在 [1]:

```
# 獲取 thinkdsp.py

導入 操作系統

如果 不是 操作系統。路徑。存在（'thinkdsp.py'）：
    ！wget  https://github.com _ _ _ _ com / AllenDowney / ThinkDSP / raw / master / code /
```

在 [2]:

```
 將numpy 導入為 np
導入 matplotlib。pyplot 作為 plt

從 thinkdsp 導入 裝飾
```

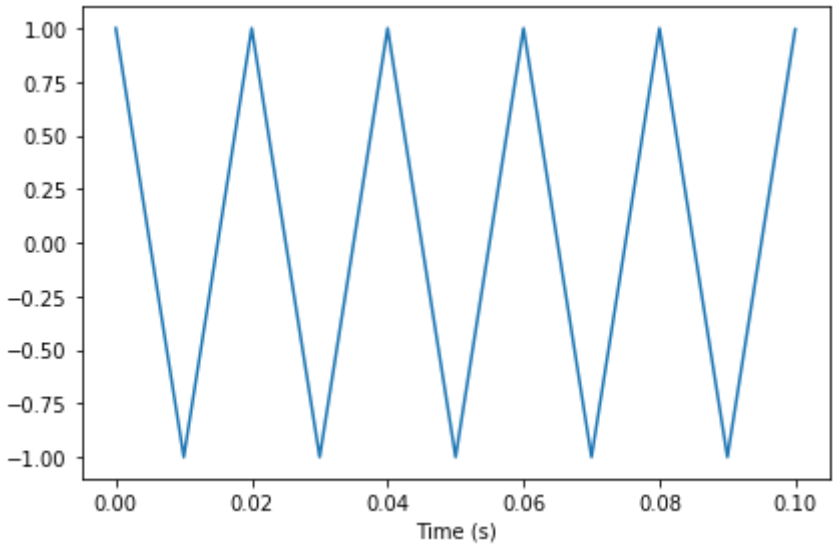
練習 1

本練習的目的是探索信號的影響 `diff` 和 `differentiate` 對信號的影響。創建一個三角波並繪製它。應用 `diff` 運算符並繪製結果。計算三角波的頻譜，應用 `differentiate` 並繪製結果。將頻譜轉換回波並繪製它。`diff` 這波的效果和 `differentiate` 對這波的影響有區別嗎？

*解決方案：*這是三角波。

在 [3]:

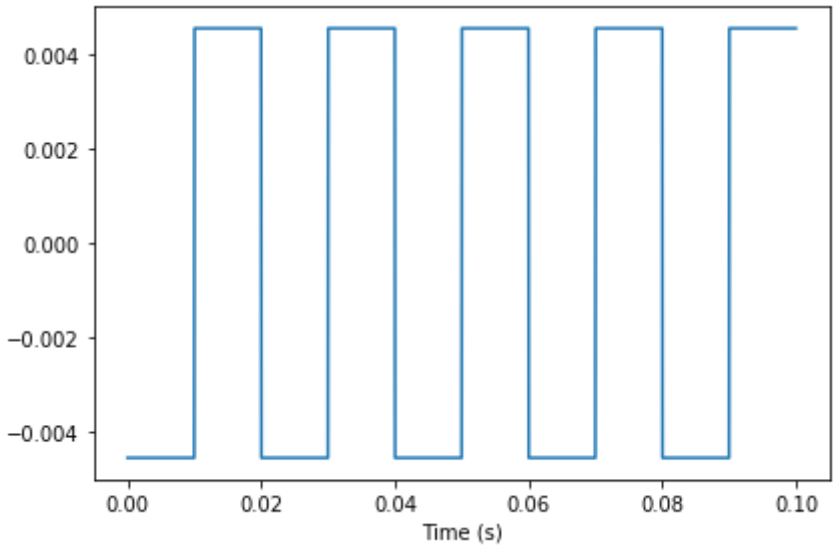
```
從 thinkdsp 導入 三角信號

in_wave  = 三角信號（頻率= 50）。make_wave（持續時間= 0.1 ,幀率= 44100 ）
in_wave。情節（ ）
裝飾( xlabel = '時間 (s)' )
```



三角波的差異是方波，這解釋了為什麼方波中的諧波會像$1 / f$ 與三角波相比，三角波像$1/f^2$.

