Square Well Ground State (Weinberg problem 1.1)

This is an ipython notebook. Lectures about Python, useful both for beginners and experts, can be found at http://scipy-lectures.github.io.

I recommend installing the Anaconda distribution. Make sure not to pay for it! Click Anaconda Academic License; it should be free for those with edu e-mail addresses.

Open the notebook by (1) copying this file into a directory, (2) in that directory typing

    ipython notebook --pylab inline

and (3) selecting the notebook.

Like Mathematica, you type in commands and then hit 'Shift Return' to execute them.

In [ ]:
```
a = 3.
```

Functions like Psi0 are created using 'def'. Note that indents are important in Python, but ipython does them for you properly in many cases. Note that 'a' in the function doesn't have to be passed separately: Python will use the outer 'namespace' if it's not defined in the function.

In [ ]:
```
def Psi0(x):
    return (1/sqrt(...)) * cos(... / (...))
Psi0(0.)
```

We import 'scipy.integrate.quad' to do numerical integrals. Typing ?quad gives a documentation window at the bottom of the browser.

In [ ]:
```
from scipy.integrate import quad
?quad
```

The integration package quad(func,a,b) demands that we define the integrand as a separate function.

In [ ]:
```
def Psi0Sq(x):
    return ...**2
Psi0Sq(0.)
```

It returns two numbers: the integral, and an error estimate.

In [ ]:
```
quad(Psi0Sq,-3.,3.)
```

To get the integral, we select the first entry of the answer, which in Python has index zero.

In [ ]:
```
quad(Psi0Sq,-3.,3.)[0]
```

We now define the trial wavefunction, without the right normalization:

In [ ]:
```
def PsiUnnormalized(x):
    return (a**2-...)
```

We can find the normalization by integrating PsiUnnormalized:

In [ ]:
```
def PsiUnnormalizedSq(x):
    return ...
InverseNormSq = quad(..., ..., ...)[0]
norm = ...
norm
```

In [ ]:
```
def Psi(x):
    """Trial wavefunction"""
    return (norm * ... * PsiUnnormalized(x))
```

Testing that the normalization is right...

In [ ]:
```
def PsiSq(x):
   ...
quad(PsiSq,...)
```

To plot, define a range of x's using 'linspace'. Python automatically will evaluate functions like Psi and Psi0 on all the x's at once (much more efficiently than it does loops).

In [ ]:
```
?linspace
```

In [ ]:
```
xCoarse = linspace(-a,a,7)
print xCoarse
print Psi0(xCoarse)
x = linspace(-a,a,100)
```

Plot takes the x values followed by the y values. Psi0 and Psi should look rather similar.

In [ ]:
```
plot(x, Psi0(x))
plot(x, Psi(x))
```

Define the integrand for the overlap of Psi0 and Psi:

In [ ]:
```
def OverlapIntegrand(x):
    return ...
```

In [ ]:
```
overlap = quad(OverlapIntegrand,-3,3)[0]
```

Finally, find the probability of Psi being in the ground state

In [ ]:
```
ProbGroundState = ...
ProbGroundState
```