test

7 апреля 2018 г.

1 My first Jupyter Notebook

In [2]:
```
from __future__ import print_function, division

%matplotlib inline

import thinkdsp
import thinkplot
import matplotlib.pyplot as plt

import numpy as np

from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets
from IPython.display import display
```

Для обработки цифровых сигналов воспользуемся готовым пакетом ThinkX и встроенными в него модулями ThinkDSP и ThinkPlot. Для представления сигнала в модуле thinkdsp.py есть класс signal, задающий Python-представление математической функции, и от которого наследуется класс для представления синусоидального сигнала - Sinusoid. Для создания синусоидального сигнала определена функция SinSignal, в тело которой передаются три параметра: freq - частота в Герцах, amp - амлитуда в относительных единицах, offset - фазовый сдвиг в радианах.

In [3]:
```
sin_sig1 = thinkdsp.SinSignal(freq=50, amp=0.5, offset=0)
sin_sig2 = thinkdsp.SinSignal(freq=100, amp=0.7, offset=0)
sin_sig3 = thinkdsp.SinSignal(freq=250, amp=0.8, offset=0)
sin_sig4 = thinkdsp.SinSignal(freq=500, amp=1, offset=0)
```

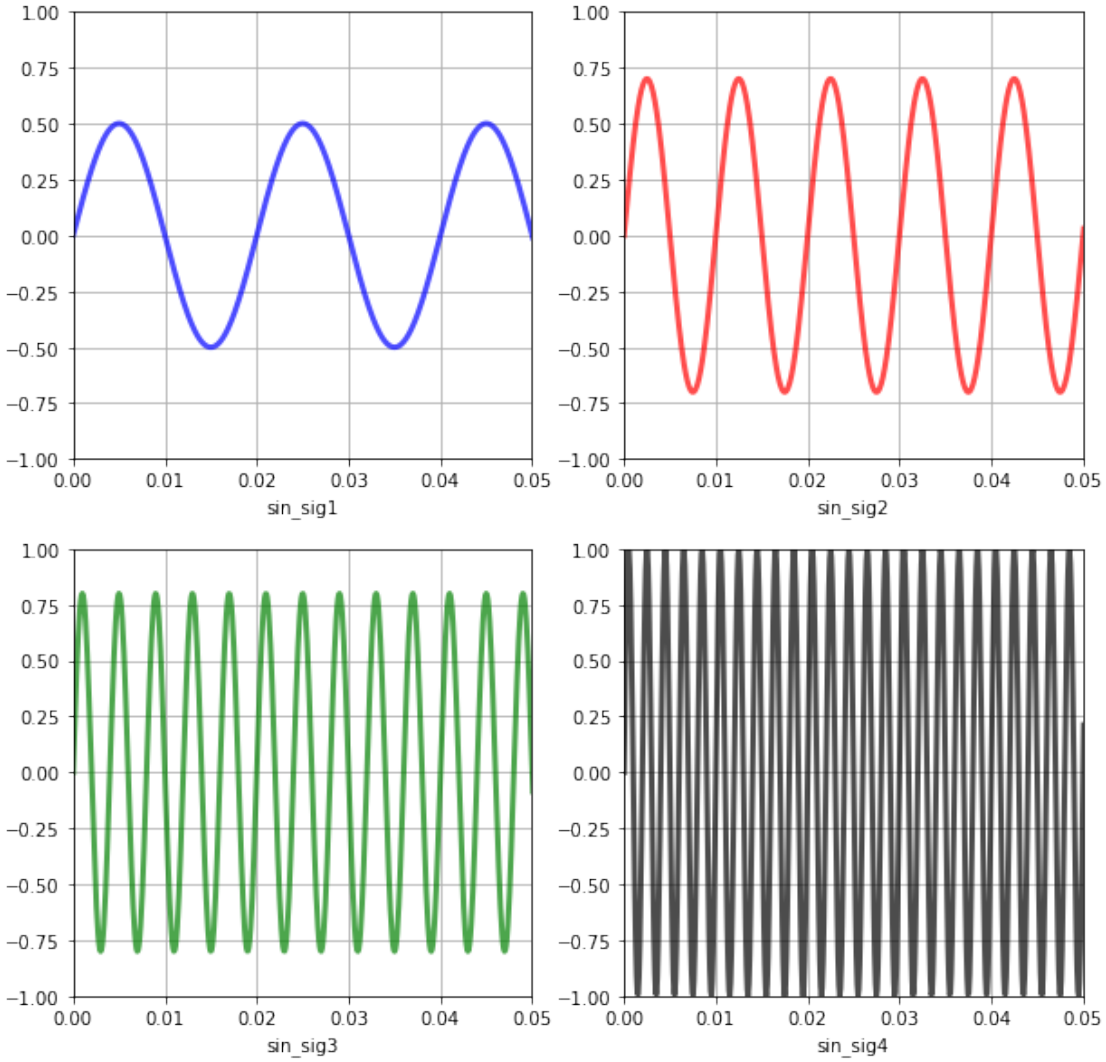
Для обработки сигнала в thinkdsp.py определён класс wave. Wave - это сигнал, обрабатываемый в последовательности моментов времени. Каждый момент врмени называется кадром (frame). Для создания экземпляра wave из существующего объекта сигнала, в классе signal определён метод make_wave с соответсвующими входными параметрами: duration - длина wave в секундах, start - время старта в секундах, framerate - число кадров (выборок) в секунду(по умолчанию устанавливается значение 11025 кадров в секунду - стандртная часта выборки, используемая в звуковых файлах разного формата). Wave поддерживает метод plot, использующий pyplot.