在 [4]:

```
out_wave  =  in_wave。差異 ( )
出波。情節 ( )
裝飾( xlabel = '時間 (s)' )
```

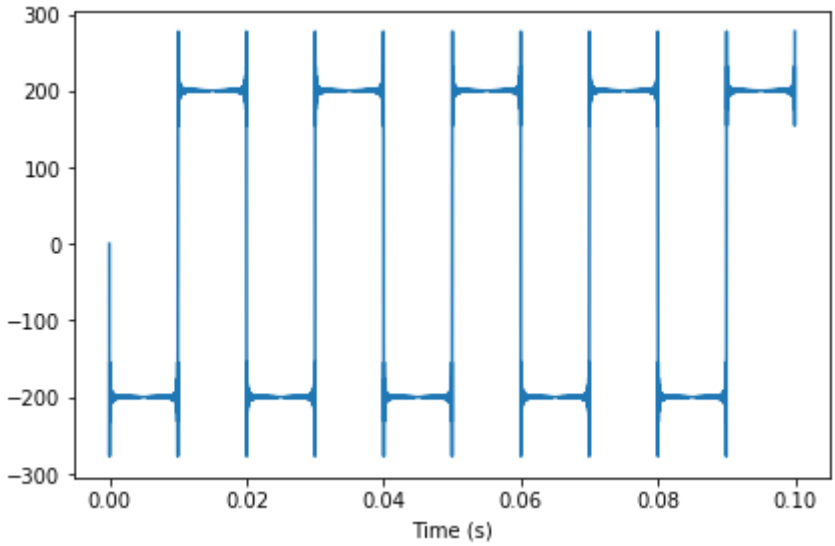


當我們採用光譜導數時，我們會在不連續點周圍“振鈴”：https://en.wikipedia.org/wiki/Ringing_(signal) (https://en.wikipedia.org/wiki/Ringing_(signal))

從數學上講，問題在於三角波的導數在三角形的點上是不確定的。

在 [5]:

```
out_wave2  =  in_wave。製造光譜 ( ) 。區分 ( ) 。make_wave ()
out_wave2。情節 ( )
裝飾( xlabel = '時間 (s)' )
```



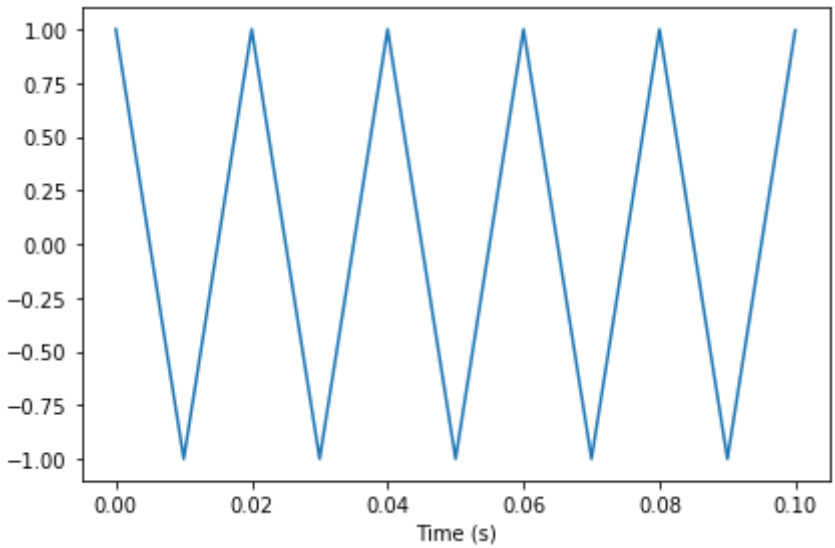
練習 1

本練習的目的是探索信號的影響 `diff` 和 `differentiate` 對信號的影響。創建一個三角波並繪製它。應用 `diff` 運算符並繪製結果。計算三角波的頻譜，應用 `differentiate` 並繪製結果。將頻譜轉換回波並繪製它。 `diff` 這波的效果和 `differentiate` 對這波的影響有區別嗎？

*解決方案：*這是三角波。

在 [3]:

```
從 thinkdsp 導入 三角信號

in_wave  = 三角信號（頻率= 50）。make_wave（持續時間= 0.1 ,幀率= 44100 ）
in_wave。情節（ ）
裝飾( xlabel = '時間 (s)' )
```



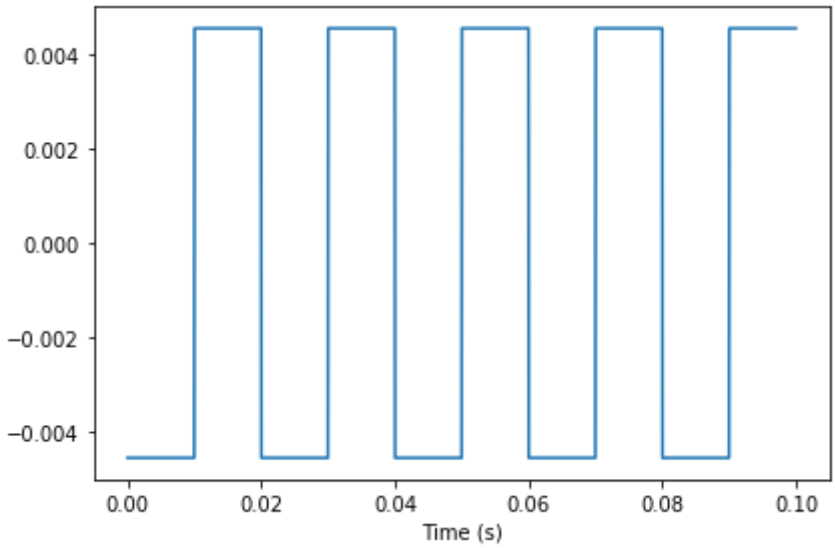
三角波的差異是方波，這解釋了為什麼方波中的諧波會像$1 / f$ 與三角波相比，三角波像$1/f^2$.

