I. Работа с файловыми системами: модули os и glob

Файловая система:

```
"""
C:
├─ Program Files
├─ Folder_1
│  ├─ Folder_2
│  │  ├─ file_3.docx
│  ├─ file1.txt
│  ├─ file2.txt
├─ file4.png
"""
```

Имя файла - это строка*

Часть строки между последней точкой и концом - его расширение

Расширение связано с содержанием файла только по договорённости

```
"file1.txt"
"file2.py"
"file3.py.txt"
"file4"
```

---

* имя файла нужно отличать от переменной, которая работает непосредственно с файлом:

```
with open("lol.txt", "w") as f:
    print(type(f))
```

В Windows имена папок и файлов разделяются между собой обратным слешем \. В Unix, Max - прямым слешем /.

Как задавать пути в Python? (Помним, что \ нужен ещё и для escape-последовательностей!)

```
r"C:\Folder_1\file1.txt"
"C:\\Folder_1\\file1.txt"
"C:/Folder_1/file1.txt"  # будет автоматически переформатировано под формат Windows
```

Пути бывают относительные и абсолютные. Абсолютные указывают путь от корневой папки (или от буквы диска в Windows). Относительные указывают путь от текущей папки.

```
# если текущая папка - Folder_1
"Folder_2/file_3.docx"
r"../file4.png"  # переход на один уровень вверх
```

Модуль os и os.path

https://docs.python.org/3/library/os.html

https://docs.python.org/3/library/os.path.html

```
import os

filepath = os.path.join("Folder_1", "file1.txt")
print(filepath)
```

```
path, file = os.path.split("C:/Folder_1/file1.txt")
print(path)
print(file)
```

```
print(os.sep)  # системный разделитель путей
```

```
file, ext = os.path.splitext("file1.txt")
print(file)
print(ext)
```

Проверить, что путь существует:

```
os.path.exists("Folder_1")
```

Создать директорию:

```
os.makedirs("Folder_1/Folder_2")  # ошибка, если уже существует
```

Перебираем все файлы внутри папки

```
os.listdir("sample_data")
```

Рекурсивный обход файлового дерева:

```
import os
for root, dirs, files in os.walk("."):  # текущая директория; сюда можно передавать относите
    print(root, dirs, files)
```

Модуль glob

https://docs.python.org/3/library/glob.html

```
import glob
```

```
glob.glob("sample_data/*.csv")  # маска файла
```

С рекурсией:

```
glob.glob("./**/*.csv", recursive=True)
```

Задание для выполнения в классе:

1. Загрузить в среду архив `cta_seg.zip` (доступ только по st-адресу!)
2. Распаковать, запустив ячейку ниже

```
!unzip cta_seg.zip
```

3. Написать функцию, которая принимает на вход имя папки (в данном случае `cta_seg`) и уровень seg (в данном случае `B1` или `Y1`), ищет там все файлы `.seg` соответствующего уровня и возвращает их список.

4. Модифицировать функцию так, чтобы она открывала каждый найденный файл `.seg`, читала оттуда названия меток (кроме пустых) и записывала их в текстовый файл - метки из каждого файла `.seg` на отдельной строке, разделённые пробелами.

```
cta_seg_B1.txt
```

```
j u0 r' i4 t r' i0 f a4 n a4 f
a1 b m' e0 n
...
```

II. Работа с многоуровневой разметкой

Задание для выполнения в классе:

1. Написать функцию, которая принимает на вход два имени файлов `.seg` и возвращает список списков следующего формата:

```
[
 [[метка1_1, метка1_2], [метка2_1, метка2_2, ...]],
 [[метка1_2, метка1_3], [метка2_5, метка2_6, ...]],
 ...
]
```

   где `метка1_*` - это метки из первого файла, а `метка2_*` - метки из второго, которые лежат **между** соответствующими метками из первого. Используйте циклы.

Считайте, что интервал, ограниченный двумя метками, полуоткрытый - т.е. если `метка2_2` совпадает с меткой `метка1_1`, она будет входить в интервал [`метка1_1`, `метка1_2`), а если совпадает с меткой `метка1_2` - не будет.

2. Модифицируйте функцию так, чтобы результат она записывала в текстовый файл в следующем формате:

```
<время метки1_1 в секундах> <время метки1_2 в секундах> <имя метки1_1> <имя метки2_1>
```

Например:

```
0.0 0.412 юрий j u0 r' i4
0.412 1.24 трифонов t r' i0 f a4 n a4 f
```

Домашнее задание:

Напишите программу, которая обрабатывает корпус файлов `.seg` и строит на его основе произносительный словарь. Результат необходимо записать в текстовый файл в алфавитном порядке. Например:

```
...
обмен a1 b m' e0 n
...
трифонов t r' i0 f a4 n a4 f
...
юрий j u0 r' i4
...
```

Помните, что у вас обязательно возникнут слова с одинаковым написанием, но разным произношением! Каждый вариант нужно записать на своей строчке, но только один раз! Абсолютно одинаковых строк в итоговом файле быть не должно:

```
...
в f
в v
...
```