In [1]:
```
import sympy as sp
from sympy.plotting import (plot, plot_parametric,plot3d_parametric_surface, plot3d_parametric_line,plot3d)
```

FINDING TAYLOR SERIES IN PYTHON

Recall the Taylor Series of a function centered at x=a is a Power Series:

sum(n=0..oo, c_n(x-a)^n)

Where c_n = (nth derivative of f)(a)/n!

Recall the key to solving most problems in Python is knowing the steps to solve them by hand and finding the appropriate commands to perform those steps. So to find a Taylor Series using this definition:

1. Find the first several derivatives at x=a
2. Determine a pattern in these values and generalize the formula for an arbitrary nth derivative
3. Replace this formula in the formula for c_n, then replace c_n in the Power Series.

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

EXAMPLE:

Use the definition to find the Taylor Series of f(x) = (cos(x))^2 centered at x = pi/2

In [2]:
```
# Step 1: Find the first several derivatives at x=pi/2
x=sp.symbols('x')
f=(sp.cos(x))**2
# We can do everything at once using list comprehension!  Let's find the first 10 derivatives at x=pi/2
nthder_at_a=[sp.diff(f,x,i).subs(x,pi/2) for i in range(11)] # Remember: range(11) is from 0 to 10!!
print('The derivatives at x=pi/2 are',nthder_at_a)
```

```
The derivatives at x=pi/2 are [0, 0, 2, 0, -8, 0, 32, 0, -128, 0, 512]
```

So we notice that only the even derivatives (from n=2 on) are nonzero, so our series will involve even (2n) powers only. Further:

f''(pi/2) = 2^1

f(4)(pi/2) = -1*(2^3)

f(6)(pi/2) = 2^5

f(8)(pi/2) = -1*(2^7)

f(10)(pi/2) = 2^9

So we conclude that f(2n)(pi/2) = (-1)^(n-1)* (2^(2n-1)). Why (-1)^(n-1) and not (-1)^n? Just substitute the numbers (n=1,2,3,4,5) and you will see.

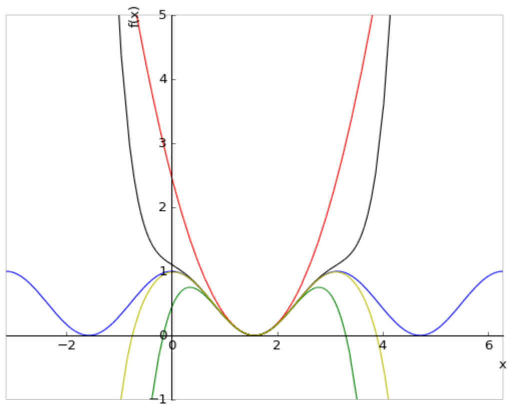
In [3]:
```
#Step 3: Replace the pattern above into the c_n formula, then c_n into the series
n=sp.symbols('n',integer=True)
cn=(-1)**(n-1)*(2**(2*n-1))/sp.factorial(2*n)  #Remember, only EVEN numbered terms are nonzero!
an=cn*(x-sp.pi/2)**(2*n)
print('The terms of the Taylor Series for f centered at x=a are',an)
```

```
The terms of the Taylor Series for f centered at x=a are (-1)**(n - 1)*2**(2*n - 1)*(x - pi/2)**(2*n)/factorial(2*n)
```

To verify this, let's plot the function in the domain [-pi, 2pi] and range [-1,5] along with the first few partial sums of the Taylor series (also called the *Taylor Polynomials*. Remember that the series starts with the second degree term, so we will start with n=1 (the "first even number"). We'll plot them in different colors to distinguish them here, though if you are printing your assignment in black & white, that won't be of use. (NOTE: the "summation" command is a much simpler way to get the partial sums of the series, as shown as an alternative below)

In [5]:
```
matplotlib notebook
```

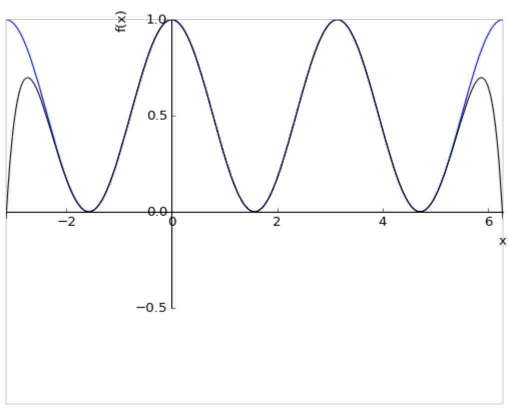
In [6]:
```
a1=an.subs(n,1) # only one term at first, so nothing to add here
a1to2=[an.subs({n:i}) for i in range(1,3)]  #NOTE: Since we are just looking at consecutive partial sums, we
S2=sum(a1to2)
a1to3=[an.subs({n:i}) for i in range(1,4)]  # could just add the next term to the previous. But if you are
S3=sum(a1to3)
a1to4=[an.subs({n:i}) for i in range(1,5)]  # looking for, example, s3, s6, and s9, then this strategy is better
S4=sum(a1to4)
# Alternate strategy: use summation: NOTE that the range INCLUDES the boundary point, unlike above!
S2=sp.summation(an,[n,1,2])
S3=sp.summation(an,[n,1,3])
S4=sp.summation(an,[n,1,4])
# Since we want to plot them in different colors, we need to do one at a time and use the .extend() command
fplot=plot(f,(x,-pi,2*pi),ylim=[-1,5],show=False)
a1plot=plot(a1,(x,-pi,2*pi),ylim=[-1,5],line_color='r',show=False)
a2plot=plot(S2,(x,-pi,2*pi),ylim=[-1,5],line_color='g',show=False)
a3plot=plot(S3,(x,-pi,2*pi),ylim=[-1,5],line_color='k',show=False)
a4plot=plot(S4,(x,-pi,2*pi),ylim=[-1,5],line_color='y',show=False)
fplot.extend(a1plot)
fplot.extend(a2plot)
fplot.extend(a3plot)
fplot.extend(a4plot)
fplot.show()
```



Notice that the higher the degree, the larger the domain where the graph resembles f(x). In fact, let's show the function (in blue) and the n=10 partial sum (in black: degree 20 since we are counting even powers only)

In [7]:
```
matplotlib notebook
```

In [8]:
```
S10=sp.summation(an,[n,1,10])
fplot=plot(f,(x,-pi,2*pi),ylim=[-1,1],show=False)
Tplot=plot(S10,(x,-pi,2*pi),ylim=[-1,1],line_color='k',show=False)
fplot.extend(Tplot)
fplot.show()
```



In [ ]: