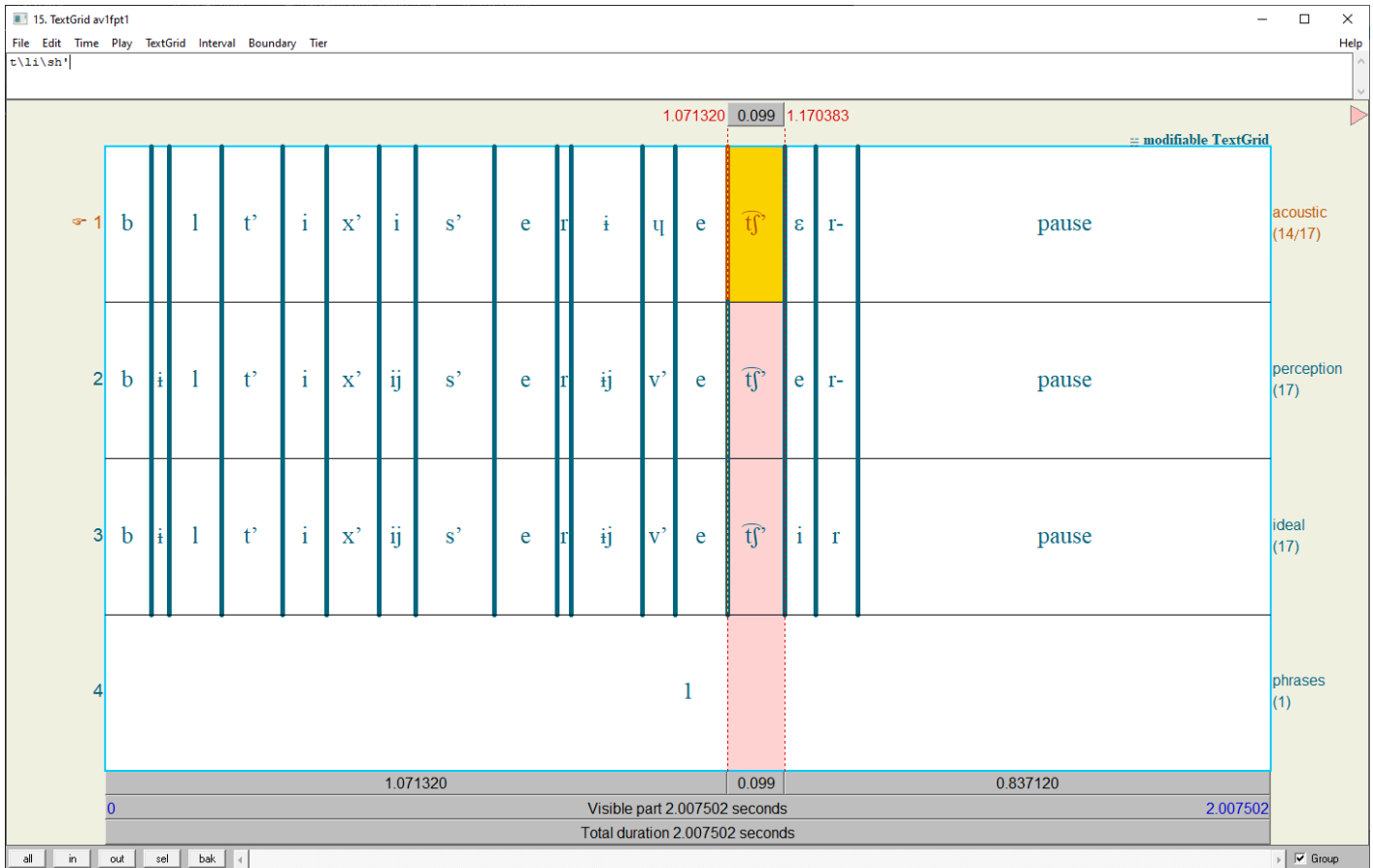


Не для всех объектов Praat в `parselmouth` есть специальные классы. Также не для всех объектов, для которых классы есть, реализованы все необходимые методы. Для таких случаев в `parselmouth` есть специальные функции, которые позволяют выполнять команды Praat напрямую.

