In [5]:
```
import sympy as sp
from sympy.plotting import (plot, plot_parametric)
```

Volume in Python

This section is all about setting up the integration by hand, then doing the integration (and other steps as needed) in Python.

NOTE: The examples here (and in future overviews) are NOT a copy/paste to solve the problems in lab. However, they will USE many of the features you will use to solve your problems.

EXAMPLES:

Given the region bounded by the curves $y = 1 - x^2$ and $y = x^6 - x + 1$:

a) Find the area of the region

b) Find the volume of the solid formed by rotating the region about the x-axis.

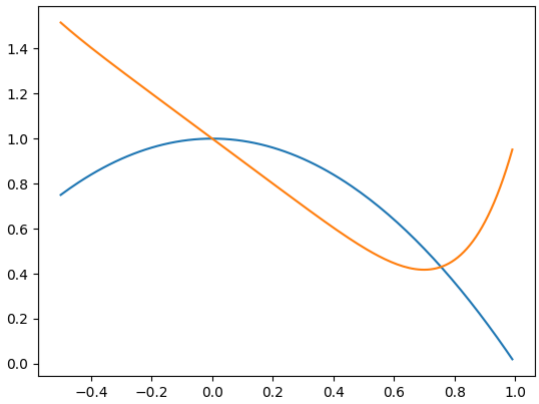
c) The volume of the solid formed by rotating the region about the line $y = c$ is 5. Find the value of c given $c < 0$.

So we follow the above process to solve these, though part c) will require a little additional thought on our part.

In [3]:
```
matplotlib notebook
```

In [6]:
```
x=sp.symbols('x')
# Graph the functions: use symbolic plot (shown in Lab 1 overview)
# or matplotlib.pyplot.plot (shown here)
import matplotlib.pyplot as plt
from numpy import arange # To get the arange command: don't want to import EVERYTHING as some commands are the same as in SymPy!

xplt=arange(-0.5,1,0.01) # Guessed at the boundaries, then adjusted
plt.plot(xplt,1-xplt**2,xplt,xplt**6-xplt+1)
```



Out[6]:
```
[<matplotlib.lines.Line2D at 0x174b9892d30>,
 <matplotlib.lines.Line2D at 0x174b9892d90>]
```

In [7]:
```
# Find points of intersection. Will try "solve" first, then
# use "nsolve" if that doesn't work.
f=1-x**2
g=x**6-x+1
inter=sp.solve(f-g,x)
print(inter) # Ran code here. The first and sixth solutions are real values
a=inter[0]
b=inter[5]
Area=sp.integrate(f-g,(x,a,b))
print('The area between the curves is',Area.evalf())
```

```
[0, 1/2 - sqrt(3)*I/2, 1/2 + sqrt(3)*I/2, -1/3 + (-1/2 - sqrt(3)*I/2)*(sqrt(69)/18 + 25/54)**(1/3) + 1/(9*(-1/2 - sqrt(3)*I/2)*(sqrt(69)/18 + 25/54)**(1/3)), -1/3 + 1/(9*(-1/2 + sqrt(3)*I/2)*(sqrt(69)/18 + 25/54)**(1/3)) + (-1/2 + sqrt(3)*I/2)*(sqrt(69)/18 + 25/54)**(1/3), -1/3 + 1/(9*(sqrt(69)/18 + 25/54)**(1/3)) + (sqrt(69)/18 + 25/54)**(1/3)]
The area between the curves is 0.121579206975124
```

We have done steps 1-2 for part B, so we simply need to integrate the volume (slicing) formula

In [9]:
```
Volb=sp.integrate(pi*(f**2-g**2),(x,a,b))
print('The volume by rotating about the x-axis is',Volb.evalf())
```

```
The volume by rotating about the x-axis is 0.544025155285806
```

We have done steps 1-2 for Part c) as well, but the rest is little more complicated. As always, we think about how we would solve it by hand:

i) Integrate to find the volume in terms of c (by adjusting your radius to be $f - c$ and $g - c$ respectively) ii) Set the volume = 5 and solve for c

So we need to start by defining c as a symbolic variable.