當我們採用光譜導數時，我們會在不連續點周圍“振鈴”：https://en.wikipedia.org/wiki/Ringing_(signal) (https://en.wikipedia.org/wiki/Ringing_(signal))

從數學上講，問題在於三角波的導數在三角形的點上是不確定的。

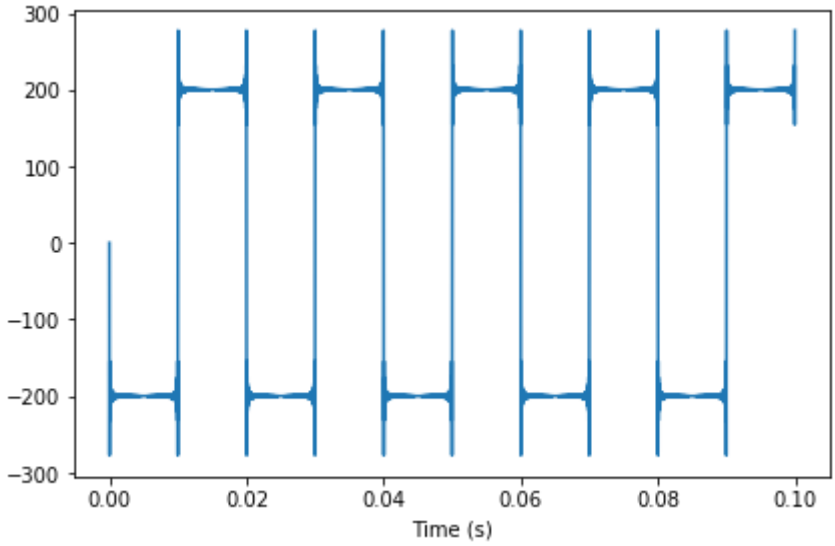
在 [4]:

```
out_wave = in_wave.diff()
out_wave.plot()
decorate(xlabel='Time (s)')
```



在 [5]:

```
out_wave2  =  in_wave。製造光譜 ( ) 。區分 ( ) 。make_wave ()
out_wave2。情節 ( )
裝飾( xlabel = '時間 (s)' )
```



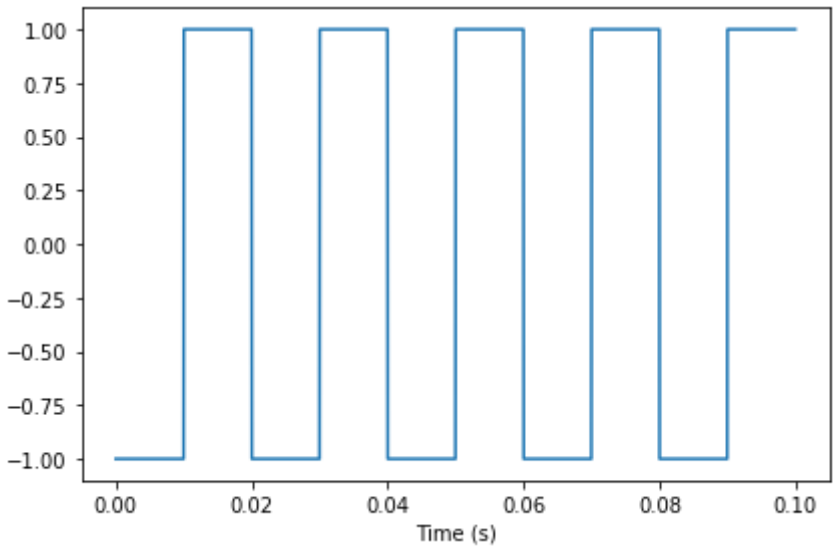
練習 2

本練習的目的是探索信號的影響 `cumsum` 和 `integrate` 對信號的影響。創建一個方波並繪製它。應用 `cumsum` 運算符並繪製結果。計算方波的頻譜，應用 `integrate` 並繪製結果。將頻譜轉換回波並繪製它。 `cumsum` 這波的效果和 `integrate` 對這波的影響有區別嗎？

*解決方案：*這是方波。

在 [6]:

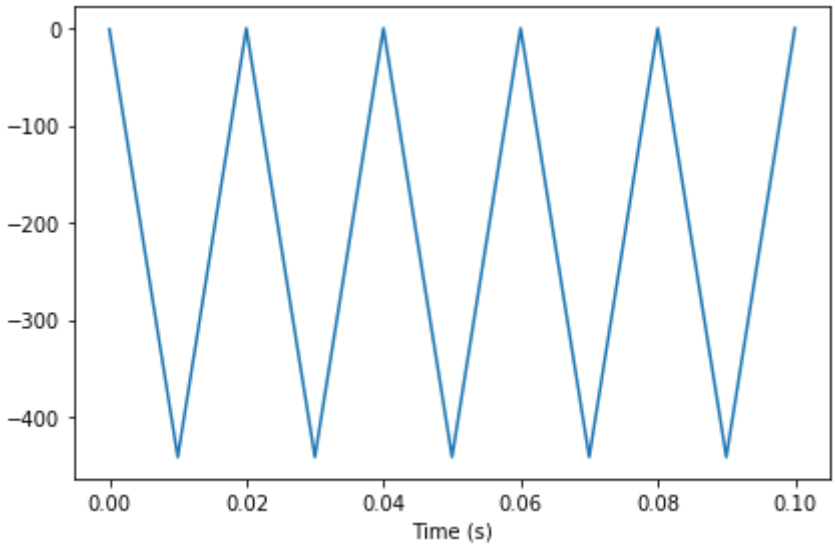
```
從 thinkdsp 導入 SquareSignal

in_wave  =  SquareSignal (頻率= 50 )。make_wave (持續時間= 0.1 ,幀率= 44100 )
in_wave。情節 ( )
裝飾( xlabel = '時間 (s)' )
```