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

```
!wget https://pkholyavin.github.io/mastersprogramming/av1fpt1.TextGrid
```

Для чтения файлов есть функция `parselmouth.read()`. Она соответствует команде Praat "Read object from file". Например, прочитаем `.TextGrid`:



```
import parselmouth
tg = parselmouth.read("av1fpt1.TextGrid")
tg
```

Мы получили объект класса `TextGrid`:

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

Посмотрим, что у него внутри:

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

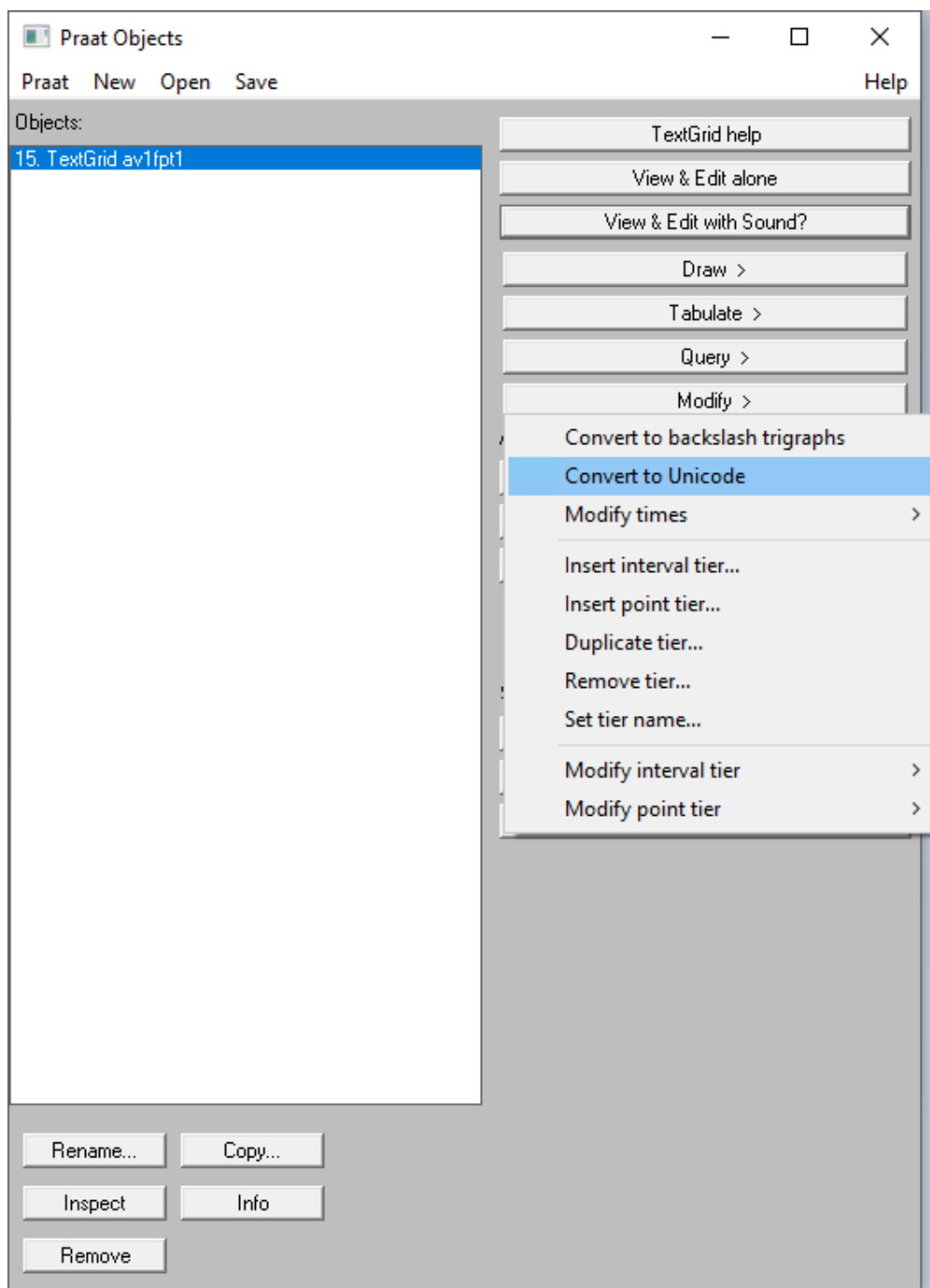
В частности, обратите внимание, что у класса `TextGrid` есть методы `from_tgt()` и `to_tgt()`, которые позволяют взаимную конвертацию с форматом `TextGridTools`.