In [4]:
```
thinkplot.preplot(num=4, rows=2, cols=2)
plt.grid(True)
plt.axis([0, 0.05, -1, 1])
plt.xlabel('sin_sig1')
wave1 = sin_sig1.make_wave(framerate=11025)
wave1.plot(color='blue')

thinkplot.subplot(2)
plt.grid(True)
plt.axis([0, 0.05, -1, 1])
plt.xlabel('sin_sig2')
wave2 = sin_sig2.make_wave(framerate=11025)
wave2.plot(color='red')

thinkplot.subplot(3)
plt.grid(True)
plt.axis([0, 0.05, -1, 1])
plt.xlabel('sin_sig3')
wave3 = sin_sig3.make_wave(framerate=11025)
wave3.plot(color='green')

thinkplot.subplot(4)
plt.grid(True)
plt.axis([0, 0.05, -1, 1])
plt.xlabel('sin_sig4')
wave4 = sin_sig4.make_wave(framerate=11025)
wave4.plot(color='black')
```



Для представления спектра сигнала в модуле thinkdsp.py определён класс Spectrum, поддерживающий функцию plot. Класс wave поддерживает make_spectrum, возвращающий спектр.

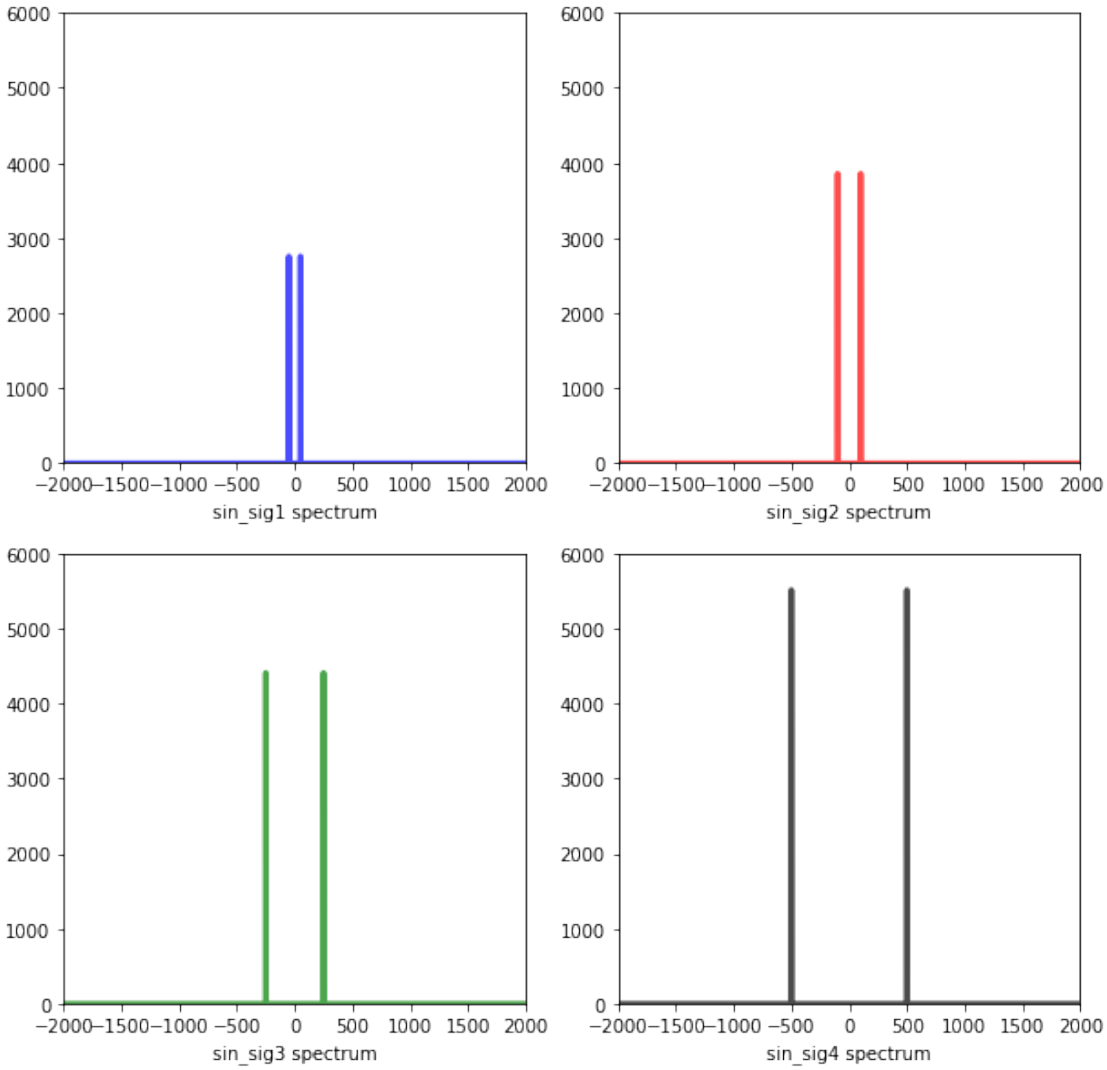
```
In [5]: thinkplot.preplot(num=4, rows=2, cols=2)
        plt.axis([-2000, 2000, 0, 6000])
        plt.xlabel('sin_sig1 spectrum')
        wave1.make_spectrum(full=True).plot(color='blue')

        thinkplot.subplot(2)
        plt.axis([-2000, 2000, 0, 6000])
        plt.xlabel('sin_sig2 spectrum')
        wave2.make_spectrum(full=True).plot(color='red')

        thinkplot.subplot(3)
```

```
plt.axis([-2000, 2000, 0, 6000])
plt.xlabel('sin_sig3 spectrum')
wave3.make_spectrum(full=True).plot(color='green')

thinkplot.subplot(4)
plt.axis([-2000, 2000, 0, 6000])
plt.xlabel('sin_sig4 spectrum')
wave4.make_spectrum(full=True).plot(color='black')
```



Выше представлены спектры соответсвующих заданных синосуидальных сигналов. У синусоиды только одна частотная компонента, чем объясняется единственный пик в её спектре.

Для создания прямоугольного сигнала thinkdsp.py предоставляет функцию SquareSignal, принимающую на вход те же параметры что и SinSignal.

```
In [ ]: sq_sig1 = thinkdsp.SquareSignal(freq=50, amp=0.5, offset=0)
        sq_sig2 = thinkdsp.SquareSignal(freq=100, amp=0.7, offset=0)
```