方波的累積和是三角波。在之前的練習之後，這應該不足為奇。

在 [7]:

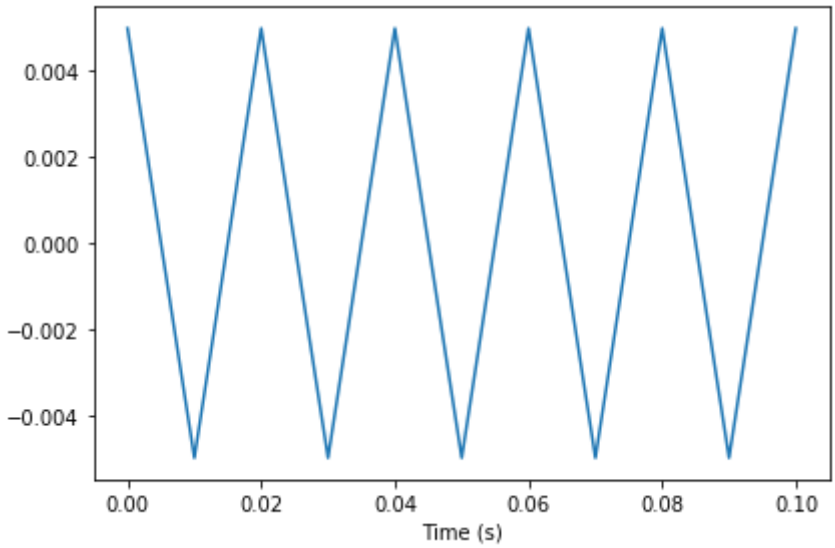
```
out_wave  =  in_wave。累積()
出波。情節 ( )
裝飾( xlabel = '時間 (s)' )
```



頻譜積分也是三角波，儘管幅度非常不同。

在 [8]:

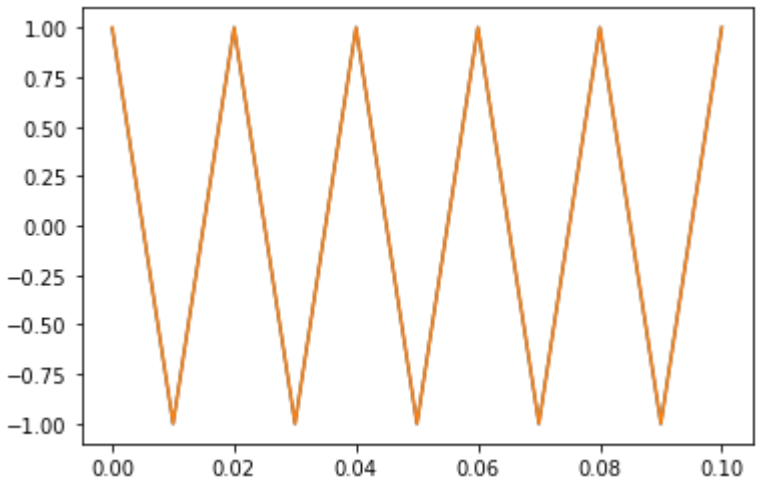
```
頻譜 =  in_wave。製造光譜 ( ) 。整合()
頻譜。hs [ 0 ] =  0
out_wave2  = 頻譜。make_wave ()
out_wave2。情節 ( )
裝飾( xlabel = '時間 (s)' )
```



如果我們對這兩個波進行去偏和歸一化，它們在視覺上是相似的。

在 [9]:

```
出波。不偏不倚()
出波。規範化 ( )
out_wave2。規範化 ( )
出波。情節 ( )
out_wave2。情節 ( )
```



它們在數字上相似，但只有大約 3 位數的精度。

在 [11]:

```
出波。最大差異( out_wave2 )
```

輸出[11]：

```
0.0045351473922902175
```

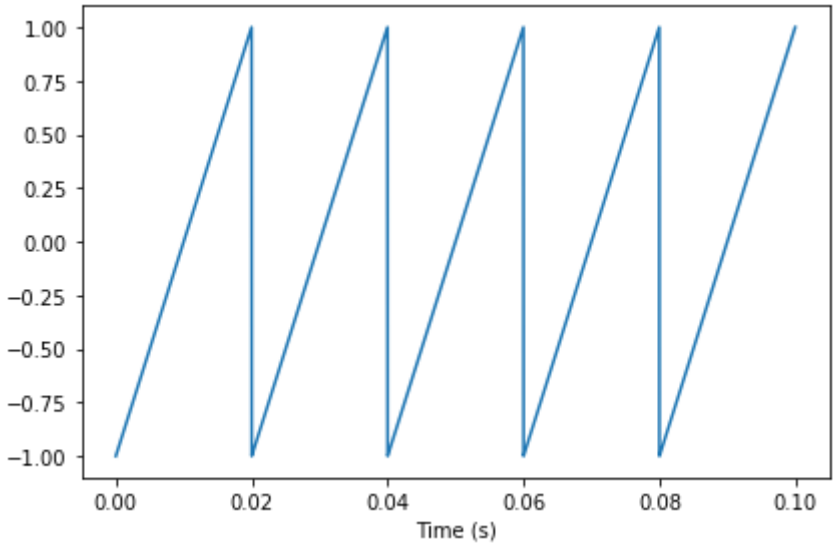
練習 3

本練習的目標是探索兩次積分的效果。創建一個鋸齒波，計算其頻譜，然後應用 integrate 兩次。繪製產生的波及其頻譜。波的數學形式是什麼？為什麼它類似於正弦曲線？

