Библиотека `parselmouth` представляет собой интерфейс, через который можно обращаться напрямую к коду Praat.

```
!pip install praat-parselmouth
```

https://parselmouth.readthedocs.io/en/stable/api_reference.html#parselmouth.Sound

Чтение файла

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

```
import parselmouth
sound = parselmouth.Sound("cta0001.wav")
```

Посмотрим, какие методы есть у класса Sound:

```
[i for i in dir(sound) if not i.startswith("_")]
```

Как видно, очень многие методы совпадают с тем, что можно сделать внутри Praat.

Например, получим объект, содержащий данные об интенсивности:

```
intensity = sound.to_intensity()
```

У объектов, которые представляют собой данные о некоторой величине, изменяющейся со временем, есть общие методы. Например, если мы хотим узнать значение величины в определённый момент времени, можем воспользоваться методом get_value().

```
print(sound.get_value(1))  # амплитуда осциллограммы на 1 секунде
```

```
print(intensity.get_value(1))  # значение интенсивности на 1 секунде
```

Также можно получить весь массив значений:

```
intensity.values.reshape(-1)
```

Построим осциллограмму из объекта Sound:

```
import matplotlib.pyplot as plt
plt.plot(sound.values.reshape(-1))
plt.xlabel("Samples")
plt.ylabel("Normalized amplitude")
plt.show()
```

А каким временам они соответствуют?

```
print(
    # информация о временных координатах:
    intensity.dt, # временной шаг
    intensity.xmin,  # время начала
    intensity.xmax,  # время конца
    intensity.centre_time  # время середины
)
```

Задание для выполнения в классе: построить два списка, в одном из которых будут значения времени от 0 до `intensity.xmax` с шагом 0.01 с, а в другом – значения интенсивности в этих точках. Построить график с помощью `matplotlib`.

```
times, intens_values = [], []
plt.plot(times, intens_values)
plt.show()
```

Объект Sound можно создать, не только прочитав из файла, но и другими методами. Передадим ndarray, полученный с помощью `scipy.io.wavfile`, в новый объект Sound.

```
from scipy.io.wavfile import read, write
fs, data = read("cta0001.wav")
new_sound = parselmouth.Sound(values=data, sampling_frequency=fs)
```

Обратите внимание, что при этом нормализации амплитуды не происходит!

```
plt.plot(new_sound.values.reshape(-1))
plt.xlabel("Samples")
plt.ylabel("Raw amplitude")
plt.show()
```

```
import numpy as np
norm_factor = np.iinfo(data.dtype).max
```

```
plt.plot(new_sound.values.reshape(-1) / norm_factor)  # проведём нормализацию вручную
plt.xlabel("Samples")
plt.ylabel("Normalized amplitude")
plt.show()
```

Получим объект Pitch, который хранит данные о частоте основного тона:

```
pitch = sound.to_pitch(pitch_floor=75, pitch_ceiling=400)
```

```
pitch.get_value_at_time(0.1)  # это звонкий участок
```

```
pitch.get_value_at_time(1.5)  # это глухой участок
```

Задание для выполнения в классе: прочитайте файл cta0001.seg_B1, определите место середины каждого звука, найдите значение ЧОТ в этом месте и постройте график, где на оси X по порядку будут отложены звуки, а на оси Y – ЧОТ в герцах.

```
!wget https://pkholyavin.github.io/mastersprogramming/cta0001.seg_B1
```

```
from itertools import product
letters = "GBRY"
nums = "1234"
levels = [ch + num for num, ch in product(nums, letters)]
level_codes = [2 ** i for i in range(len(levels))]
code_to_level = {i: j for i, j in zip(level_codes, levels)}
level_to_code = {j: i for i, j in zip(level_codes, levels)}


def read_seg(filename: str, encoding: str = "utf-8-sig") -> tuple[dict, list[dict]]:
    with open(filename, encoding=encoding) as f:
        lines = [line.strip() for line in f.readlines()]

    # найдём границы секций в списке строк:
    header_start = lines.index("[PARAMETERS]") + 1
    data_start = lines.index("[LABELS]") + 1

    # прочитаем параметры
    params = {}
    for line in lines[header_start:data_start - 1]:
        key, value = line.split("=")
        params[key] = int(value)

    # прочитаем метки
    labels = []
    for line in lines[data_start:]:
        # если в строке нет запятых, значит, это не метка и метки закончились
```

```
            if line.count(",") < 2:
                break
            pos, level, name = line.split(",", maxsplit=2)
            label = {
                "position": int(pos) // params["BYTE_PER_SAMPLE"] // params["N_CHANNEL"],
                "level": code_to_level[int(level)],
                "name": name
            }
            labels.append(label)
        return params, labels


    sounds = []
    f0_values = []
    plt.plot(f0_values)
    plt.xticks(range(len(sounds)), labels=sounds)  # отложим названия звуков на оси X
    plt.show()
```

Получим объект Spectrogram, который хранит данные динамической спектрограммы:

```
spectrogram = sound.to_spectrogram()
```

```
spectrogram.get_power_at(0.23, 100)
```

Посмотрим на мгновенную спектрограмму:

```
slice = spectrogram.to_spectrum_slice(0.23)
```

```
num_bins = slice.get_number_of_bins()
```

```
freqs = [slice.get_frequency_from_bin_number(i) for i in range(1, num_bins + 1)]
vals = [np.log10(abs(slice.get_value_in_bin(i))) for i in range(1, num_bins + 1)]
plt.plot(freqs, vals)
plt.show()
```

Получим объект Formant, который хранит данные о значениях формант:

```
formants = sound.to_formant_burg()
```

У него метод `get_value_at_time()` принимает два аргумента: номер форманты (от 1 до 4) и время. Найдём форманты гласного [u0] в слове "Юрий":

```
time = 0.23  # гласный u0 в слове юрий
for f in range(1, 5):
    print(f"F{f}: {formants.get_value_at_time(f, 0.23)}")
```

Задание для выполнения в классе: модифицировать код из предыдущего задания так, чтобы в трёх точках каждого гласного (1/4, 1/2 и 3/4 длины) вычислялись первая и вторая форманты. Нарисовать график.

```
x_ticks = []
f1_values = []
f2_values = []

plt.plot(f1_values)
plt.plot(f2_values)
plt.xticks(range(len(f1_values)), labels=x_ticks)
plt.show()
```

Домашнее задание: прочитать файлы по ссылке. Взять уровень ideal из каждого .TextGrid. Вычислить значения формант в серединке каждого гласного. Построить точечный график зависимости второй форманты от первой (каждый гласный своим цветом).

```
import os
folder = "/content/drive/MyDrive/fpt1"
os.listdir(folder)
```

Как оформить такой график:

```
plt.scatter([1, 2, 3, 4, 5], [3, 6, 1, 2, 8], label="vowel 1")  # первый список - F1 разных
plt.scatter([6, 7, 8, 9, 10], [-1, 0, 2, -4, -3], label="vowel 2")
plt.legend()
plt.show()
```

Обратите внимание, что в файлах некоторые символы переданы в виде т.н. триграфов:

```
trigraph2unicode = {
    "\\as": "ɑ",
    "\\i-": "ɨ",
}
```

Подробнее о триграфах см. в тетрадке про .TextGrid.