Санкт-Петербургский государственный политехнический
университет Петра Великого.
**Высшая школа интеллектуальных сисем и
суперкомпьютерных технологий**

Лабораторная работа

Автокорреляция

Работу выполнила студентка:
_____ А. И.Луцкевич
«____» _____ 2021 г.

Преподаватель лабораторных
работ:
_____ Н. В.Богач
«___» _____ 2021 г.

Санкт-Петербург, 2021 г.

Суть работы 5.1:

Необходимо вычислить автокорреляцию для различных Lag и оценить высоты тона вокального чирпа

Изначально напишем две функции, которые помогут в выполнении этого задания - serial-corr и autocorr.

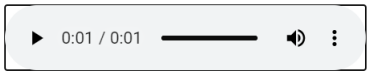
In [3]:
```
def autocorr(wave):
    lags = np.arange(len(wave.ys)//2)
    corrs = [serial_corr(wave, lag) for lag in lags]
    return lags, corrs
```

In [4]:
```
def serial_corr(wave, lag=1):
    n = len(wave)
    y1 = wave.ys[lag:]
    y2 = wave.ys[:n-lag]
    corr_mat = np.corrcoef(y1, y2)
    return corr_mat[0, 1]
```

Далее считаем чирп с записью голоса и прослушаем его.

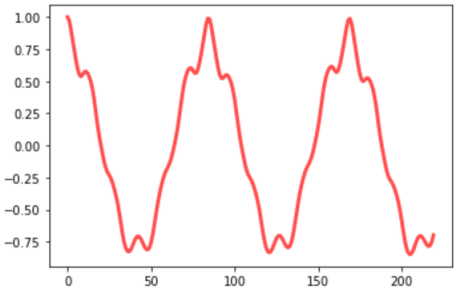
In [7]:
```
from thinkdsp import read_wave
w = read_wave('5_1.wav')
w.normalize()
w.make_audio()
```

Out[7]:



Далее неообходимо построить график автокорреляции. Для правильного построения графика надо будет воспользоваться функцией autocorr.

In [11]:
```
import thinkplot
seg = w.segment(0, 0.01)
lags, corrs = autocorr(seg)
lagx = np.array(corrs[90:110]).argmax()+90
thinkplot.plot(lags, corrs, color = 'red')
```



Проанализировав данный график можно увидеть период 90 Lag (по горизонтальной оси).

Суть работы 5.2:

Необходимо написать функцию estimate-fundamental, отслеживающую высоту тона записанного звука и проверить ее работоспособность, накладывая оценки высоты тона на спектрограмму записи.

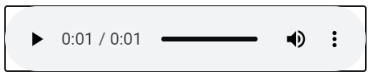
Напишем функцию estimate-fundamental, которая при помощи функции автокорреляции помогает отследить высоту тона.

In [12]:
```
def estimate_fundamental(segment, low=70, high=150):
    lags, corrs = autocorr(segment)
    lag = np.array(corrs[low:high]).argmax() + low
    period = lag / segment.framerate
    frequency = 1 / period
    return frequency
```

Далее считаем запись чирпа и прослушаем его.

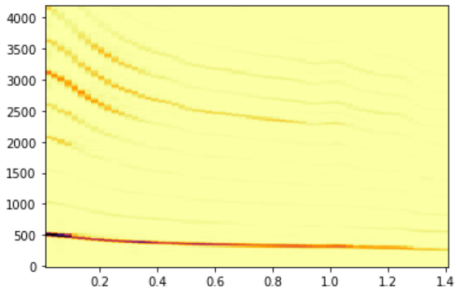
In [7]:
```
from thinkdsp import read_wave
w = read_wave('5_1.wav')
w.normalize()
w.make_audio()
```

Out[7]:



Необходимо построить спектрограмму полученной записи.

In [14]:
```
w.make_spectrogram(2048).plot(high=4200)
```



Затем получим частоту минимального сегмента. Это сделаем с помощью написанной функции estimate_fundamental.