這是鋸齒。

在 [12]:

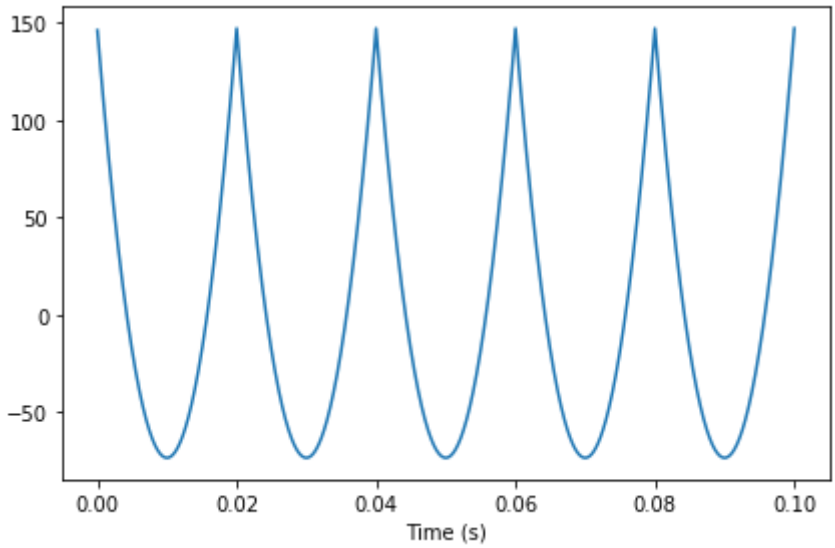
```
從 thinkdsp 導入 鋸齒信號

in_wave  = 鋸齒信號（頻率= 50）。make_wave (持續時間= 0.1 ,幀率= 44100 )
in_wave。情節（ ）
裝飾( xlabel = '時間 (s)' )
```



鋸齒的第一個累積和是拋物線：

在 [13]:

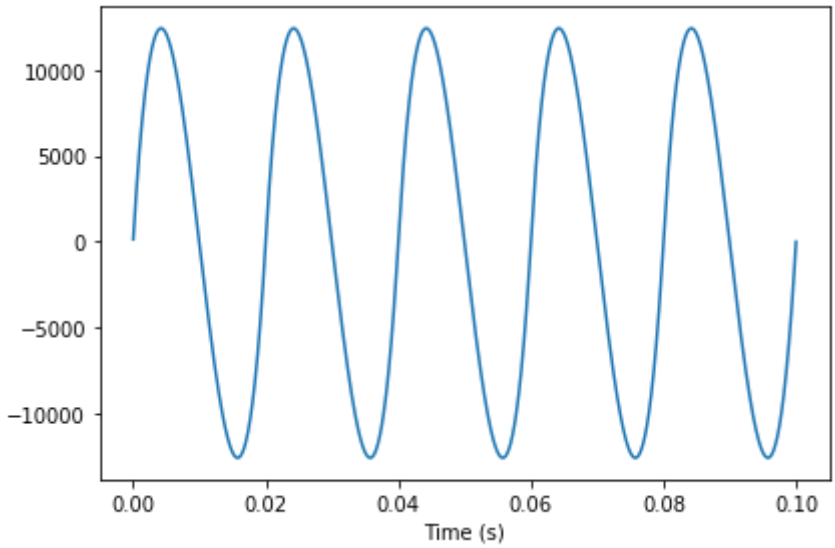
```
out_wave  =  in_wave。累積()
出波。不偏不倚()
出波。情節 ( )
裝飾( xlabel = '時間 (s)' )
```



第二個累積和是一條三次曲線：

在 [14]:

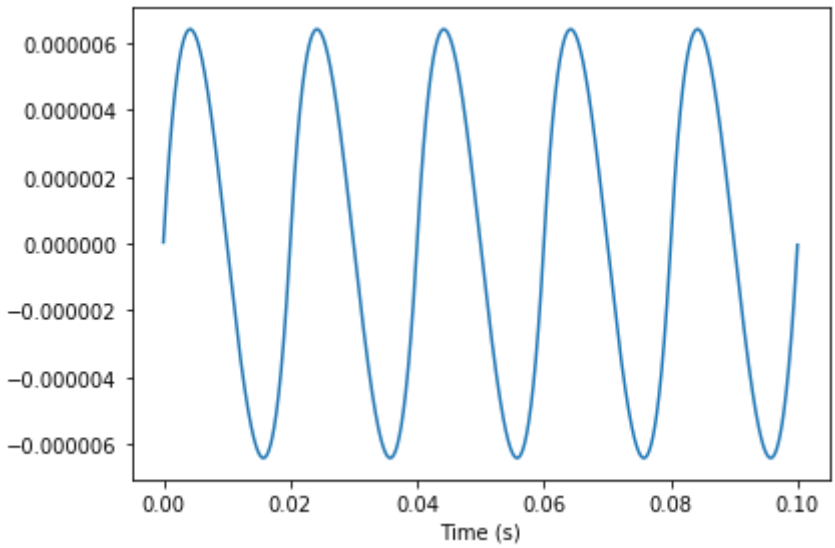
```
out_wave  =  out_wave。累積()
出波。情節 ( )
裝飾( xlabel = '時間 (s)' )
```



