In [1]:
```
from sympy import *
from sympy.plotting import (plot, plot_parametric)
```

INTEGRATION AND AREA IN PYTHON

In the last lab we reviewed the key symbolic commands used in 151: **simplify** (among other commands to rewrite an expression), **subs**, **solve**, and **plot**. This week we will look at definite and indefinite integrals in Python, then put all of these commands together to solve typical 152 problems you are doing now (as well as the entire first half of the course!)

Definite and indefinite integrals are done in Python using the **integrate** command.

Example 1: Compute the indefinite integral of f(x) = x^3 * sqrt(x^2 + 4)

Example 2: Compute the integral of f from x=0 to x=2.

In [2]:
```
x=symbols('x')  # Remember the symbols command allows x to be defined as ju
st "x"
f=x**3*sqrt(x**2+4)  # Recall the ** for exponents. Also notice sqrt for th
e square root
F=integrate(f,x)
print('The indefinite integral of f is',F)
```

```
The indefinite integral of f is x**4*sqrt(x**2 + 4)/5 + 4*x**2*sqrt(x**2 +
4)/15 - 32*sqrt(x**2 + 4)/15
```

Notice two things: 1) The answer Python gives is close, but would NOT get full credit (do you notice why?) 2) If we try to rewrite the answer (using **factor**, **expand**, or **simpify**), sometimes we are able to see how to solve this by hand using u-substitution, although this doesn't work here.

In [3]:
```
print('The indefinite integral of f is actually',F.simplify(),'+ C')
print('which can also be written as',F.factor(),'+ C')
print('or',F.expand(),'+ C')
```

```
The indefinite integral of f is actually sqrt(x**2 + 4)*(3*x**4 + 4*x**2 -
32)/15 + C
which can also be written as (x**2 + 4)**(3/2)*(3*x**2 - 8)/15 + C
or x**4*sqrt(x**2 + 4)/5 + 4*x**2*sqrt(x**2 + 4)/15 - 32*sqrt(x**2 + 4)/15
+ C
```

Definite integrals have a similar syntax in Python, but notice how the boundaries are presented. You may have learned that the triplet in parentheses is called a *tuple* in Python.

In [4]:
```
integrate(f,(x,0,2))
```

Out[4]:

$$\frac{64}{15} + \frac{64\sqrt{2}}{15}$$

Since our boundaries were integers, Python returned an exact answer. We can get a decimal approximation by either converting the answer to floating point (using **evalf**) or by writing our bounds as floating point numbers

In [5]:
```
#Converting the answer using evalf
print('The definite integral from x=0 to x=2 is',(integrate(f,(x,0,2))).evalf())
#Converting the bounds to floating point type
print('The definite integral from x=0 to x=2 is',integrate(f,(x,0.0,2.0)))
```

```
The definite integral from x=0 to x=2 is 10.3006445327919
The definite integral from x=0 to x=2 is 10.3006445327919
```

We are now ready to put all the tools together to solve our standard calculus problems.

**IMPORTANT!!!!!! Just because we are using a computer does NOT mean we don't have to think!!! In fact, the easiest way to approach most of these Python calculus problems is to ask yourself:

1) How would I solve this problem by hand?

2) What Python command(s) will allow me to do each step?

Example 3: Find the area between the graph of f from problems 1-2 and the graph of g(x)=x^5-x

So how do you find area? From what you have learned in class, there are three steps:

STEP 1) Graph the functions (Key Python command: **plot**)

STEP 2) Find the point(s) of intersection (Key Python command: **solve** or **nsolve**)

