In [ ]:
```
from thinkdsp import *
import numpy as np
import matplotlib.pyplot as plt
```

Лабораторная работа 5

Упражнение 5.1

> Оцените высоты тона вокального чирпа для нескольких времен начала сегмента

Файл lab5_1

Упражнение 5.2

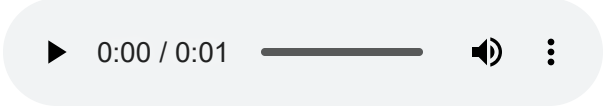
> Пример кода в `chap05.ipynb` показывает, как использовать автокорреляцию для оценки основной частоты периодического сигнала. Инкапсулируйте этот код в функцию, названную `estimate_fundamental` ,и используйте ее для отслеживания высоты тона записанного звука.
> Проверьте, насколько хорошо онп работает, накладывая оценки высоты тона на спектрограмму записи

In [ ]:
```
def serial_corr(wave, lag=1):
    """Computes serial correlation with given lag.

    wave: Wave
    lag: integer, how much to shift the wave

    returns: float correlation coefficient
    """
    n = len(wave)
    y1 = wave.ys[lag:]
    y2 = wave.ys[:n-lag]
    corr_mat = np.corrcoef(y1, y2)
    return corr_mat[0, 1]
```

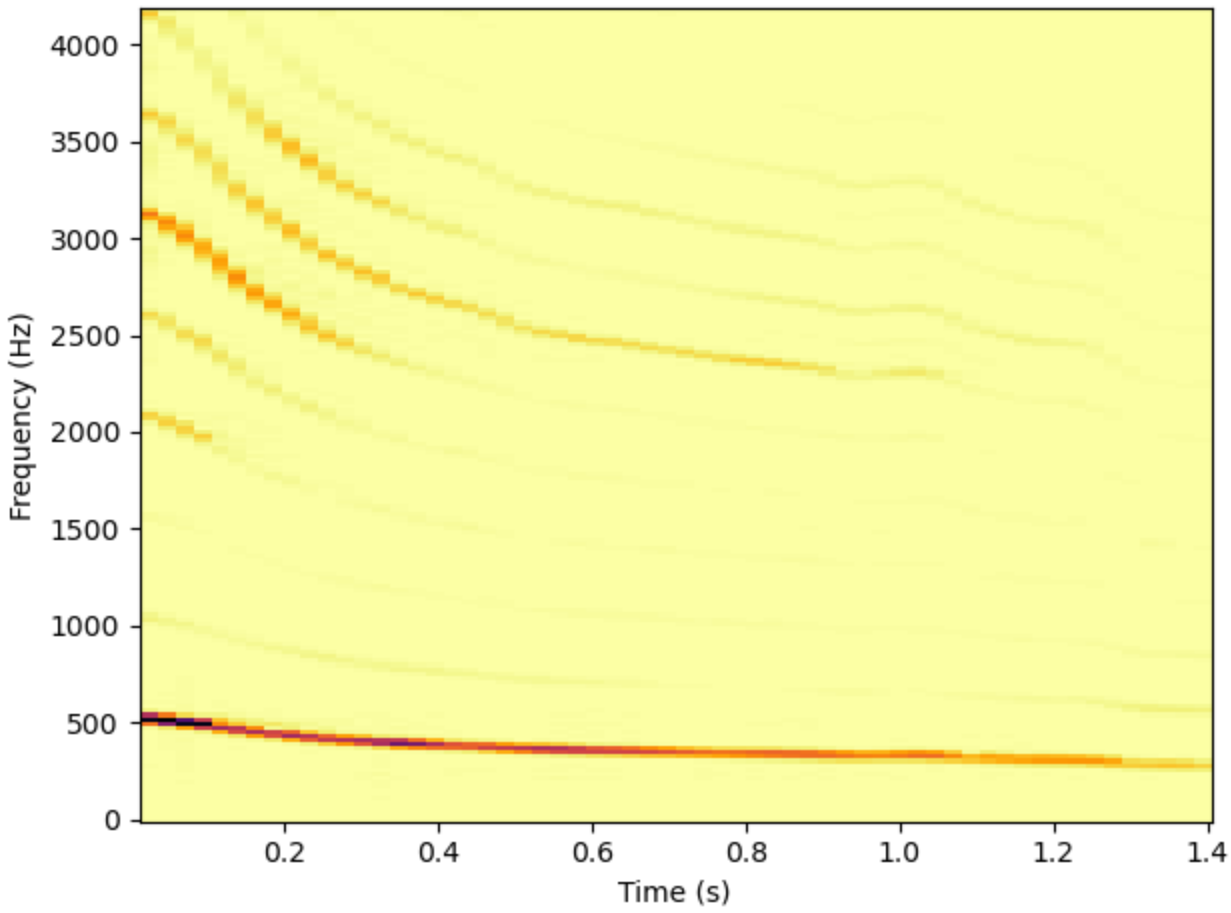
In [ ]:
```
def autocorr(wave):
    """Computes and plots the autocorrelation function.

    wave: Wave
    """
    lags = np.arange(len(wave.ys)//2)
    corrs = [serial_corr(wave, lag) for lag in lags]
    return lags, corrs
```

