plt.plot(x_circ, y_circ, lw=2)
plt.axis('equal')
plt.xlabel('x')
plt.ylabel('y')
initial_point = np.array([0.2, -0.3])

x_val, y_val = initial_point

plt.scatter(x_val, y_val, color='red')
stage = 'find_x_dist_given_y'
line = None
```



In [4]:

```
### Interactive demo cell
# Keep running this over and over to generate points on the graph from the
# previous cell. Make sure you're using `%matplotlib notebook`
# Each time you run this cell, the variable `stage` controls which of the
# 4 if/elif blocks executes, and then changes it so that the next block
# will execute next time.

print(stage)
# Draw the horizontal line
if stage == "find_x_dist_given_y":
    if line:
        line.remove()
    x_max = np.sqrt(1 - y_val ** 2)
    x_min = -x_max
    # Draw horizontal line
    line,  = plt.plot([x_min, x_max], [y_val, y_val], color='black', lw=1)
    stage = 'sample_x_given_y'

# Sample a point
elif stage == 'sample_x_given_y':
    plt.scatter(x_val, y_val, color='gray')
    x_val = np.random.uniform(x_min, x_max)
    plt.scatter(x_val, y_val, color='red')
    stage = "find_y_dist_given_x"

# Draw the vertical line
elif stage == "find_y_dist_given_x":
    if line:
        line.remove()
    y_max = np.sqrt(1 - x_val ** 2)
    y_min = -y_max
```

```
    # Draw vertical line
    line,  = plt.plot([x_val, x_val], [y_min, y_max], color='black', lw=1)
    stage = 'sample_y_given_x'

# Sample a point
elif stage == 'sample_y_given_x':
    plt.scatter(x_val, y_val, color='gray')
    y_val = np.random.uniform(y_min, y_max)
    plt.scatter(x_val, y_val, color='red')
    stage = 'find_x_dist_given_y'
```

find_x_dist_given_y

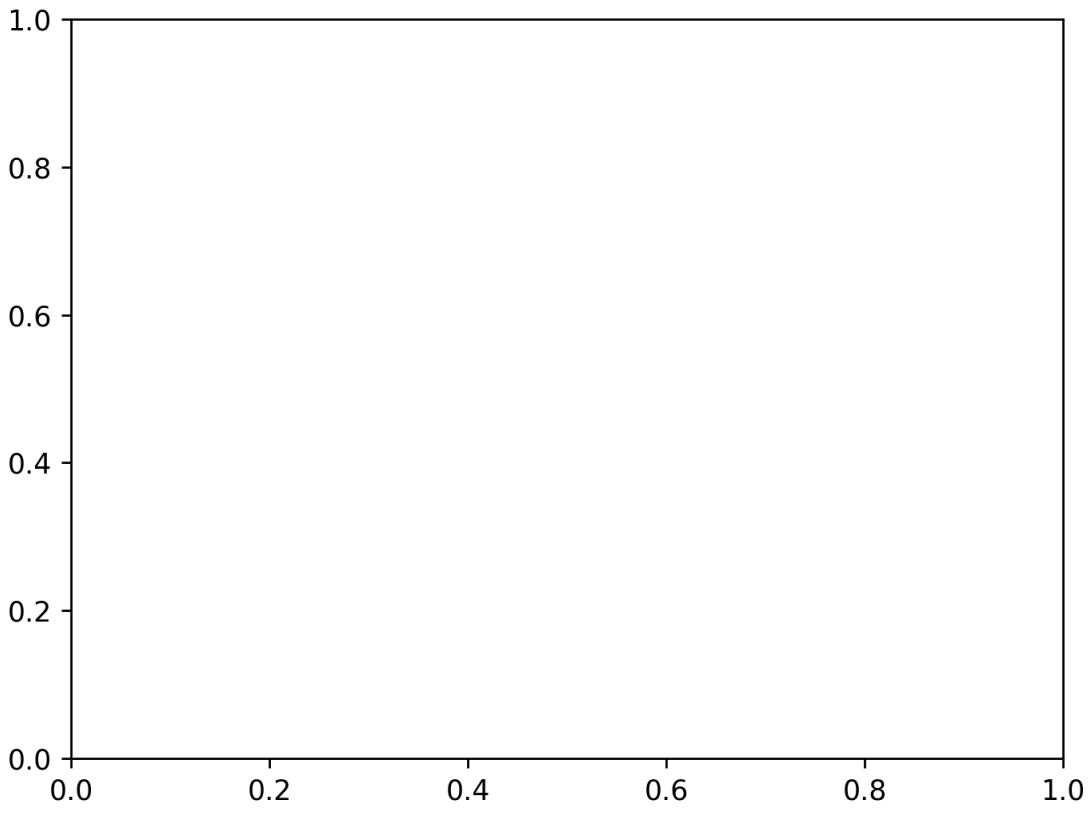
In [5]:
```
NUM_ITERATIONS = 500

x, y = [-0.1, 0.7]
samples = []

for i in range(500):
    # Sample x given y (hint: use Pythagorean theorem)
    x = ...
    samples.append([x, y])

