Существует несколько библиотек, которые помогают работать с фонетической транскрипцией. Одна из них – `phonecodes`, которая позволяет конвертировать транскрипцию из одной системы в другую.

https://pypi.org/project/phonecodes/

```
!pip install phonecodes
```

Выведем поддерживаемые системы транскрипции:

```
from phonecodes import phonecodes
print(phonecodes.CODES)
```

IPA – это Международный фонетический алфавит. X-SAMPA – это способ записи МФА с помощью только символов ASCII, предложенный Дж. Уэллзом.

```
phonecodes.convert("'bɨl 'tʲixʲij 'sʲerɨj 'vʲet͡ʃʲir", "ipa", "xsampa")
```

Эквивалентное использование:

```
phonecodes.ipa2xsampa("'bɨl 'tʲixʲij 'sʲerɨj 'vʲet͡ʃʲir")
```

Обратите внимание, что некоторые символы не конвертируются – например, дуги, соединяющие аффрикаты и двойные артикуляции. Их можно заменить нижним подчёркиванием:

```
phonecodes.ipa2xsampa("'bɨl 'tʲixʲij 'sʲerɨj 'vʲet͡ʃʲir").replace("\u0361", "_")
```

Также обратите внимание на способ передачи символов Unicode: вместо того, чтобы печатать символ ◌͡◌ напрямую (он комбинирующий, поэтому сам по себе плохо читается в коде), укажем его шестнадцатеричное представление после специальной последовательности `\u`.

Другая система транскрипции, представленная в `phonecodes`, – ARPABET. Это система для английского языка, также целиком основанная на символах ASCII.

```
phonecodes.convert("ð ɪ s ɪ z ə t ˈɛ s t", "ipa", "arpabet")
```

Стоит заметить, что ARPABET может выглядеть немного по-разному в разных источниках. В частности, в корпусе TIMIT версия ARPABET отличается от базовой. Версия, представленная в `phonecodes`, подойдёт, например, чтобы работать со словарём CMU Pronouncing Dictionary.

```
phonecodes.convert("G R IY N", "arpabet", "ipa")
```

Задание для выполнения в классе: создайте для файла av1fpt1.TextGrid дополнительный уровень "ideal x-sampa", где продублируйте транскрипцию с уровня ideal в системе X-SAMPA. Подумайте, какие дополнительные преобразования нужно совершить с транскрипцией.

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

Ещё одна интересная библиотека – `panphon`, которая позволяет конвертировать символы МФА в наборы бинарных фонологических признаков.

https://pypi.org/project/panphon/0.5/

```
!pip install panphon==0.5
```

```
import panphon._panphon as panphon
ft = panphon.FeatureTable()
```

Функция `word_fts()` сгенерирует список наборов признаков для каждого звука в слове:

```
ft.word_fts('pʲãk')
```

Функция `fts()` вернёт набор признаков для одного звука (или `None`, если поданная строка не является символом МФА):

```
ft.fts("tʰ")
```

```
ft.fts("ы")
```

Некоторые из этих признаков сомнительны:

```
f1, f2 = ft.word_fts('aɑ')
print(f1.difference(f2))
print(f2.difference(f1))
```

Задание для выполнения в классе: создайте для файла av1fpt1.TextGrid дополнительный уровень "vc", где интервалы будут обозначать стечения гласных (V) и согласных (C). Ориентируйтесь на уровень acoustic.

Обратите внимание, что в разметке корпуса символ МФА ɛ отражён не как `\u025b`, а как `\u03b5` (греческий эпсилон). `panphon` требует первый вариант.

Дефис после [r] обозначает оглушение, что в МФА обозначается кружочком снизу: [r̥]. В Unicode это символ `\u0325`.

```
print("\u025b", "\u03b5")
```

Следующая полезная библиотека – `pympi`, которая позволяет читать файлы разметки ELAN.

https://dopefishh.github.io/pympi/index.html

```
!pip install pympi-ling
```

```
import pympi
```

```
!wget https://pkholyavin.github.io/mastersprogramming/kholyavin_-_dolg_i.eaf
```

```
eaf = pympi.Eaf("kholyavin_-_dolg_i.eaf")
```

Посмотрим на все аннотации (интервалы) во всех уровнях:

```
for tier in eaf.tiers:
    print(f"{tier=}")
    annotations = eaf.tiers[tier][0]
    for id in annotations:
```

```
        begin_ts, end_ts, value, _ = annotations[id]
        print(f"{begin_ts=}, {end_ts=}, {value=}")
```

Словарь, который содержит время каждого таймслота в миллисекундах:

```
eaf.timeslots
```

Повторим то же самое, но вместо идентификаторов таймслотов подставим их значения:

```
for tier in eaf.tiers:
    print(f"{tier=}")
    annotations = eaf.tiers[tier][0]
    for id in annotations:
        begin_ts, end_ts, value, _ = annotations[id]
        begin_time = eaf.timeslots[begin_ts]
        end_time = eaf.timeslots[end_ts]
        print(f"{begin_time=}, {end_time=}, {value=}")
```

Сконвертируем в .TextGrid:

```
new_tg = eaf.to_textgrid()
new_tg.to_file("kholyavin_-_dolg_i.TextGrid")
```

Файлы .eaf – это разновидность файлов XML (Extensible Markup Language). Для чтения файлов XML и HTML есть библиотека Beautiful Soup (установлена в Colab по умолчанию).

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
from bs4 import BeautifulSoup
with open("kholyavin_-_dolg_i.eaf") as f:
    soup = BeautifulSoup(f.read(), features="xml")
```

```
print(soup.prettify())
```

Посмотрим на аннотацию к корпусу LeaP (формат TASX):

```
!wget https://pkholyavin.github.io/mastersprogramming/ab_pol_eng_f_free_c1.xml
```

```
with open("ab_pol_eng_f_free_c1.xml") as f:
    soup = BeautifulSoup(f.read(), features="xml")
```

```
print(soup.prettify())
```

Переберём все уровни аннотации, у каждого посмотрим метаданные. Для этого воспользуемся функцией `find_all()`, в которую в качестве аргумента передадим имя тега, который нас интересует. Функция вернёт нам список всех подходящих тегов. Внутри каждого тега тоже есть теги, можем использовать функцию ещё раз, чтобы найти внутри интересующую нас информацию.

Отберём уровень, в котором содержатся слова. Каждое слово – это отдельное "событие", у которого время начала и конца хранятся как атрибуты, а сам текст – как содержание тега.

Содержание (`contents`) – это всё то, что хранится между открывающим и закрывающим тегом. Это может быть как текст (строка), так и другие теги. В `bs4` в `.contents` хранится список.

Все атрибуты тега (то, что пишется внутри открывающего тега со знаком =) хранятся в `.attrs` в виде словаря.

```
layers = soup.find_all("layer")
for lr in layers:
    metadata = {}
    for name in lr.find_all("name"):
        val = name.find_next_sibling("val")
        metadata["".join(name.contents)] = "".join(val.contents)
    if metadata.get("layer-type") != "words":
        continue
    events = lr.find_all("event")
    for ev in events:
        print(ev.attrs["start"], ev.attrs["end"], "".join(ev.contents))
```

Задание для выполнения в классе: переберите все интервалы на уровне "syll" и создайте файл .TextGrid на его основе, предварительно преобразовав транскрипцию из X-SAMPA в IPA.

Домашнее задание: основываясь на записях fpt1, попробуйте определить, есть ли зависимость между значениями формант и значениями признаков гласных hi, lo, back. Постройте соответствующие графики.