In [ ]:
```
wave = read_wave('28042__bcjordan__voicedownbew.wav')
wave.normalize()
wave.make_audio()
```

Out[ ]:



In [ ]:
```
wave.make_spectrogram(2048).plot(high=4200)
decorate(xlabel='Time (s)',
         ylabel='Frequency (Hz)')
```



In [ ]:
```
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
```

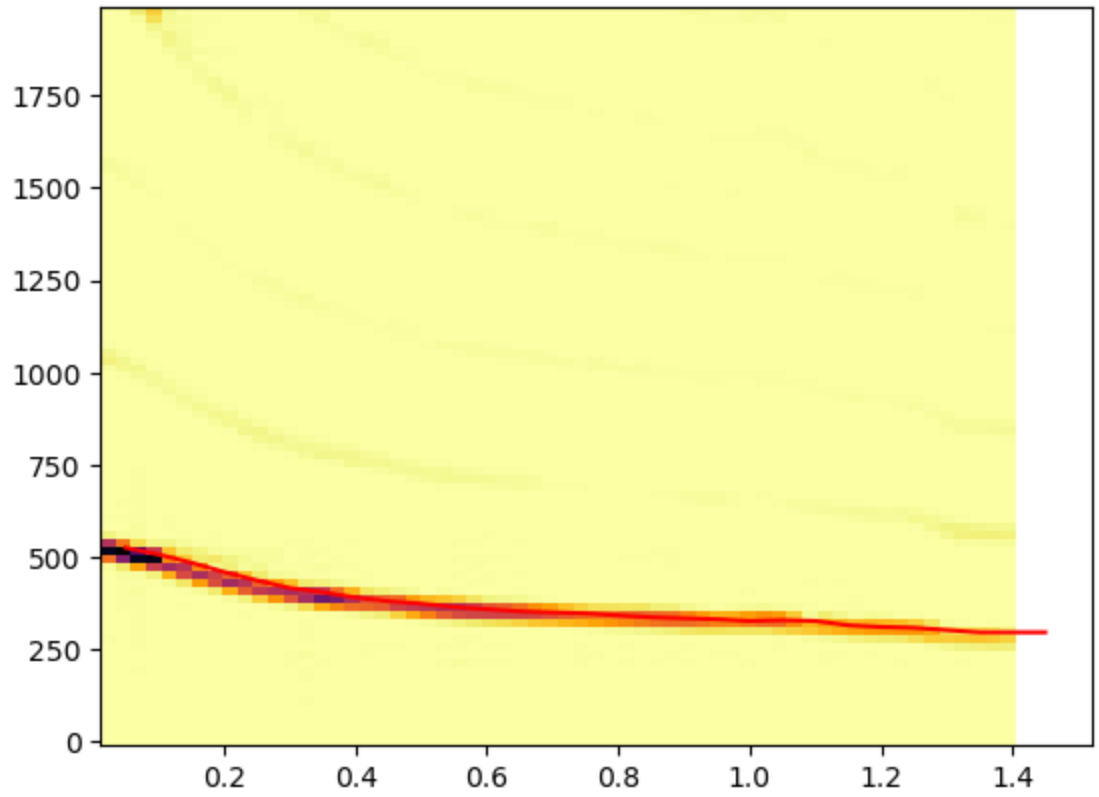
In [ ]:
```
duration = 0.01
segment = wave.segment(start=0.2, duration=duration)
freq = estimate_fundamental(segment)
freq
```

Out[ ]:
```
436.63366336633663
```

Основная частота ~436 Гц. Для проверки вычислим частоту на маленьких промежутках и наложим ее на спектрограмму

In [ ]:
```
x = np.arange(0, wave.duration, 0.05)
ts = []
freqs = []

for el in x:
    ts.append(el + 0.05)
    segment = wave.segment(el, duration)
    freqs.append(estimate_fundamental(segment))

wave.make_spectrogram(2048).plot(high=2000)
plt.plot(ts, freqs, color='red')
```

Out[ ]: [<matplotlib.lines.Line2D at 0x21eac93c610>]

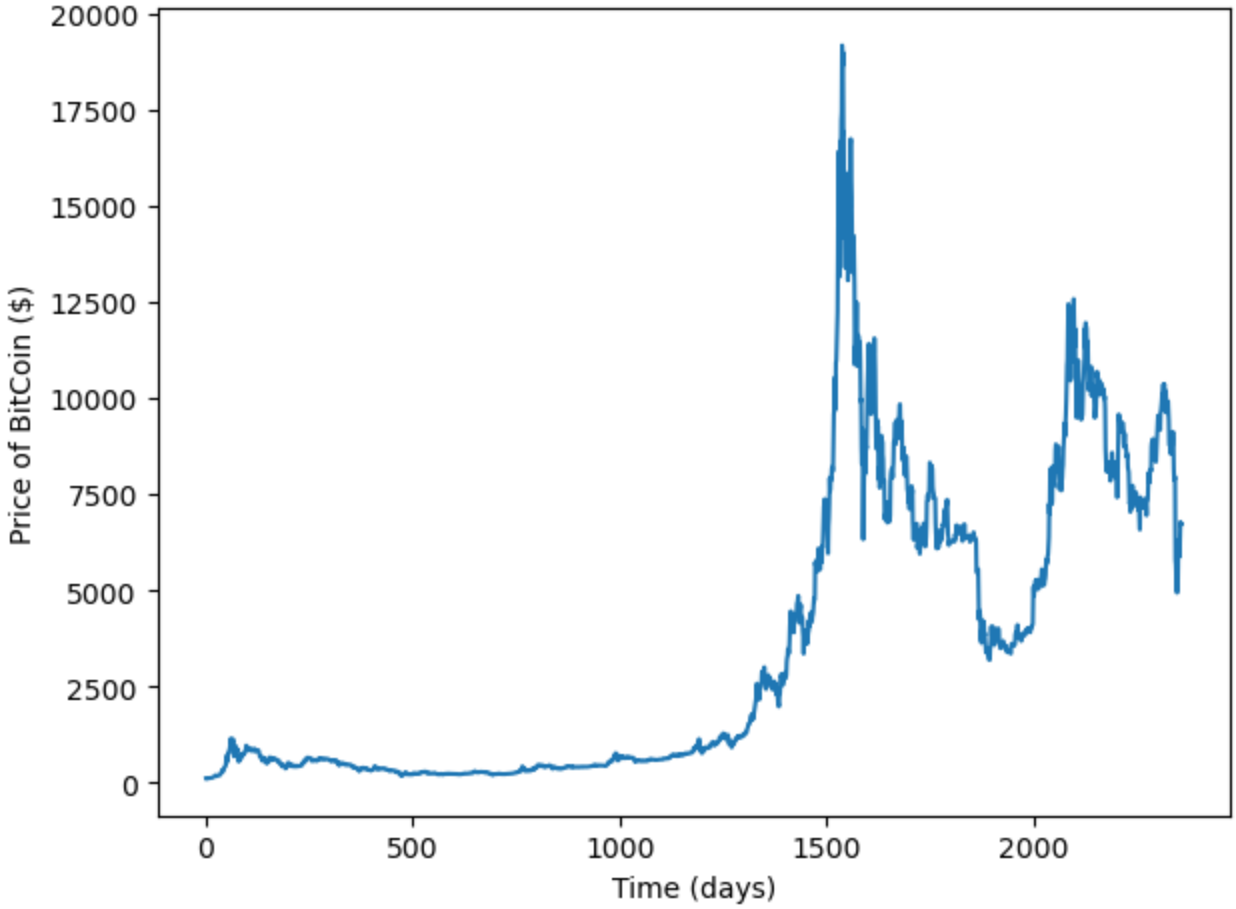


Упражнение 5.3

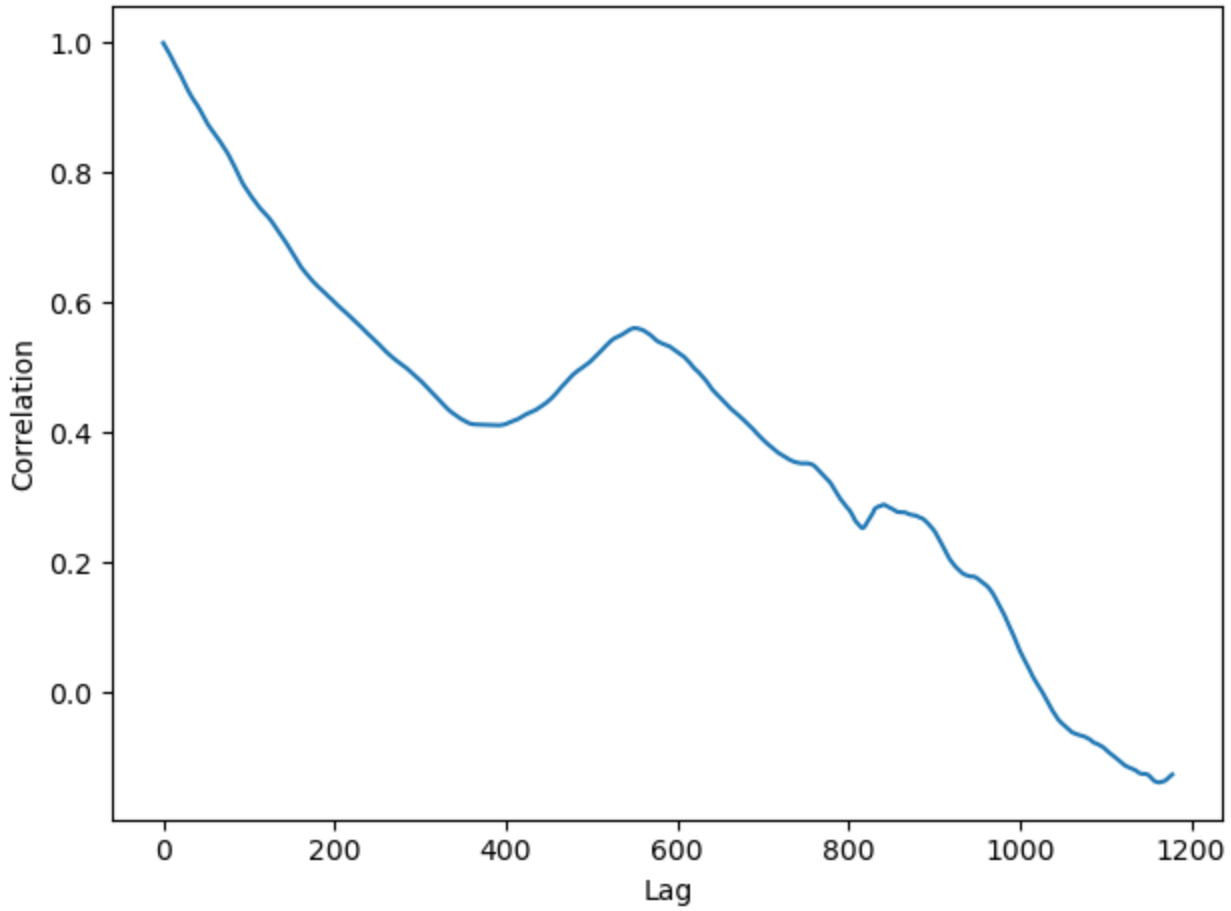
> Вычислите автокорреляции цен в платежной системе Bitcoin. Быстро ли спадает автокорреляционная функция, есть ли признаки периодичности процесса?

In [ ]:
```
import pandas as pd

df = pd.read_csv('BTC_USD_2013-10-01_2020-03-26-CoinDesk.csv',
                 parse_dates=[0])

ys = df['Closing Price (USD)']
ts = df.index
```

In [ ]:
```
wave = Wave(ys, ts, framerate=1)
wave.plot()
decorate(xlabel='Time (days)',
         ylabel='Price of BitCoin ($)')
```



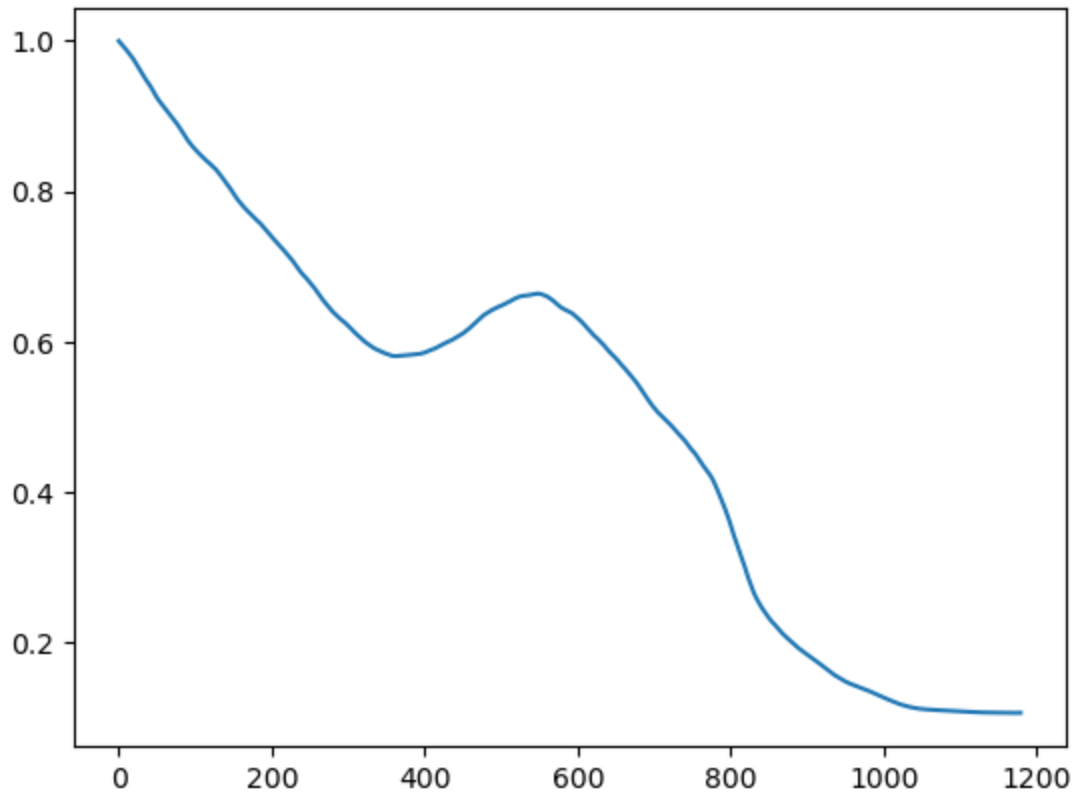
In [ ]:
```
lags, corrs = autocorr(wave)
plt.plot(lags, corrs)
decorate(xlabel='Lag',
         ylabel='Correlation')
```



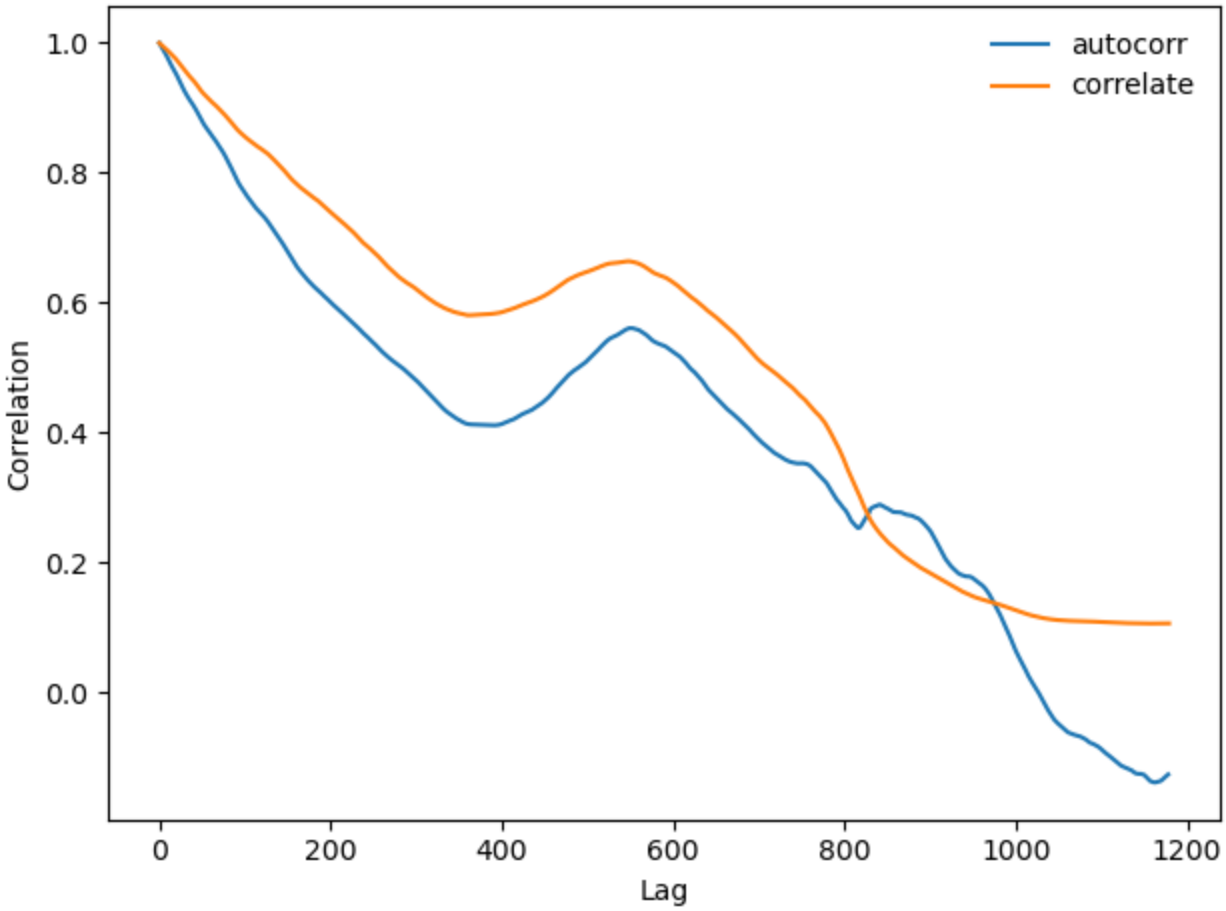
Автокорреляционная функция убывает, показывая какой-то вид розового шума. Сравним нашу автокорреляционную функцию с функцией корреляции пакета numpy

In [ ]:
```
N = len(wave)
corrs2 = np.correlate(wave.ys, wave.ys, mode='same')[N//2:]
lags = np.arange(-N//2, N//2)
lengths = range(N, N//2, -1)
corrs2 /= lengths
corrs2 /= corrs2[0]

plt.plot(corrs2)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x21eaffe9090>]
```



In [ ]:
```
plt.plot(corrs, label='autocorr')
plt.plot(corrs2, label='correlate')
decorate(xlabel='Lag', ylabel='Correlation')
```



Анализируя форму полученных функций модем заметить, что они достаточно отличаются по значениям, но схожи по форме. Периодичность функций не наблюдается