```
!wget https://pkholyavin.github.io/mastersprogramming/cta0001.wav
!wget https://pkholyavin.github.io/mastersprogramming/cta0001_stereo.wav
!wget https://pkholyavin.github.io/mastersprogramming/cta0001.sbl
```

Модуль wave

1. Чтение

```
import wave
import struct

f = wave.open("cta0001.wav")
# f = wave.open("cta0001_stereo.wav")

num_samples = f.getnframes()
print(num_samples)
samplerate = f.getframerate()
sampwidth = f.getsampwidth()
num_channels = f.getnchannels()

sampwidth_to_char = {1: "c", 2: "h", 4: "i"}  # на практике обычно 2 байта на о
# полную таблицу соответствий см. здесь: https://docs.python.org/3/library/stru
fmt = str(num_samples * num_channels) + sampwidth_to_char[sampwidth]

signal = struct.unpack(fmt, f.readframes(num_samples * num_channels))


# stereo file
left = signal[::2]
right = signal[1::2]
print(left == right)  # данный файл я создал так, чтобы левый и правый канал бы
# как видно, это правда


import matplotlib.pyplot as plt
plt.plot(signal)
plt.show()
```

Дополнительно: чтение 24-битных файлов

```
!wget https://pkholyavin.github.io/mastersprogramming/24bit.wav


f = wave.open("24bit.wav")

num_samples = f.getnframes()
samplerate = f.getframerate()
sampwidth = f.getsampwidth() # 3
```

```
num_channels = f.getnchannels()

data = f.readframes(num_samples * num_channels)
data = b''.join(b''.join((b'\x00', data[i:i+3])) for i in range(0, len(data), 3))
fmt = "i"
fmt = str(num_samples * num_channels) + fmt

signal = struct.unpack(fmt, data)
signal = [i >> 8 for i in signal]


plt.plot(signal)
plt.show()
```

В целом, если ваши файлы в каком-то малораспространённом формате, лучше сначала обработать их внешним инструментом и привести к нормальному виду. Это следует делать с осторожностью, чтобы случайно не внести искажения в данные.

https://ffmpeg.org/

2. Запись

```
from math import sin

samplerate = 22050
sampwidth = 2
num_channels = 1
num_samples = samplerate * 2
sampwidth_to_char = {1: "c", 2: "h", 4: "i"}
fmt = str(num_samples) + sampwidth_to_char[sampwidth]

ampl = 2 ** 14
sine = [int(ampl * sin(x / 20)) for x in range(num_samples)]
signal = struct.pack(fmt, *sine)

f = wave.open("output.wav", "wb")  # открываем на запись (w) в бинарном режиме (b)
f.setnchannels(num_channels)
f.setsampwidth(sampwidth)
f.setframerate(samplerate)
f.writeframes(signal)
f.close()
```

3. Файлы без заголовка (.sbl)

```
samplerate = 22050
sampwidth = 2
num_channels = 1
```

```
sampwidth_to_char = {1: "c", 2: "h", 4: "i"}

with open("cta0001.sbl", "rb") as f:
    raw_signal = f.read()

num_samples = len(raw_signal) // sampwidth
fmt = str(num_samples) + sampwidth_to_char[sampwidth]
signal = struct.unpack(fmt, raw_signal)


import matplotlib.pyplot as plt
plt.plot(signal)
plt.show()
```

Модуль scipy.io.wavfile

1. Чтение

```
import scipy.io.wavfile as wav
samplerate, signal = wav.read("cta0001.wav")
# samplerate, signal = wav.read("cta0001_stereo.wav")
print(signal.shape)


plt.plot(signal)
plt.show()
```

2. Запись

```
import numpy as np

samplerate = 22050
num_samples = samplerate * 2

ampl = 2 ** 14
sine = np.array([int(ampl * sin(x / 20)) for x in range(num_samples)], dtype=np.int16)

wav.write("output2.wav", samplerate, sine)
```