Вопрос: что выведет следующий код?

```
tgt_grid = tg.to_tgt()  
print(" ".join([i.text for i in tgt_grid.get_tier_by_name("ideal")]))
```

А если нам нужно сделать что-то, для чего нет специального метода? Используем функцию `parselmouth.praat.call()`. В неё первым аргументом будем передавать объект, над которым хотим совершить операцию, а вторым – название соответствующей команды Praat.

Например, если мы хотим перевести текст разметки из триграфов Praat в символы Unicode, нам нужна команда "Convert to Unicode" (обратите внимание, что команда должна быть передана в точности так, как она отображается в Praat; регистр важен!):



```

parselmouth.praat.call(tg, "Convert to Unicode")
tgt_grid_uni = tg.to_tgt()
print(" ".join([i.text for i in tgt_grid_uni.get_tier_by_name("ideal")]))

```

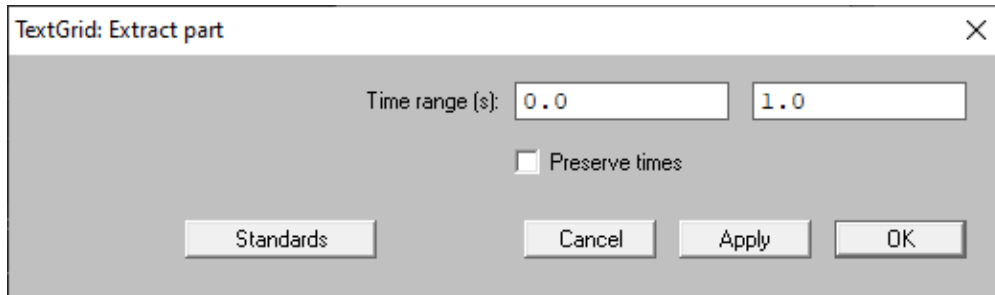
Сохраним результат:

```
tg.save("av1fpt1_unicode.TextGrid")
```

А если для выполнения операции нужно задать какие-то параметры? Тогда будем передавать их как дальнейшие аргументы. Например, если мы хотим извлечь кусок

TextGrid, то нам нужна команда "Extract part". Она требует три аргумента: время начала, время конца (в секундах) и чекбокс, который определяет, нужно ли сохранять изначальные значения времени. Время начала и конца передадим как числа, а значение чекбокса – как True или False.

Обратите внимание, что предыдущая команда изменяла объект, над которым проводилась операция. Эта команда генерирует уже новый объект; соответственно, функция `call()` возвращает его, и мы можем присвоить его в какую-то переменную.



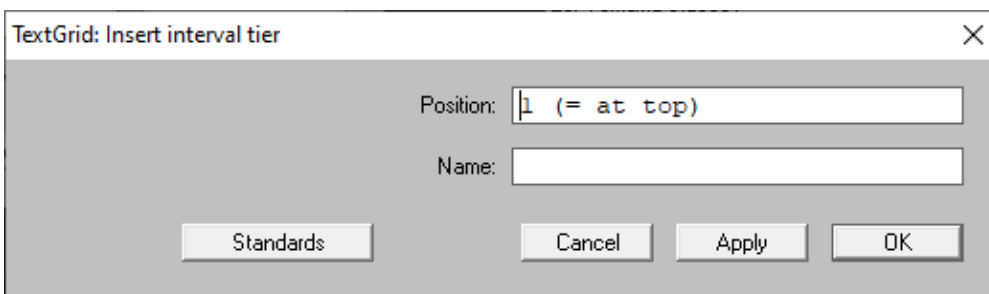
```
start_time, end_time = 0.0, 1.0 # s
pres_times = False
new_tg = parselmouth.praat.call(tg, "Extract part", start_time, end_time, pres_times)
tgt_grid_part = new_tg.to_tgt()
print(" ".join([i.text for i in tgt_grid_part.get_tier_by_name("ideal")]))
```

