In [2]:

```
# Get thinkdsp.py

import os

if not os.path.exists('thinkdsp.py'):
    !wget https://github.com/AllenDowney/ThinkDSP/raw/master/code/thinkdsp.py
```

```
--2022-05-12 10:52:41--  https://github.com/AllenDowney/ThinkDSP/raw/master/code/thinkdsp
.py
Resolving github.com (github.com)... 192.30.255.113
Connecting to github.com (github.com)|192.30.255.113|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/AllenDowney/ThinkDSP/master/code/thinkdsp.py
[following]
--2022-05-12 10:52:41--  https://raw.githubusercontent.com/AllenDowney/ThinkDSP/master/co
de/thinkdsp.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.1
99.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443.
.. connected.
HTTP request sent, awaiting response... 200 OK
Length: 48687 (48K) [text/plain]
Saving to: ‘thinkdsp.py’

thinkdsp.py         100%[===================>]  47.55K  --.-KB/s    in 0.005s

2022-05-12 10:52:41 (9.49 MB/s) - ‘thinkdsp.py’ saved [48687/48687]
```

In [16]:

```
import numpy as np
import scipy.signal
import matplotlib.pyplot as plt

from thinkdsp import decorate
from thinkdsp import Wave
```

In [4]:

```
# suppress scientific notation for small numbers
np.set_printoptions(precision=3, suppress=True)
```

HW02

In [11]:

```
def zero_pad(array, n):
    """Extends an array with zeros.

    array: NumPy array
    n: length of result

    returns: new NumPy array
    """
    res = np.zeros(n)
    res[:len(array)] = array
    return res
```

In [13]:

```
from thinkdsp import SquareSignal

def plot_filter(M=11, std=2):
```

```python
    signal = SquareSignal(freq=440)
    wave = signal.make_wave(duration=1, framerate=44100)
    spectrum = wave.make_spectrum()

    gaussian = scipy.signal.gaussian(M=M, std=std)
    gaussian /= sum(gaussian)

    ys = np.convolve(wave.ys, gaussian, mode='same')
    smooth =  Wave(ys, framerate=wave.framerate)
    spectrum2 = smooth.make_spectrum()

    # plot the ratio of the original and smoothed spectrum
    amps = spectrum.amps
    amps2 = spectrum2.amps
    ratio = amps2 / amps
    ratio[amps<560] = 0

    # plot the same ratio along with the FFT of the window
    padded =  zero_pad(gaussian, len(wave))
    dft_gaussian = np.fft.rfft(padded)

    plt.plot(np.abs(dft_gaussian), color='gray', label='Gaussian filter')
    plt.plot(ratio, label='amplitude ratio')

    decorate(xlabel='Frequency (Hz)', ylabel='Amplitude ratio')
    plt.show()
```

In [14]:

```python
from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets

slider = widgets.IntSlider(min=2, max=100, value=11)
slider2 = widgets.FloatSlider(min=0, max=20, value=2)
interact(plot_filter, M=slider, std=slider2);
```

在以上的程式碼，可以在執行時拖動**std**變量，觀察**fft(dft)**結果。

在**gaussian distribution**中，**std**代表數據的分散程度。當**std**越高，窗函數越集中，在卷積運算時平滑程度較差。當**std**越低時，窗函數分散，平滑效果越好，呈現的結果與**smoothing window**較為相近，同樣會出現旁瓣（**sidelobes**）的問題。

In [25]:

```python
from thinkdsp import SquareSignal

def plot_filter(M=11, std=2):
    signal = SquareSignal(freq=440)
    wave = signal.make_wave(duration=1, framerate=44100)
    spectrum = wave.make_spectrum()

    gaussian = scipy.signal.gaussian(M=M, std=std)
    gaussian /= sum(gaussian)

    ys = np.convolve(wave.ys, gaussian, mode='same')
    smooth =  Wave(ys, framerate=wave.framerate)
    spectrum2 = smooth.make_spectrum()

    # plot the ratio of the original and smoothed spectrum
    amps = spectrum.amps
    amps2 = spectrum2.amps
    ratio = amps2 / amps
    ratio[amps<560] = 0

    # plot the same ratio along with the FFT of the window
    padded =  zero_pad(gaussian, len(wave))
    dft_gaussian = np.fft.rfft(padded)

    plt.plot(np.abs(dft_gaussian), color='gray', label='Gaussian filter')
    plt.yscale("log")
    plt.plot(ratio, label='amplitude ratio')
```

```
    decorate(xlabel='Frequency (Hz)', ylabel='Amplitude ratio')
    plt.show()

from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets

slider = widgets.IntSlider(min=2, max=100, value=11)
slider2 = widgets.FloatSlider(min=0, max=20, value=2)
interact(plot_filter, M=slider, std=slider2);
```