3. Чтение файлов без заголовка

```
with open("cta0001.sbl", "rb") as f:
    raw_signal = f.read()

signal = np.frombuffer(raw_signal, dtype=np.int16)
plt.plot(signal)
plt.show()
```

Модуль wavio

```
!pip install wavio
```

1. Чтение

```
import wavio
data = wavio.read("cta0001.wav")

signal, samplerate, sampwidth = data.data, data.rate, data.sampwidth
```

2. Запись

```
samplerate = 22050
num_samples = samplerate * 2

ampl = 2 ** 14
sine = np.array([int(ampl * sin(x / 20)) for x in range(num_samples)], dtype=np.int16)

wavio.write("output3.wav", sine, samplerate)
# wavio.write("output3.wav", sine, samplerate, sampwidth=2)
```

Модуль librosa

```
import librosa
```

1. Чтение

```
data, samplerate = librosa.load("cta0001.wav", sr=None)  # так сохраняем исходную частоту ди
print(samplerate)
```

Амплитуды в сигнале при этом нормализуются (приводятся в диапазон от -1 до 1 путём деления на максимально возможное значение)!

```
plt.plot(data)
plt.show()
```

```
data, samplerate = librosa.load("cta0001.wav")  # по умолчанию частота дискретизации приводи
print(samplerate)
```

```
data, samplerate = librosa.load("cta0001.wav", sr=44100)  # но можем задать какую угодно
print(samplerate)
```

Работа с массивами numpy (которые возвращают scipy и wavio)

```
new_array = np.asarray([1, 2, 3, 4, 5], dtype=np.int16) # используем int16 для работы с 16-б
print(new_array)
print(new_array.shape)
```

Обратите внимание, что у numpy-массива единый тип для всех его элементов!

Типы numpy более дифференцированы, чем базовые типы Python, поэтому будьте внимательны!

```
new_2d_array = np.asarray([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]], dtype=np.int16) # так д
print(new_2d_array)
print(new_2d_array.shape)
```

Сделаем из двух одномерных массивов двумерный:

```
array_2d = np.asarray((new_array, new_array), dtype=np.int16).T # T - это транспонирование
print(array_2d)
print(array_2d.shape)
```

```
zeros_array = np.zeros(5, dtype=np.int16) # массив, заполненный нулями
print(zeros_array.shape)
```

```
zeros_array2d = np.zeros((5, 2), dtype=np.int16)
print(zeros_array2d.shape)
```

```
print(new_array)
new_array = np.flip(new_array)
print(new_array)
```

```
big_array = np.concatenate((new_2d_array, zeros_array, new_2d_array))
print(big_array)
```

Будьте внимательны! При чтении моно-файлов scipy и librosa возвращают одномерный массив, wavio - двумерный, у которого второе измерение равно 1.
Конвертация из одной формы в другую:

```
array2d = np.asarray([[1], [2], [3], [4], [5]])
print(array2d.shape)
array1d = np.squeeze(array2d)
print(array1d.shape)
```

```
array1d = np.asarray([1, 2, 3, 4, 5])
print(array1d.shape)
array2d = np.reshape(array1d, (-1, 1))
print(array2d.shape)
```

Задания для выполнения в классе:

1. Написать программу, которая считывает .wav-файл, делит его на две половины и каждую записывает в отдельный файл. Оформите в виде функции, которая берёт на вход имя файла и возвращает имена получившихся файлов.

2. Написать программу, которая считывает .sbl-файл и сохраняет его как .wav. Оформите в виде функции, которая берёт на вход имя файла, частоту дискретизации, количество байт на отсчёт (по умолчанию - 2) и количество каналов (по умолчанию - 1) и возвращает имя получившегося файла.

Домашнее задание:

1. Считать моноканальный файл .wav
2. Сделать из него стереофайл так: в правый канал положить отсчёты левого в обратном порядке.
3. Вставить паузы 200 мс (или любое другое число, но явно прописанное в коде) на 1/4, 1/2 и 3/4 длительности.
4. Записать в новый файл.