В документации к функции `call()` можно прочитать, в каком формате необходимо передавать параметры команд:

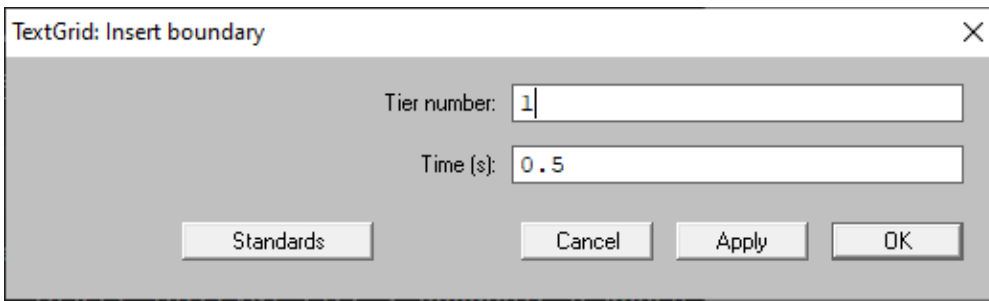
https://parselmouth.readthedocs.io/en/stable/api_reference.html#parselmouth.praat.call

Задание для выполнения в классе:

1. Используя команду "Insert interval tier", добавьте в конец нашего TextGrid (на позицию 5) новый уровень. Придумайте для него название.



2. Используя команду "Insert boundary", разделите его на пять равных интервалов. Воспользуйтесь атрибутом `xmax` класса `TextGrid`, чтобы найти длину файла.

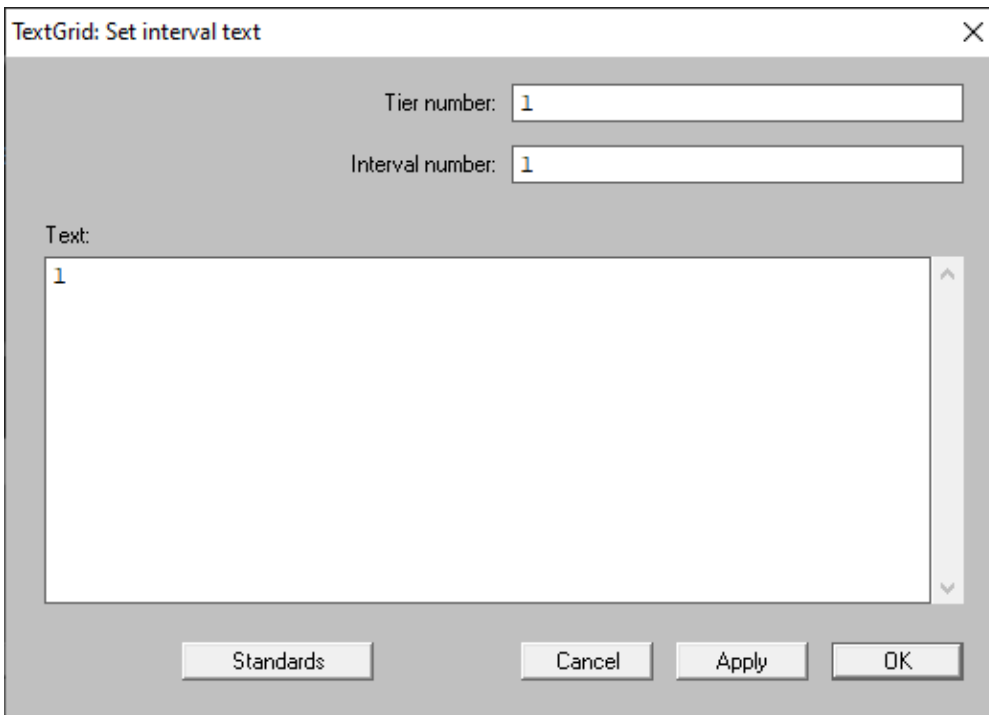


TextGrid: Insert boundary

Tier number:

Time (s):

3. Используя команду "Set interval text", назовите каждый интервал буквой латинского алфавита от А до Е.