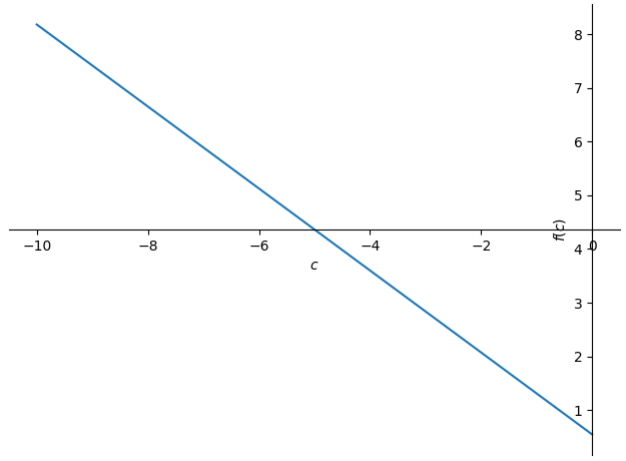
In [10]:
```
c=sp.symbols('c')
# step i: integrate
Volc=sp.integrate(pi*((f-c)**2-(g-c)**2),(x,a,b))
# step ii: solve
cval=sp.solve(Volc-5,c)
print('To get a volume of 5, rotate the region about the line y=',cval)
# Let's try that with evalf instead! NOTE that it's still in a list even though there is only one solution.
print('To get a volume of 5, rotate the region about the line y=',cval[0].evalf())
```

```
To get a volume of 5, rotate the region about the line y= [(-33257126344713561273*sqrt(69)*pi*(3*sqrt(69) + 25)**(1/3)/26 - 690636098267697592479*pi*(3*sqrt(69) + 25)**(1/3)/65 - 6586470671597856225 3*pi*(6*sqrt(69) + 50)**(2/3)/260 - 304968419471041 20*sqrt(69)*pi*(6*sqrt(69) + 50)**(2/3) + 155640637431317433761*2**(1/3)*pi/52 + 936846545599429468251*2**(1/3)*sqrt(69)*pi/260 + 92087805661488003 15*(6*sqrt(69) + 50)**(2/3) + 110860690433547600 0*sqrt(69)*(6*sqrt(69) + 50)**(2/3))/(pi*(-10291977232794784961*(3*sqrt(69) + 25)**(1/3) - 1239008459109314423*sqrt(69)*(3*sqrt(69) + 25)**(1/3) + 2923422401952000 1*2**(2/3)*(3*sqrt(69) + 25)**(2/3) + 3519386997890400*2**(2/3)*sqrt(69)*(3*sqrt(69) + 25)**(2/3) + 2192872877048748996 8*2**(1/3) + 2639908720121587246*2**(1/3)*sqrt(69)))]
To get a volume of 5, rotate the region about the line y= -5.83315552448461
```

It worked here, but if Python's solve command does not work, you can use the nsolve command, though you will first want to plot the equation to get an estimated "starting guess". We illustrate this below.

In [11]:
```
matplotlib notebook
```

In [12]:
```
plot(Volc,(c,-10,0)) # We are given that c has to be negative.
```



Out[12]:
```
<sympy.plotting.plot.Plot at 0x174bb2ff0a0>
```

In [13]:
```
# It appears from the plot that Volume = 5 when c is about -6, so we use that as our starting guess
cval=nsolve(Volc-5,c,-6)
print('Using nsolve, we see we need to rotate the region about the line y=',cval)
```

```
Using nsolve, we see we need to rotate the region about the line y= -5.83315552448461
```

We end with an optional section on how to GRAPH the solids of revolution. The mathematical details will be discussed in MATH 251, though looking at the formulas, you may be able to get an idea of why they work.

Basically, we will graph the boundary surfaces by using paramatric surfaces, which are like parametric curves, but require TWO input variables (since they are 2-dimensional objects!). The surface formed by rotating $y = f(x)$, $a <= x <= b$, about the x-axis is parametrized by:

$x = x$

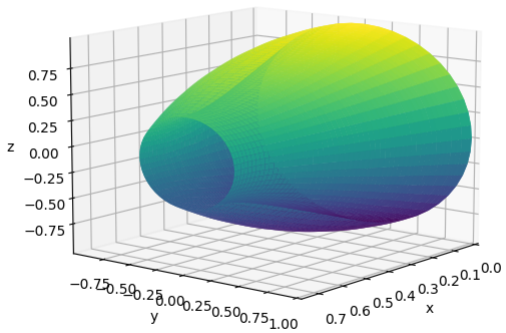
$y = f(x)^* \cos(\theta)$

$z = f(x)^* \sin(\theta)$

where x is in $[a, b]$ and θ is in $[0, 2\pi]$. Notice in the plot commands we imported at the start of the lab (re-imported below) that we have the command plot3d_parametric_surface. So let's use it to draw the solid in part b, by drawing the outer surface ($f(x)$ rotated) and the inner surface ($g(x)$ rotated)

In [14]:
```
matplotlib notebook
```

In [16]:
```
from sympy.plotting import (plot, plot_parametric,plot3d_parametric_surface, plot3d_parametric_line,plot3d)
x,theta=sp.symbols('x theta')
youter=(1-x**2)*sp.cos(theta)
zouter=(1-x**2)*sp.sin(theta)
yinner=(x**6-x+1)*sp.cos(theta)
zinner=(x**6-x+1)*sp.sin(theta)
plot3d_parametric_surface((x,youter,zouter,(x,a,b),(theta,0,2*sp.pi)),(x,yinner,zinner,(x,a,b),(theta,0,2*sp.pi)))
```



Out[16]:
```
<sympy.plotting.plot.Plot at 0x174bba11f40>
```

If you want to, you can execute the code in the block above and click on the plot to rotate it to see it from any perspective!

In [ ]:
```

```