積分兩次也會產生三次曲線。

在 [15]:

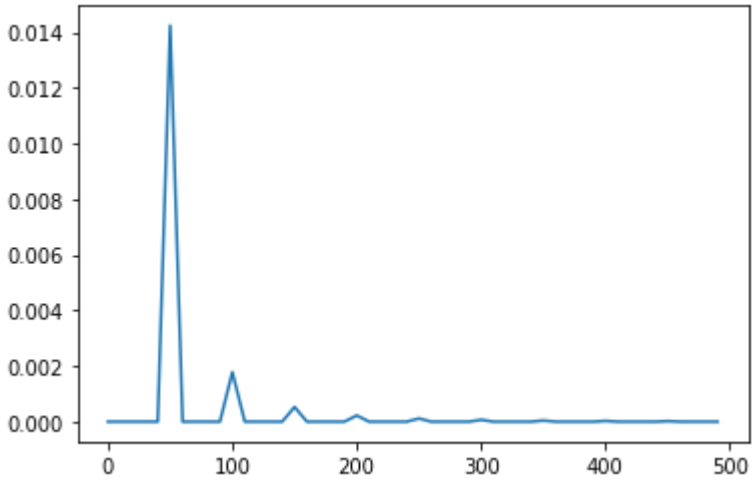
```
頻譜 =  in_wave。製造光譜 ( ) 。整合 ( ) 。整合()
頻譜。hs [ 0 ] =  0
out_wave2  = 頻譜。make_wave ()
out_wave2。情節 ( )
裝飾( xlabel = '時間 (s)' )
```



此時，結果看起來越來越像正弦曲線。原因是積分就像一個低通濾波器。至此，我們已經過濾掉了除基本之外的幾乎所有內容，如下圖所示：

在 [16]:

```
out_wave2。製造光譜（ ）。情節（高= 500）
```



練習 4

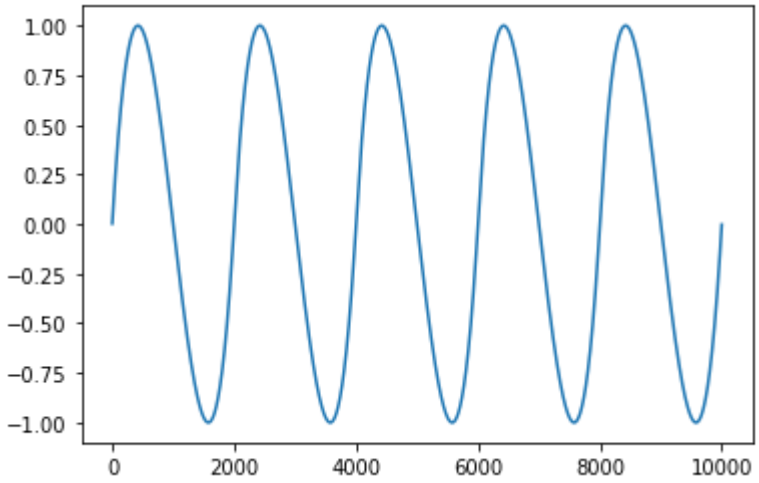
本練習的目的是探索二階差分和二階導數的影響。創建一個 `CubicSignal` ，它在 中定義 `thinkdsp` 。 `diff` 通過應用兩次計算第二個差異。結果是什麼樣子的。 `differentiate` 通過應用兩次計算二階導數。結果看起來一樣嗎？

繪製對應於二階差分和二階導數的濾波器並進行比較。提示：為了獲得相同比例的過濾器，請使用幀速率為 1 的波形。

*解決方案：*這是三次信號

在 [17]:

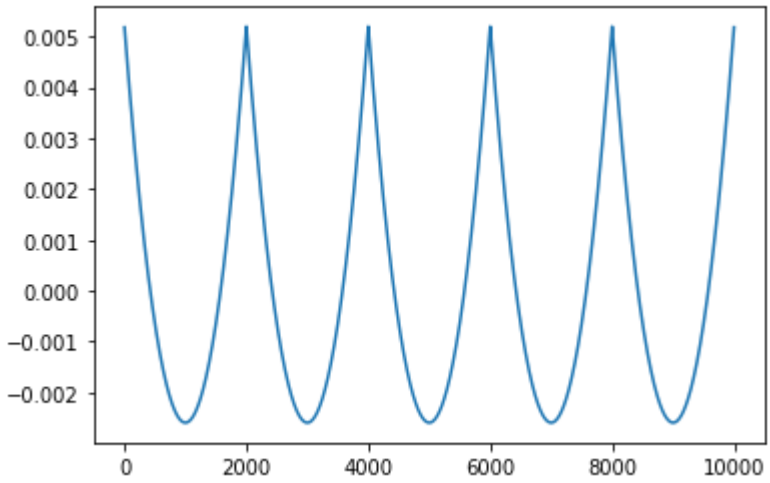
```
從 thinkdsp 導入 CubicSignal

in_wave  =  CubicSignal (頻率= 0.0005 )。make_wave (持續時間= 10000 ,幀率= 1 )
in_wave。情節 ( )
```