In [16]:
```
seg = w.segment(start = 0.2, duration =0.01)
f = estimate_fundamental(seg)
f
```

Out[16]:
```
436.63366336633663
```

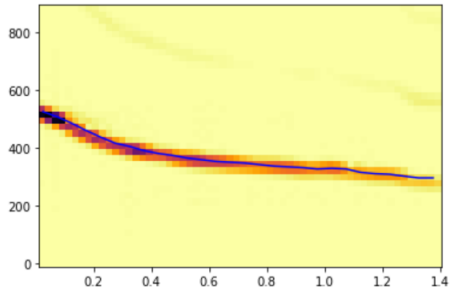
Далее найдем этот сегмент (шаг - 0.05).

```
In [18]: s = 0.05
         starts = np.arange(0.0, 1.4, 0.05)
         ts = []
         freqs = []
         for start in starts:
             ts.append(start + s/2)
             segment = w.segment(start=start, duration=0.01)
             freq = estimate_fundamental(seg)
             freqs.append(freq)
```

Выведем спектрограмму, на которой можно увидеть частоту тона в каждый момент времени.

```
In [23]: w.make_spectrogram(2048).plot(high=900)
         plt.plot(ts, freqs, color = 'blue')

Out[23]: [<matplotlib.lines.Line2D at 0x28cf03b8dc0>]
```

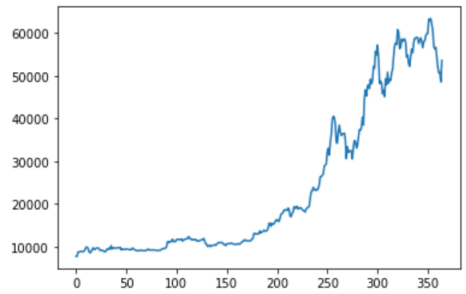


Суть работы 5.3:

Необходимо используя данные цен BitCoin из прошлой лабораторной работы вычислить автокорреляцию цен на BitCoin.

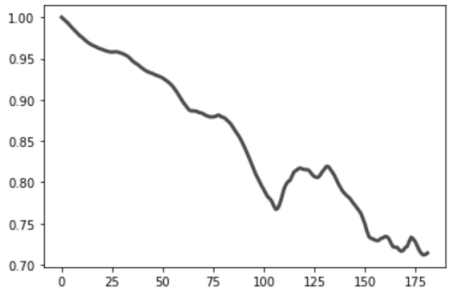
Из прошлой лабороторной работы возьмем файл с данными по BitCoin и представим данные в виде графика.

In [25]:
```
import pandas as pd
df = pd.read_csv('BTC_USD_2020-04-28_2021-04-27-CoinDesk.csv', nrows=600, parse_dates=[0])
ys = df['Closing Price (USD)']
ts = df.index
wave = Wave(ys, ts, framerate=1)
wave.plot()
```



И с помощью функции автокорреляции построим график. Периодичности в графике нет.

In [26]:
```
lags, corrs = autocorr(wave)
thinkplot.plot(lags, corrs, color = 'black')
```



Суть работы 5.4:

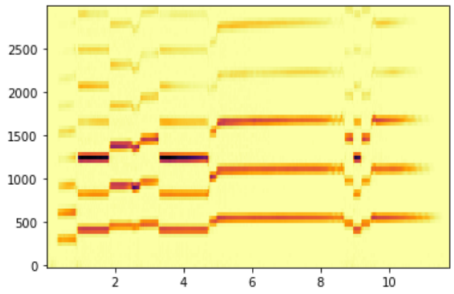
Необходимо прочитать блокнот saxophpne.ipynb, пройтись по всем примерам, затем выбрать другой сегмент записи и поработать с ним.