以上的程式碼是**y**軸在**log**維度下的結果。

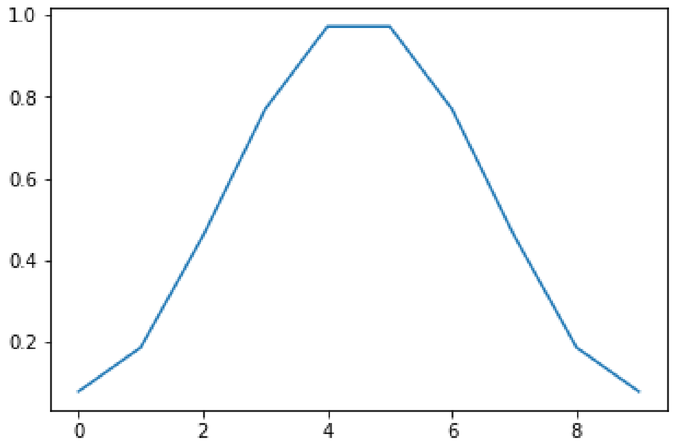
HW03

In [17]:

```
def hamming_window(M=10):
  window = scipy.signal.hamming(M)
  return window

my_window = hamming_window()
plt.plot(my_window)
```

Out[17]:

```
[<matplotlib.lines.Line2D at 0x7fd25c597990>]
```



In [23]:

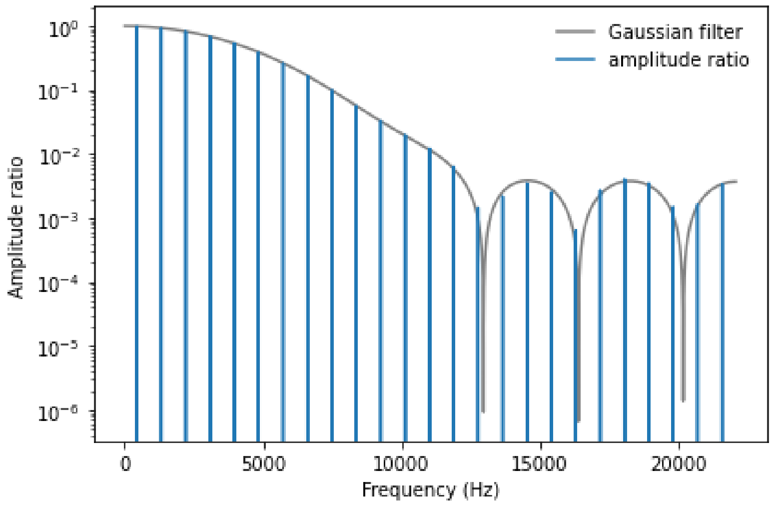
```
from thinkdsp import SquareSignal

def plot_filter2(M=11):
    signal = SquareSignal(freq=440)
    wave = signal.make_wave(duration=1, framerate=44100)
    spectrum = wave.make_spectrum()

    hamming = hamming_window(M)

    ys = np.convolve(wave.ys, hamming, mode='same')
    smooth =  Wave(ys, framerate=wave.framerate)
    spectrum2 = smooth.make_spectrum()

    # plot the ratio of the original and smoothed spectrum
    amps = spectrum.amps
    amps2 = spectrum2.amps
    ratio = amps2 / amps
    ratio[amps<560] = 0

    # plot the same ratio along with the FFT of the window
    padded =  zero_pad(hamming, len(wave))
    dft_gaussian = np.fft.rfft(padded)
```

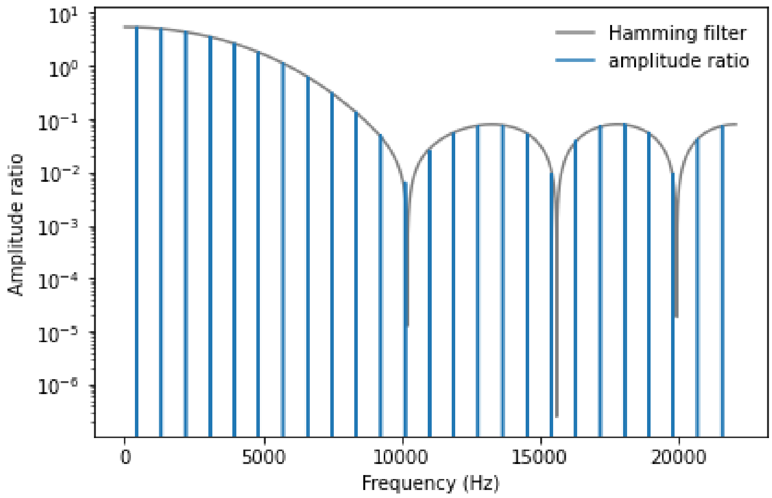
```
    plt.plot(np.abs(dft_gaussian), color='gray', label='Hamming filter')
    plt.yscale("log")
    plt.plot(ratio, label='amplitude ratio')

    decorate(xlabel='Frequency (Hz)', ylabel='Amplitude ratio')
    plt.show()
```

In [24]:

```
from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets

slider = widgets.IntSlider(min=2, max=100, value=11)
interact(plot_filter2, M=slider);
```

以下為兩種結果在**y=log**維度下的比對。





在**M(**窗大小相同**)**的情況，我看不出差別啊 **qwq**