###Let's start off with a definition for discrete uniform distributions.

Definition: A random variable X has a discrete uniform distribution if each of the n values in its range: x1,x2,x3....xn has equal probability:

$$f(x_i) = 1/n$$

Now let's use python to show a simple example!

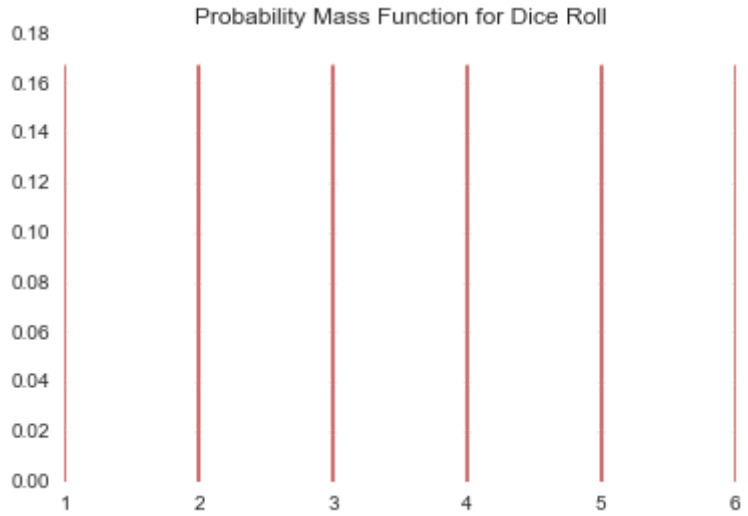
First the imports:

In [3]:
```
# Import all the usual imports from the Python for Data Analysis and Visualizatio
import numpy as np
from numpy.random import randn
import pandas as pd
from scipy import stats
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from __future__ import division
%matplotlib inline
```

Now let's set up a dice roll and plot the distribution using seaborn before we go use matplotlib by itself.

In [31]:
```
# How about a roll of a dice?

# Let's check out the Probability Mass function!

# Each number
roll_options = [1,2,3,4,5,6]

# Total probability space
tprob = 1

# Each roll has same odds of appearing (on a fair die at least)
prob_roll = tprob / len(roll_options)

# Plot using seaborn rugplot (note this is not really a rugplot), setting height
uni_plot = sns.rugplot(roll_options,height=prob_roll,c='indianred')

# Set Title
uni_plot.set_title('Probability Mass Function for Dice Roll')
```

Out[31]:
```
<matplotlib.text.Text at 0x1b8cbf28>
```



We can see in the above example that the f(x) value on the plot is just equal to 1/(Total Possible Outcomes)

So what's the mean and variance of this uniform distribution?

The mean is simply the max and min value divided by two, just like the mean of two numbers.

$$\mu = (b + a)/2$$

With a variance of:

$$\sigma^2 = \frac{(b-a+1)^2 - 1}{12}$$

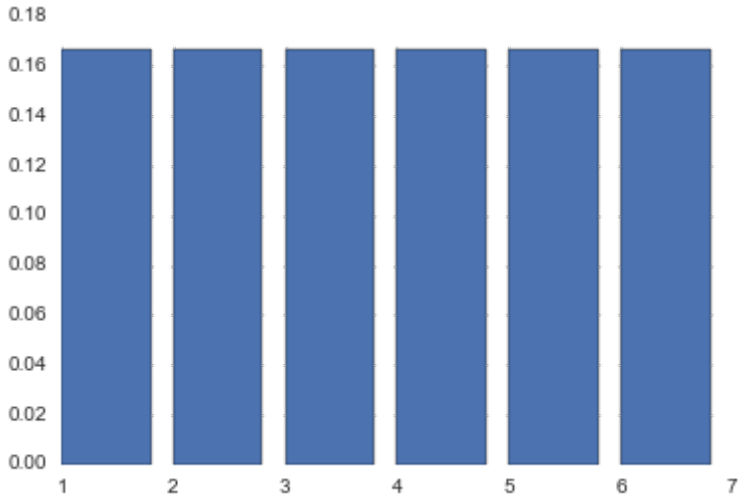
Now let's see how to automatically create a Discrete Uniform Distribution using Scipy.

In [32]:
```
# Imports
from scipy.stats import randint

#Set up a low and high boundary for dice roll ( go to 7 since index starts at 0)
low,high = 1,7

# Get mean and variance
mean,var = randint.stats(low,high)

print 'The mean is %2.1f' %mean
```

```
The mean is 3.5
```

In [33]:
```
# Now we can make a simple bar plot
plt.bar(roll_options,randint.pmf(roll_options,low,high))
```

Out[33]:
```
<Container object of 6 artists>
```



#####That's basically it for a discrete uniform distribution, check out the rest of the reading below if you're still interested.

Example of real world use: German Tank Problem

So now that we know some information about the uniform discrete distribution function, how about we use it to solve a problem?

In this case we'll solve the famous German Tank Problem.

For background, first read the wikipedia page: http://en.wikipedia.org/wiki/German_tank_problem (http://en.wikipedia.org/wiki/German_tank_problem)

Excerpt from Wikipedia:

"In the statistical theory of estimation, the problem of estimating the maximum of a discrete uniform distribution from sampling without replacement is known in English as the German tank problem, due to its application in World War II to the estimation of the number of German tanks. Estimating the population maximum based on a single sample yields divergent results, while the estimation based on multiple samples is an instructive practical estimation question whose answer is simple but not obvious."

After reading the Wikipedia article, check out the following code for an example Python workout of the example problem.

Using a Minimum-variance unbiased estimator we obtain the population max is equal to :

$$Population\max = sample\max + \frac{sample\max}{sample\ size} - 1$$

If we for instance captured 5 tanks with the serial numbers 3,7,11,16 then we know the max observed serial number was m=16. This is our sample max with a sample size of 5 tanks. Plugging into the MVUE results in:

In [21]:
```
tank_estimate = 16 + (16/5) - 1
tank_estimate
```

Out[21]:
```
18.2
```

For a Bayseian Approach:

In [22]:
```
m=16
k=5
tank_b_estimate = (m-1)*( (k-1)/ ( k-2) )
tank_b_estimate
```

Out[22]:
```
20.0
```

Remember, this is still missing the STD

In [ ]:
```

```