В лабораторной будет использован звук саксофона. Прослушаем его и построим спектрограмму.

```
In [27]: w = read_wave('5_4.wav')
         w.normalize()
         w.make_audio()
```

Out[27]: 


```
In [29]: spec = w.make_spectrogram(seg_length = 1024)
         spec.plot(high=3000)
```

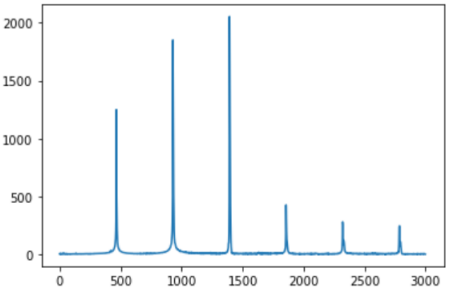


Выделим участок длительностью 0.5 секунды со 2ой секунды записи. Получим сразу спектр из данного сегмента.

```
In [30]: seg = w.segment(start = 2.0, duration = 0.5)
         seg.make_audio()
```

Out[30]: 


```
In [31]: spec = seg.make_spectrum()
         spec.plot(high=3000)
```



Выведем все пики полученного спектра.

```
In [32]: spec.peaks()[:10]
```

```
Out[32]: [(2054.0622639591206, 1392.0),
          (1850.8544230639036, 928.0),
          (1684.8468845494765, 1394.0),
          (1332.8150506072802, 930.0),
          (1249.1774991462646, 464.0),
          (1177.6718910227576, 1396.0),
          (857.3729096557305, 932.0),
          (742.841588837269, 1398.0),
          (515.1804113061312, 934.0),
          (513.7226300908811, 466.0)]
```

Перепишем функцию autocorr и сделаем автокорреляцию данного сегмента.

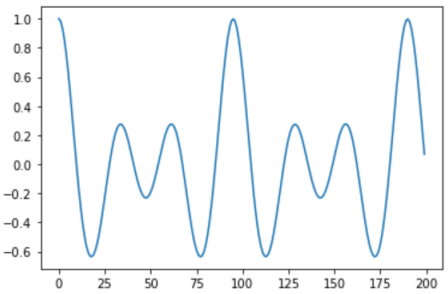
In [35]:
```
def autocorr(segment):
    corrs = np.correlate(segment.ys, segment.ys, mode='same')
    N = len(corrs)
    lengths = range(N, N//2, -1)

    half = corrs[N//2:].copy()
    half /= lengths
    half /= half[0]
    return half
```

Теперь с помощью новой функции построим на экране график.

In [36]:
```
corrs = autocorr(seg)
plt.plot(corrs[:200])
```

Out[36]:
```
[<matplotlib.lines.Line2D at 0x28cf0996cd0>]
```



Для нахождения частоты напишем функцию find_frequency и вызовем ее, передав на вход значения начала и конца lag (75 и 110).

In [37]:
```
def find_frequency(corrs, low, high):
    lag = np.array(corrs[low:high]).argmax() + low
    print(lag)
    period = lag / segment.framerate
    frequency = 1 / period
    return frequency
```

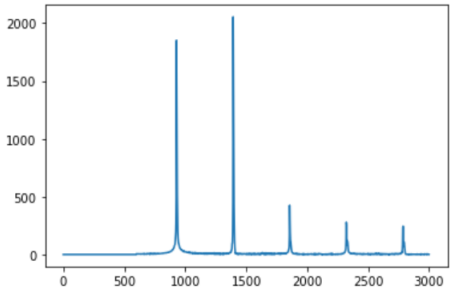
In [41]:
```
find_frequency(corrs, 75, 110)
```

```
95
```

Out[41]:
```
464.2105263157895
```

Пик находится на lag = 95.
Отфильтруем сегмент через фильтр низких частот и сравним звук с исходным сегментом.

In [39]:
```
spec2 = seg.make_spectrum()
spec2.high_pass(600)
spec2.plot(high=3000)
```



In [40]:
```
seg2 = spec2.make_wave()
seg2.make_audio()
```

Out[40]:

Звуки очень похожи друг на друга, но последний звучит более тише и приглушеннее.

Заключение:

По итогу выполнения данной лаборатной работы я изучила понятие автокорреляции, а также вычислять ее для разных lag. Была создана и использована новая функция estimate_fundamental, которая отслеживает высоту тона звука. Был использован файл из прошлой лабороторной работы (BitCoin) и вычислена автокорреляция цен, а также автокорреляция для звука саксофона.