TextGrid: Set interval text

Tier number:

Interval number:

Text:

1

Функция `call()` позволяет работать и с командами, которые не работают с существующими объектами, а создают новые. Например, если мы хотим прочитать файл `.sbl`, нас интересует команда "Read Sound from raw 16-bit Little Endian file". При этом функция `call()` вернула нам объект класса `Sound`.

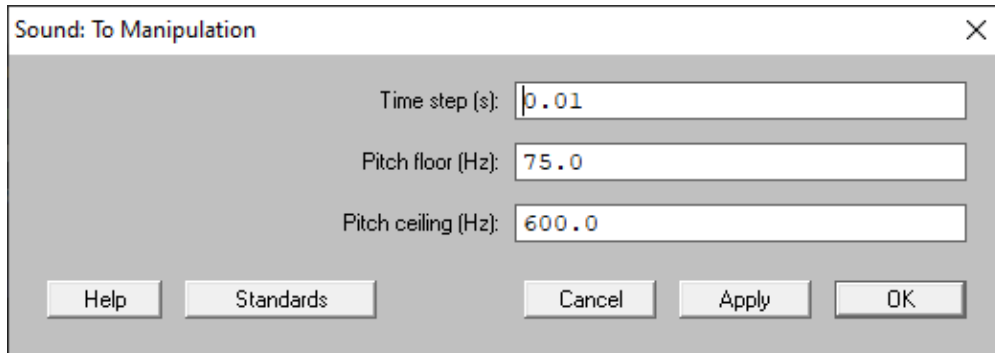
```
!wget https://pkholyavin.github.io/mastersprogramming/cta0001.sbl
```

```
sound_sbl = parselmouth.praat.call("Read Sound from raw 16-bit Little Endian file", "cta0001
```

При этом ЧД по умолчанию задаётся 16 кГц, поэтому восстановим истинное значение с помощью метода `override_sampling_frequency()`.

