Соответствие уровней и их кодов

```
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)}
```

Функция для чтения файлов .seg

```
def read_seg(filename: str, encoding: str = "utf-8-sig") -> tuple[dict, list[di
    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_CHAN
            "level": code_to_level[int(level)],
            "name": name
        }
        labels.append(label)
    return params, labels
```

Функция для записи файлов .seg (желательно написать свою, не подсматривая)

```
def write_seg(params: dict, labels: list, filename: str, encoding: str = "utf-8-sig") -> Nor
    # зададим значения параметров по умолчанию
    # вы можете изменить функцию так, чтобы параметры можно было передавать как ключевые слс
```

```
    param_defaults = {
        "SAMPLING_FREQ": 44100,
        "BYTE_PER_SAMPLE": 2,
        "CODE": 0,
        "N_CHANNEL": 1,
        "N_LABEL": 0
    }
    # запишем в словарь переданные в функцию значения параметров
    param_defaults.update(params)
    # количество меток определим как длину списка labels
    param_defaults["N_LABEL"] = len(labels)
    with open(filename, "w", encoding=encoding) as f:
        f.write("[PARAMETERS]\n")
        for key, value in param_defaults.items():
            f.write(f"{key}={value}\n")
        f.write("[LABELS]\n")
        for label in labels:
            f.write(f"{param_defaults['BYTE_PER_SAMPLE'] * param_defaults['N_CHANNEL'] * lab
            f.write(f"{level_to_code[label['level']]},")
            f.write(f"{label['name']}\n")
```

Функция для вывода пар меток на экран

```
def print_label_pairs(filename):
    params, labels = read_seg(filename)
    for start, end in zip(labels, labels[1:]):
        print(start, end)
```

Улучшим функцию, чтобы для каждого интервала печаталась только релевантная информация (начало, конец, имя открывающей метки)

```
def print_intervals(filename):
    params, labels = read_seg(filename)
    for start, end in zip(labels, labels[1:]):
        print(start["name"], start["position"], end["position"])
```

Можно использовать функцию pairwise() (https://docs.python.org/3/library/itertools.html#itertools.pairwise) из модуля itertools (в версии Python 3.10 и позже):

```
from itertools import pairwise
def print_intervals(filename):
    params, labels = read_seg(filename)
    for start, end in pairwise(labels):
        print(start["name"], start["position"], end["position"])
```

```
!wget https://pkholyavin.github.io/mastersprogramming/cta0001.seg_B2


print_intervals("cta0001.seg_B2")
```