第一個區別是拋物線，第二個區別是鋸齒波（到目前為止沒有意外）：

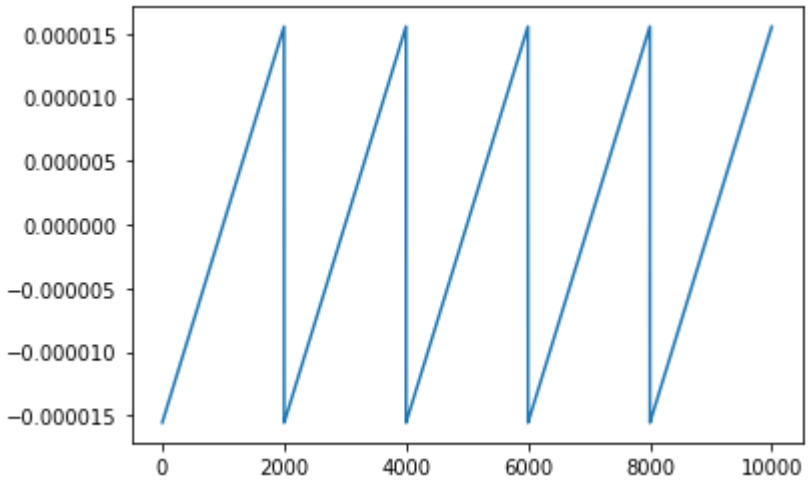
在 [18]:

```
out_wave  =  in_wave。差異 ( )
出波。情節 ( )
```



在 [19]:

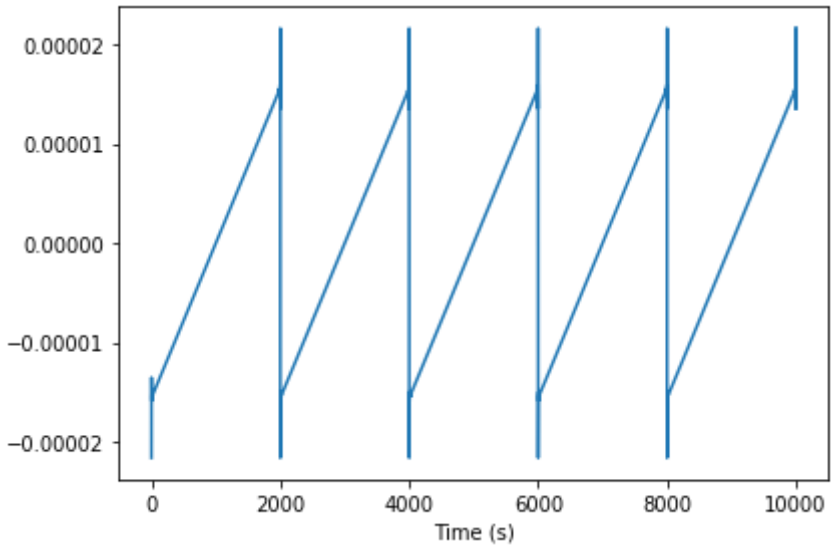
```
out_wave  =  out_wave。差異( )
出波。情節( )
```



當我們區分兩次時，我們會得到一個帶有一些振鈴的鋸齒。同樣，問題是拋物線信號的導數在這些點上是未定義的。

在 [20]:

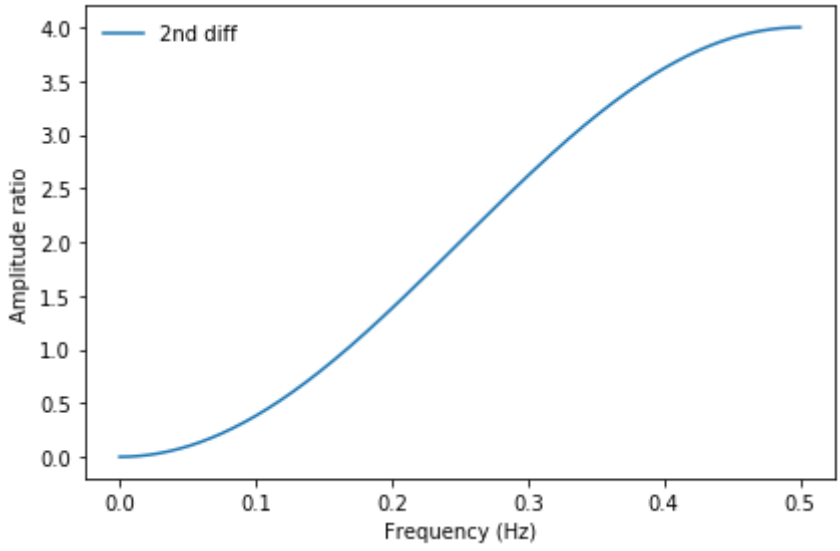
```
頻譜 =  in_wave。製造光譜 ( ) 。區分 ( ) 。區分()
out_wave2  = 頻譜。make_wave ()
out_wave2。情節 ( )
裝飾( xlabel = '時間 (s)' )
```