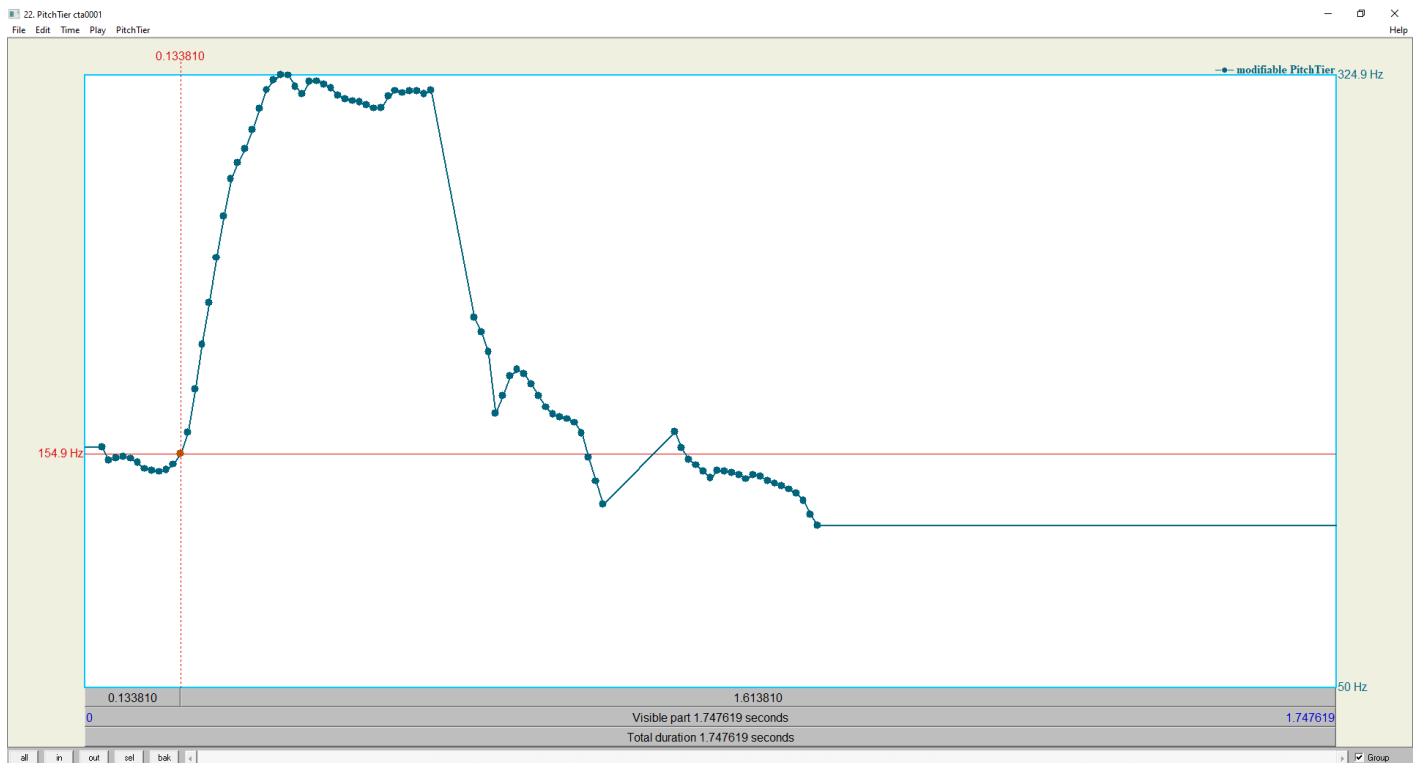
```
sound_sb1.override_sampling_frequency(22050)
```

Ещё одна возможность Praat, для которой не реализована прямая поддержка в `parselmouth`, – это манипуляция звуком. Если мы выполним команду "To Manipulation" на звуке, то получим объект Manipulation, с которым можем потом работать.



```
step, min_f0, max_f0 = 0.01, 75, 350 # s, Hz, Hz
manip = parselmouth.praat.call(sound_sb1, "To Manipulation", step, min_f0, max_f0)
```