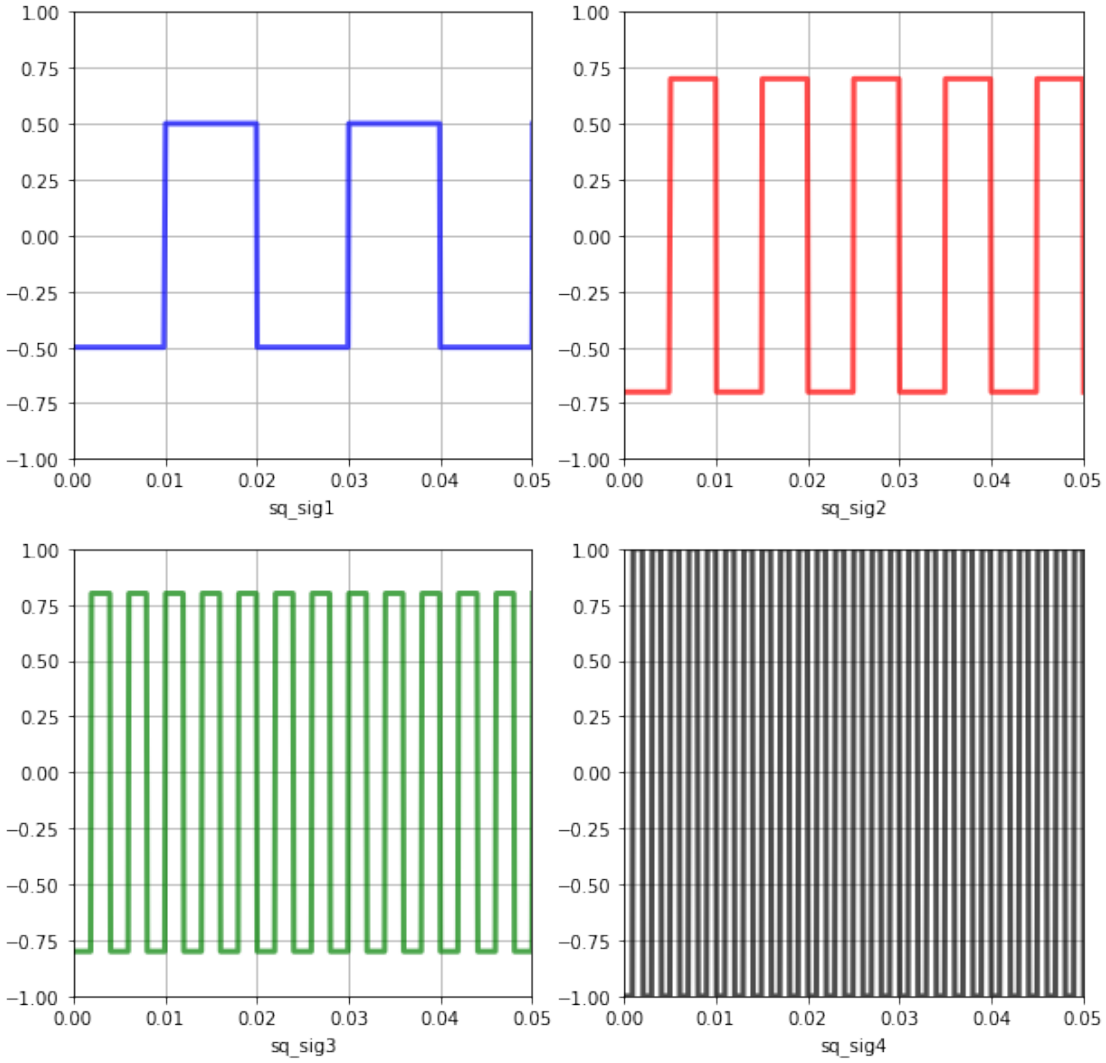
```
        sq_sig3 = thinkdsp.SquareSignal(freq=250, amp=0.8, offset=0)
        sq_sig4 = thinkdsp.SquareSignal(freq=500, amp=1, offset=0)
```

In [8]:
```
        thinkplot.preplot(num=4, rows=2, cols= 2)
        plt.grid(True)
        plt.axis([0, 0.05, -1, 1])
        plt.xlabel('sq_sig1')
        sq_wave1 = sq_sig1.make_wave(framerate=11025)
        sq_wave1.plot(color='blue')

        thinkplot.subplot(2)
        plt.grid(True)
        plt.axis([0, 0.05, -1, 1])
        plt.xlabel('sq_sig2')
        sq_wave2 = sq_sig2.make_wave(framerate=11025)
        sq_wave2.plot(color='red')

        thinkplot.subplot(3)
        plt.grid(True)
        plt.axis([0, 0.05, -1, 1])
        plt.xlabel('sq_sig3')
        sq_wave3 = sq_sig3.make_wave(framerate=11025)
        sq_wave3.plot(color='green')

        thinkplot.subplot(4)
        plt.grid(True)
        plt.axis([0, 0.05, -1, 1])
        plt.xlabel('sq_sig4')
        sq_wave4 = sq_sig4.make_wave(framerate=11025)
        sq_wave4.plot(color='black')
```