第二個差異的窗口是-1, 2, -1。通過計算窗口的 DFT，我們可以找到對應的濾波器。

在 [21]:

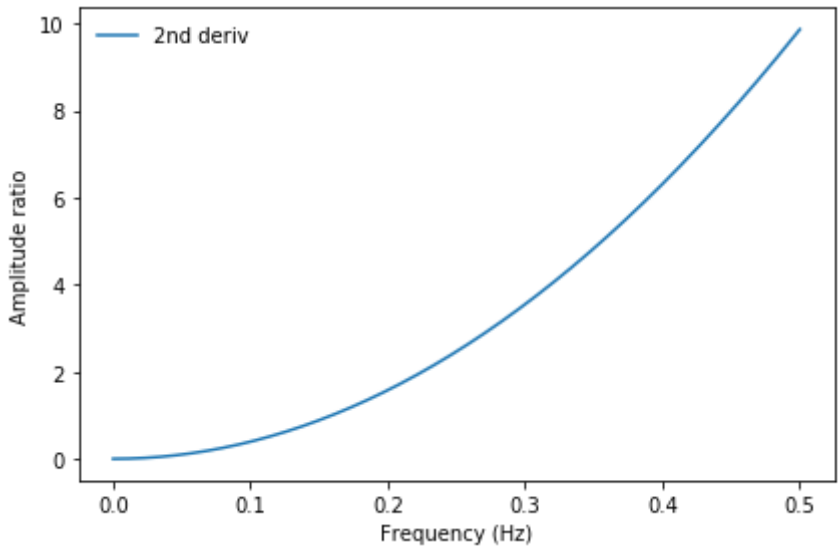
```
從 thinkdsp 導入 zero_pad
從 thinkdsp 導入 Wave

diff_window  =  np。數組([ - 1.0 , 2.0 , - 1.0 ])
填充 =  zero_pad ( diff_window , len ( in_wave ))
diff_wave  =  Wave ( 填充，幀率= in_wave。幀率 )
diff_filter  =  diff_wave。make_spectrum ()
diff_filter。情節 ( 標籤= '第二個差異' )

裝飾( xlabel = '頻率 (Hz)' ,
                ylabel = '幅度比' )
```



而對於二階導數，我們可以通過計算一階導數的濾波器並對其進行平方來找到對應的濾波器。

在 [22]:

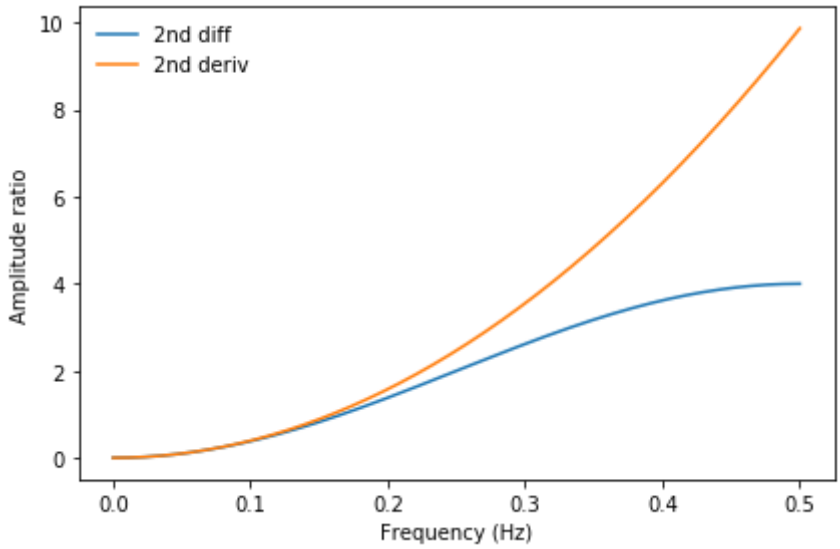
```
PI2  =  np。圓周率 *  2

deriv_filter  =  in_wave。make_spectrum ()
派生過濾器。hs  = ( PI2  *  1j  *  deriv_filter . fs ) ** 2
派生過濾器。情節 ( 標籤= '2nd deriv' )

裝飾( xlabel = '頻率 (Hz)' ,
                ylabel = '幅度比' )
```



這是兩個過濾器在相同比例下的樣子：

在 [23]:

```
diff_filter。情節 ( 標籤= '第二個差異' )
派生過濾器。情節 ( 標籤= '2nd deriv' )

裝飾( xlabel = '頻率 (Hz)' ,
                ylabel = '幅度比' )
```



兩者都是放大最高頻率分量的高通濾波器。二階導數是拋物線的，因此它放大了最高頻率。二階差分是二階導數的良好近似，僅在最低頻率處，然後它顯著偏離。

在 [ ]:

```
```