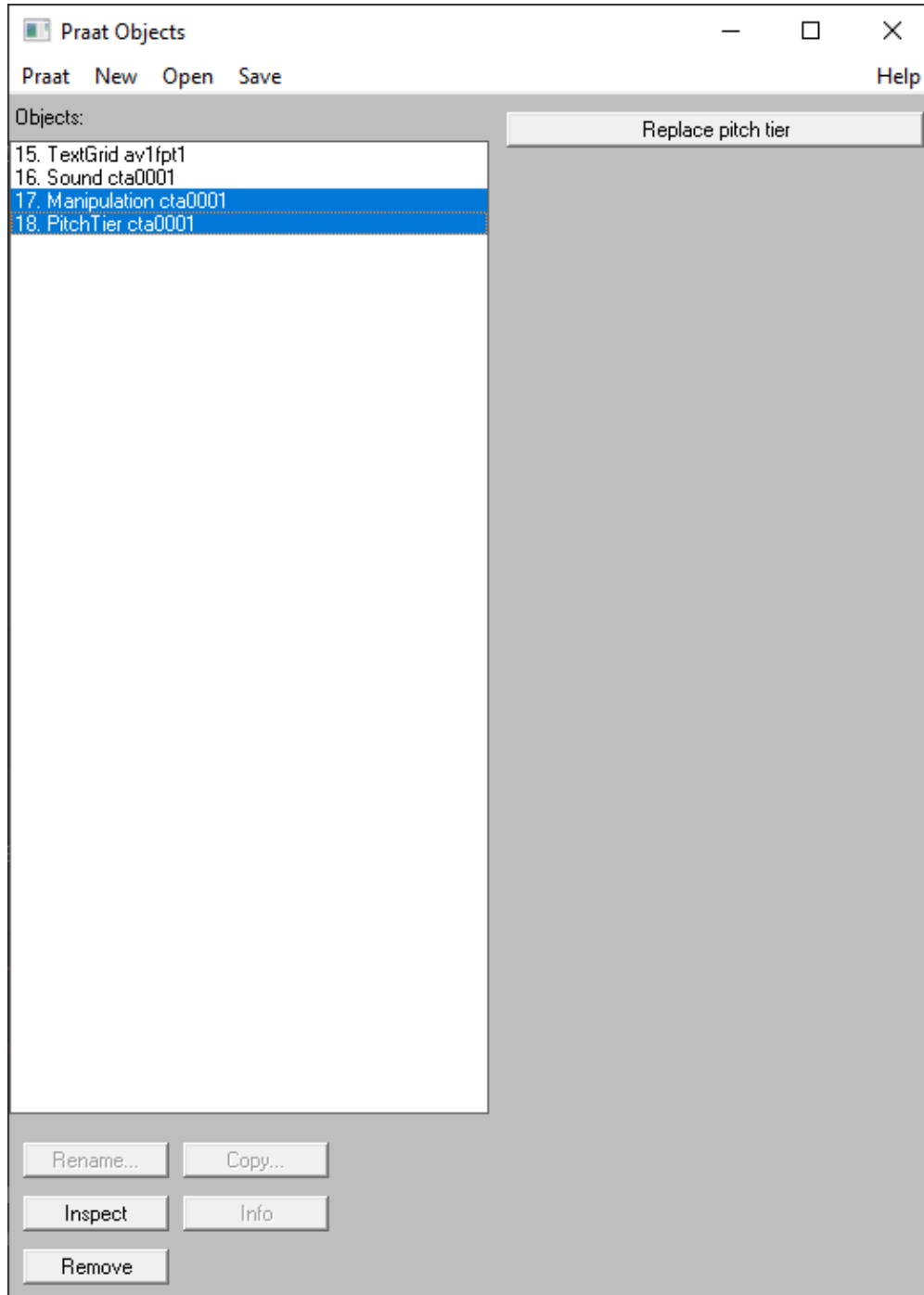
Извлечём из объекта Manipulation объект PitchTier и умножим в нём все значения на 1.2. Каждый PitchTier – это последовательность точек, которые описываются двумя параметрами: время (в секундах) и значение ЧОТ (в герцах).



```
pitch_tier = parselmouth.praat.call(manip, "Extract pitch tier")
factor = 1.2
parselmouth.praat.call(pitch_tier, "Multiply frequencies", sound_sbl.xmin, sound_sbl.xmax, f
```

Теперь воспользуемся командой "Replace pitch tier", чтобы наложить наш изменённый PitchTier на объект Manipulation. Обратите внимание, что, поскольку эта команда оперирует сразу двумя объектами, теперь мы передаём первым аргументом *список*, содержащий эти объекты.

(Если бы вы делали это вживую, то вы бы выделили оба объекта мышкой вместе)



```
parselmouth.praat.call([manip, pitch_tier], "Replace pitch tier")
```

Теперь получим из объекта Manipulation новый звук и запишем его в файл:

```
snd_new = parselmouth.praat.call(manip, "Get resynthesis (overlap-add)")  
snd_new.save("cta0001_mod.wav", "WAV")
```

Воспользуемся библиотекой IPython.display, чтобы послушать, что у нас получилось:

```
from IPython.display import Audio  
Audio(snd_new.values, rate=snd_new.sampling_frequency)
```

С помощью функции call() можно также выполнять команды, которые не меняют объекты, а возвращают какую-то информацию о них. Например, узнаем, сколько точек в нашем PitchTier'e:

```
num_points = parselmouth.praat.call(pitch_tier, "Get number of points")  
print(num_points)
```

Посмотрим, какая ЧОТ задана в первой точке (обратите внимание, что нумерация начинается с единицы):

```
print(parselmouth.praat.call(pitch_tier, "Get value at index", 1))
```

И посмотрим, на какой временной отметке (в секундах) располагается эта точка:

```
print(parselmouth.praat.call(pitch_tier, "Get time from index", 1))
```