    # Sample x2 given x1 similarly
    x = ...
    samples.append([x, y])

samples_arr = np.array(samples)
plt.figure()
plt.scatter(samples_arr[:, 0], samples_arr[:, 1], color='orange')
plt.axis('equal')
```



```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [5], line 17
     15 samples_arr = np.array(samples)
     16 plt.figure()
---> 17 plt.scatter(samples_arr[:, 0], samples_arr[:, 1], color='orange')
     18 plt.axis('equal')

File /opt/conda/lib/python3.9/site-packages/matplotlib/pyplot.py:2817, in scatter(x, y,
 s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths, edgecolors, plotnonfinite, dat
a, **kwargs)
   2812 @_copy_docstring_and_deprecators(Axes.scatter)
   2813 def scatter(
   2814         x, y, s=None, c=None, marker=None, cmap=None, norm=None,
   2815         vmin=None, vmax=None, alpha=None, linewidths=None, *,
   2816         edgecolors=None, plotnonfinite=False, data=None, **kwargs):
-> 2817     __ret = gca().scatter(
   2818         x, y, s=s, c=c, marker=marker, cmap=cmap, norm=norm,
   2819         vmin=vmin, vmax=vmax, alpha=alpha, linewidths=linewidths,
   2820         edgecolors=edgecolors, plotnonfinite=plotnonfinite,
   2821         **({"data": data} if data is not None else {}), **kwargs)
   2822     sci(__ret)
   2823     return __ret

File /opt/conda/lib/python3.9/site-packages/matplotlib/__init__.py:1414, in _preprocess_
data.<locals>.inner(ax, data, *args, **kwargs)
   1411 @functools.wraps(func)
   1412 def inner(ax, *args, data=None, **kwargs):
   1413     if data is None:
-> 1414         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1416     bound = new_sig.bind(ax, *args, **kwargs)
   1417     auto_label = (bound.arguments.get(label_namer)
   1418                   or bound.kwargs.get(label_namer))

File /opt/conda/lib/python3.9/site-packages/matplotlib/axes/_axes.py:4457, in Axes.scatt
er(self, x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths, edgecolors, plot
nonfinite, **kwargs)
   4451         linewidths = [
   4452             lw if lw is not None else rcParams['lines.linewidth']
   4453             for lw in linewidths]
   4455 offsets = np.ma.column_stack([x, y])
-> 4457 collection = mcoll.PathCollection(
   4458         (path,), scales,
   4459         facecolors=colors,
   4460         edgecolors=edgecolors,
   4461         linewidths=linewidths,
   4462         offsets=offsets,
   4463         transOffset=kwargs.pop('transform', self.transData),
   4464         alpha=alpha
   4465         )
   4466 collection.set_transform(mtransforms.IdentityTransform())
   4467 collection.update(kwargs)

File /opt/conda/lib/python3.9/site-packages/matplotlib/collections.py:1012, in PathColle
ction.__init__(self, paths, sizes, **kwargs)
    998 def __init__(self, paths, sizes=None, **kwargs):
    999     """
   1000     Parameters
   1001     ----------
   (...)
   1009         Forwarded to `.Collection`.
```

```
   1010     """
-> 1012     super().__init__(**kwargs)
   1013     self.set_paths(paths)
   1014     self.set_sizes(sizes)

File /opt/conda/lib/python3.9/site-packages/matplotlib/collections.py:196, in Collectio
n.__init__(self, edgecolors, facecolors, linewidths, linestyles, capstyle, joinstyle, an
tialiaseds, offsets, transOffset, norm, cmap, pickradius, hatch, urls, zorder, **kwargs)
    193     self._joinstyle = None
    195 if offsets is not None:
--> 196     offsets = np.asanyarray(offsets, float)
    197     # Broadcast (2,) -> (1, 2) but nothing else.
    198     if offsets.shape == (2,):

TypeError: float() argument must be a string or a number, not 'ellipsis'
```

In [ ]: