```python
from os.path import dirname, join as pjoin
from scipy.io import wavfile
import matplotlib.pyplot as plt
import numpy as np


if __name__ == '__main__':
    plt.style.use('seaborn')

    wave_path = pjoin(dirname(__file__), "voice_test.wav")
    wave_path_crop = pjoin(dirname(__file__), "voice_test_from2to3.wav")

    samplerate, data = wavfile.read(wave_path)
    length = data.shape[0] / samplerate

    samplerate_crop, data_crop = wavfile.read(wave_path_crop)
    length_crop = data_crop.shape[0] / samplerate_crop

    print(f"Original sample rate: {samplerate}")
    print(f"Original Audio length: {length}")

    print(f"Cropped version sample rate: {samplerate_crop}")
    print(f"Cropped version audio length: {length_crop}")

    # used to normalize the samples
    m1 = np.amax(np.abs(data))

    # changing from int16 to float32. Note that we need to normalize the
    # amplitudes since int16 has a range from (-32768, +32768) and float32
    # has (-1.0, +1.0)
    data = (data/m1).astype(np.float32)
    data_crop = (data_crop/m1).astype(np.float32)

    # voice_test and voice_test_float32 should sound the same
    wavfile.write("voice_test_float.wav", samplerate, data)

    # voice_test_from2to3.wav and voice_test_float_from2to3.wav should sound
    # the same
    wavfile.write("voice_test_float_from2to3.wav", samplerate_crop, data_crop)

    # defining correlation between original and cropped audio
    corr_signal = np.correlate(data[:, 0], data_crop[:, 0], 'same')
    corr_signal = corr_signal / np.amax(np.abs(corr_signal))

    plt.subplot(3, 1, 1)
    time = np.linspace(0., length, data.shape[0])

    plt.title("Original")
    plt.plot(time, data[:, 0], label="Left channel")
    plt.plot(time, data[:, 1], label="Right channel")
    plt.legend()
    plt.xlabel("Time [s]")
    plt.ylabel("Amplitude")

    plt.subplot(3, 1, 2)

    # create an array for better visualization of the cropped audio
    offset = 2 * samplerate
    data_crop_scaled = np.zeros_like(data)
    for i in range(data_crop.shape[0]):
        data_crop_scaled[offset + i] = data_crop[i]

    plt.title("Cropped")
    plt.plot(time, data_crop_scaled[:, 0], label="Left channel")
    plt.plot(time, data_crop_scaled[:, 1], label="Right channel")
    plt.legend()
    plt.xlabel("Time [s]")
```

```
plt.ylabel("Amplitude")

plt.subplot(3, 1, 3)

plt.title("Correlation")
plt.plot(time, corr_signal, label="correlation")
plt.legend()
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")

plt.tight_layout()
plt.show()
```