Задание для выполнения в классе: напишите программу, которая получает информацию о всех точках в PitchTier'e и строит график зависимости ЧОТ от времени. Для этого сделайте два списка, каждый длиной num_points, в одном из которых будут значения времени, а во втором – значения ЧОТ.

```
import matplotlib.pyplot as plt
```

```
time_values = []  
f0_values = []
```



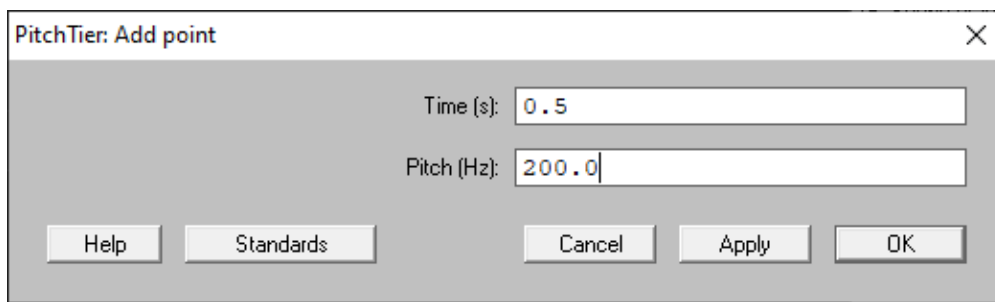
```
# нарисуем график отдельными точками
plt.plot(time_values, f0_values, linestyle="", marker="o")
plt.xlabel("Time, s")
plt.ylabel("F0, Hz")
plt.show()
```

Мы можем создать и пустой PitchTier с нужными параметрами, в который сможем добавлять свои точки по нашему желанию:

```
start_time = sound_sbl.xmin
end_time = sound_sbl.xmax
```

```
new_pitch_tier = parselmouth.praat.call("Create PitchTier", "new_pitch_tier", start_time, er
```

Давайте добавим в новый PitchTier две точки: одну с частотой 150 Гц на 1/4 длины файла, а другую – с частотой 300 Гц на 3/4 длины файла. Для этого воспользуемся командой "Add point".



```
parselmouth.praat.call(new_pitch_tier, "Add point", end_time * 0.25, 150)
parselmouth.praat.call(new_pitch_tier, "Add point", end_time * 0.75, 300)
```

Воткнём его в нашу манипуляцию:

```
parselmouth.praat.call([manip, new_pitch_tier], "Replace pitch tier")
simple_sound = parselmouth.praat.call(manip, "Get resynthesis (overlap-add)")
Audio(simple_sound.values, rate=simple_sound.sampling_frequency)
```

Задание для выполнения в классе: похулиганим! Создайте копию изначального PitchTier'a, где значение каждой точки будет случайным (от минимального до максимального в изначальном PitchTier'e) и синтезируйте звук.

```
import random
random.uniform(1, 10)
```

Домашнее задание: пересадить мелодический контур из файла cta0001.wav (донор) в файл kta0001.wav (акцептор) по следующему алгоритму:

1. Сгенерировать PitchTier для файла cta0001.wav
2. Перебрать все точки в нём
3. Для каждой точки определить, внутри какого звука она находится (по файлу cta0001.seg_B1), и на какой доле его длины она находится (е.g. 0.5, если она находится в самой середине, или 0.25, если она находится в конце первой четверти)
4. На основании этой информации и файла kta0001.seg_B1 определить, где эта точка лежала бы в новом файле
5. Создать новый (пустой) PitchTier под звук kta0001.wav
6. По очереди добавить в него точки из старого, исправляя их временные позиции так, чтобы в новом файле они лежали в тех же местах тех же звуков, что и в старом
7. Ресинтезировать kta0001 с новым PitchTier'ом
8. При желании повторить всё то же самое в обратную сторону

Примечание: внимательно отнеситесь к выбору `min_f0` и `max_f0` при создании объекта манипуляции для каждого из файлов! Проверьте (можно вручную через Praat), не возникает ли грубых ошибок.