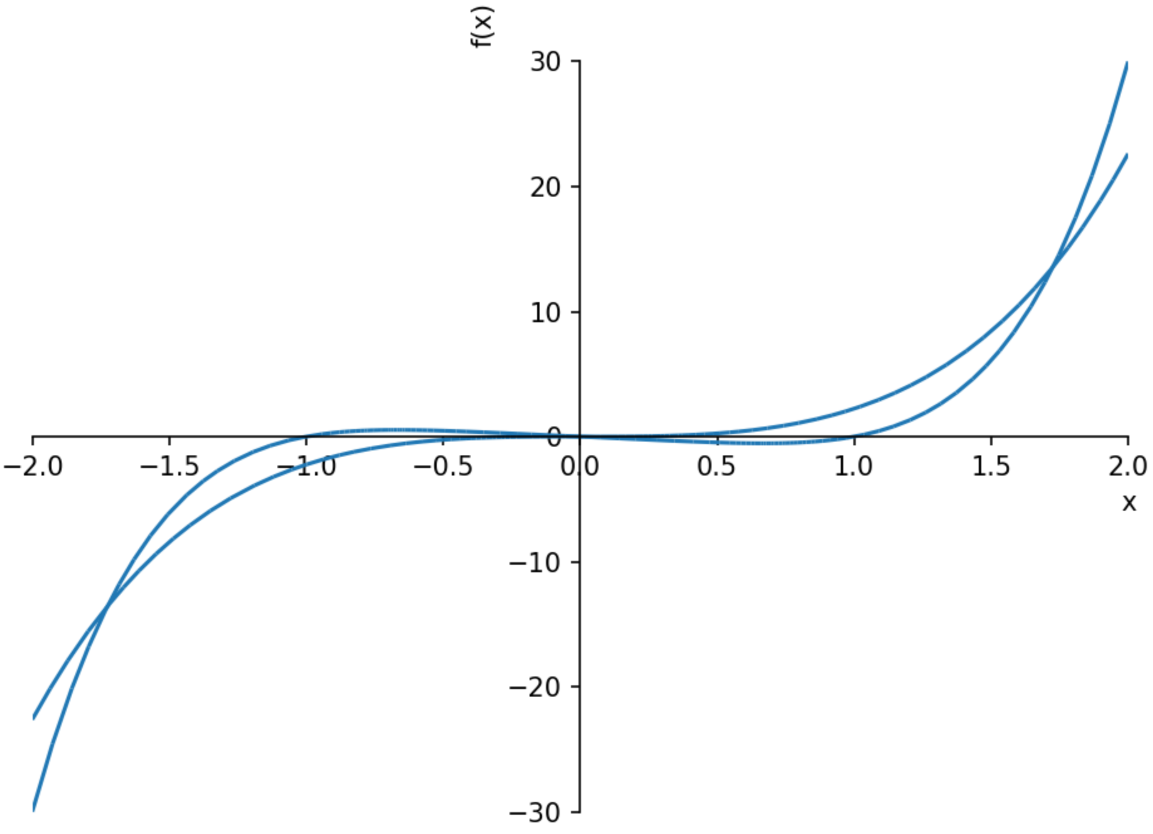
STEP 3) Integrate Top - Bottom over the appropriate intervals (Key Python command: **integrate**)

Remember to display the graph correctly, you need the line below (which should ALWAYS be by itself for some reason)

In [6]:
```
matplotlib notebook
```

In [7]:

```
# STEP 1: Graph the functions  Use tuples to graph both functions in the sa
me plot command
g=x**5-x
plot((f,(x,-2,2)),(g,(x,-2,2))) #If you are not given a domain, experiment
 to make sure you find ALL area between the curves!
```



Out[7]:

```
<sympy.plotting.plot.Plot at 0xc0327d0>
```

In [8]:

```
# STEP 2: find the points of intersection
inter=solve(f-g,x)
print(inter) # We get 0 and a bunch of nasty stuff.  Let's use nsolve and t
he graph.  Solutions are close to +/-1.5
a=nsolve(f-g,x,-1.5)
print('a=',a)
c=0 # from the solve command above
b=nsolve(f-g,x,1.5)
print('b=',b) # NOTE: it is always a good idea to view your intermediate co
mputations, at least while you are working on it!
```

```
[0, -I*sqrt(-sqrt(6)*sqrt(-16*sqrt(3)*cos(atan(sqrt(49071)/9)/3) - 75*sqrt
(3)/sqrt(32*sqrt(3)*cos(atan(sqrt(49071)/9)/3) + 51) + 51)/12 - 1/4 + sqrt
(3)*sqrt(32*sqrt(3)*cos(atan(sqrt(49071)/9)/3) + 51)/12), I*sqrt(-sqrt(6)*s
qrt(-16*sqrt(3)*cos(atan(sqrt(49071)/9)/3) - 75*sqrt(3)/sqrt(32*sqrt(3)*cos
(atan(sqrt(49071)/9)/3) + 51) + 51)/12 - 1/4 + sqrt(3)*sqrt(32*sqrt(3)*cos
(atan(sqrt(49071)/9)/3) + 51)/12), -sqrt(1/4 + sqrt(6)*sqrt(-16*sqrt(3)*cos
(atan(sqrt(49071)/9)/3) + 75*sqrt(3)/sqrt(32*sqrt(3)*cos(atan(sqrt(49071)/
9)/3) + 51) + 51)/12 + sqrt(3)*sqrt(32*sqrt(3)*cos(atan(sqrt(49071)/9)/3) +
51)/12), sqrt(1/4 + sqrt(6)*sqrt(-16*sqrt(3)*cos(atan(sqrt(49071)/9)/3) + 7
5*sqrt(3)/sqrt(32*sqrt(3)*cos(atan(sqrt(49071)/9)/3) + 51) + 51)/12 + sqrt
(3)*sqrt(32*sqrt(3)*cos(atan(sqrt(49071)/9)/3) + 51)/12)]
a= -1.72549570442084
b= 1.72549570442084
```

You may have noticed that the solutions are opposites. You may have also noticed from the graph that we have odd functions, so the regions are symmetric about the origin and therefore have the same area. So our area is two times the area from 0 to b. To figure out Top-Bottom, you can graph the functions in different colors (see the help documentation for **plot** for how to do this), or notice that one curve dips below the x-axis on this interval. It cannot be f since on the domain [0,b], x^3 and the square root are always positive. So our top curve is f and our bottom curve is g.

In [9]:

```
# STEP 3: integrate top - bottom (then multiply by 2 because of the symmetr
y)
Area=2*integrate(f-g,(x,0,b))
print('The area between the curves is',Area)
```

```
The area between the curves is 5.00341066745982
```

In [ ]:

```

```