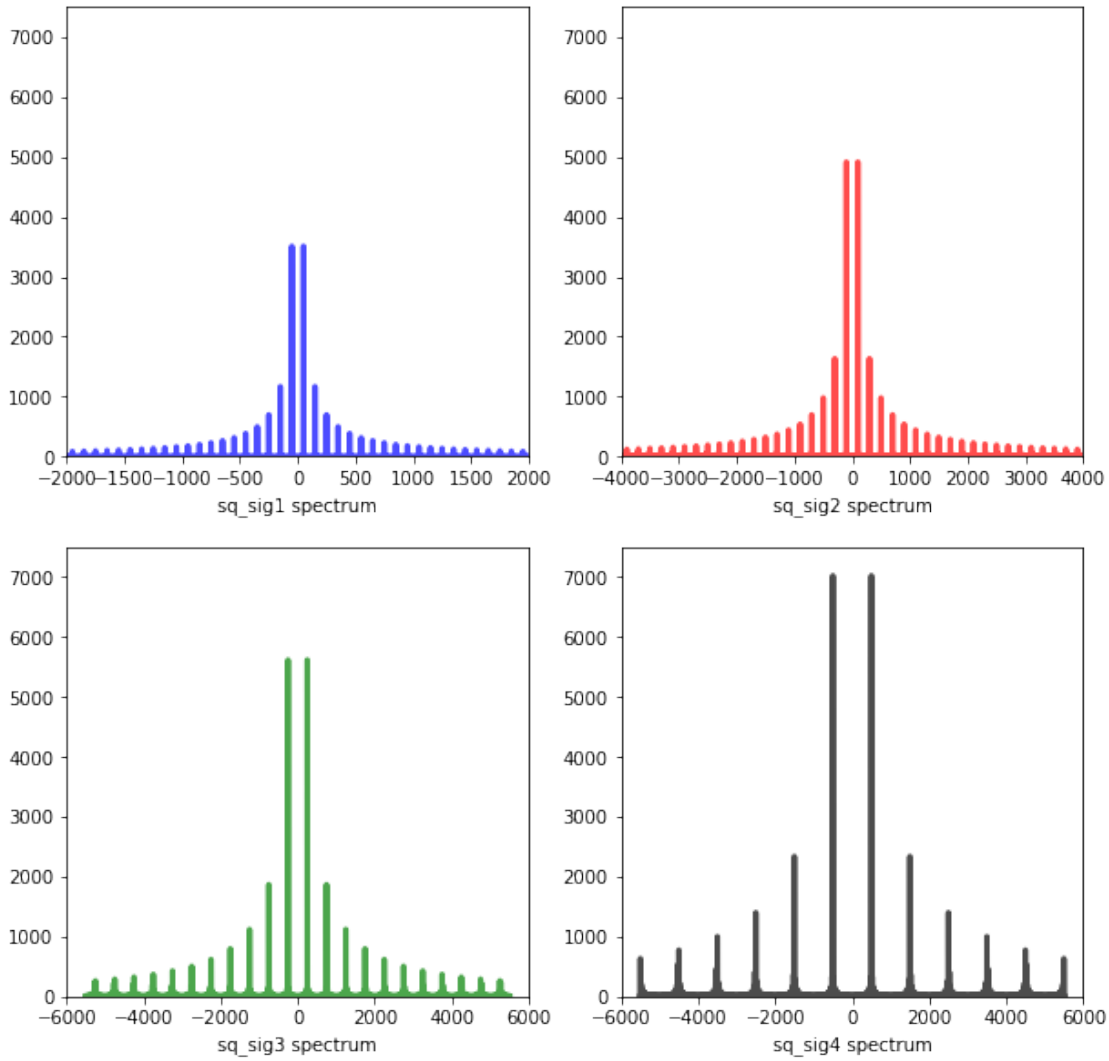

```
In [9]: thinkplot.preplot(num=4, rows=2, cols=2)
        plt.axis([-2000, 2000, 0, 7500])
        plt.xlabel('sq_sig1 spectrum')
        sq_spectre1 = sq_wave1.make_spectrum(True)
        sq_spectre1.plot(color='blue')

        thinkplot.subplot(2)
        plt.axis([-4000, 4000, 0, 7500])
        plt.xlabel('sq_sig2 spectrum')
        sq_spectre2 = sq_wave2.make_spectrum(True)
        sq_spectre2.plot(color='red')

        thinkplot.subplot(3)
        plt.axis([-6000, 6000, 0, 7500])
```

```
plt.xlabel('sq_sig3 spectrum')
sq_spectre3 = sq_wave3.make_spectrum(True)
sq_spectre3.plot(color='green')

thinkplot.subplot(4)
plt.axis([-6000, 6000, 0, 7500])
plt.xlabel('sq_sig4 spectrum')
sq_spectre4 = sq_wave4.make_spectrum(True)
sq_spectre4.plot(color='black')
```



На рисунках выше представлены графики спектров соответсвующих прямоугольных сигналов. Прямоугольный сигнал содержит только нечётные гармоники. Амлитуда гармоник спадает пропорционально частоте. С ростом частоты сигнала, растёт расстояние между пиками в его спектре.