Проверочная

Форма сдачи

Создайте папку на рабочем столе вашего компьютера, которая называется Фамилия_Имя (например, *Гослинг_Райан* или *Булгаков_Михаил*). В этой папке могут находиться:

- одна IPYNB-тетрадка `tasks.ipynb`, которая **явно** разделена на блоки по заданиям (например, так)
- три PY-файла: `task1.py`, `task2.py` и `task3.py`, - в каждом из которых находится код к соответствующему заданию

NB! Не забывайте про кодстайл: `.md` и `.pdf`!

Задачи

Задача №1

Одной из задач NLP является NER (распознавание именованных сущностей, *named entity recognition*). В общем смысле она сводится к выделению и классификации слов и словосочетаний, обозначающих людей, географические названия, события и т.д. Например, в предложении "*Гоголь умер в Москве.*" должны быть выделены две именованные сущности: "*Гоголь*" и "*Москва*". Обычно для решения задач NER используются нейросети, однако вам предлагается сделать собственное распознавание именованных сущностей при помощи `pymorphy2`.

У вас есть TXT-файл с текстом "Мёртвых душ" Н. В. Гоголя, из которого вам надо выделить все **имена, фамилии и места**. Создайте три частотных словаря, в которые записывайте соответствующие леммы (начальные формы) подходящих слов и количество раз, что они встретились в тексте. Оставьте лишь те именованные сущности, которые встретились в тексте 5+ раз и запишите их в три отсортированные по убыванию частотности CSV-файла соответственно: `names.csv`, `surnames.csv` и `places.csv`.

P.S. Дополнительный балл вы можете получить, если при проверке также будете учитывать, что оценка вероятности того, что данный разбор правильный, должна быть больше или равна **0.5**.

Задача №2

Ограничение по памяти	256 мегабайт

Ограничение по времени	2 секунды

Жадной суммой элементов массива назовём число C, получаемое следующим образом:

1. Создаётся счётчик c.
2. Совершается проход по массиву, и для каждого элемента массива a_i возможен один из двух вариантов действий:
   - $c = c + a_i$
   - $c = |c + a_i|$
3. C — максимально возможное значение счётчика c.

На стандартный поток ввода подаётся число n ($1 \leq n \leq 10^4$). Затем поступает ввод n массивов различной длины, каждый на отдельной строке. Длина каждого массива не превосходит 10^5 Для каждого массива a необходимо вывести его жадную сумму.

Ввод	**Вывод**
4 10 -7 -5 4 1 4 3 4 1 4 3 4 -5 -7 -6 -1000000 1000000 1000000 1000000	6 24 18 4000000

NB! При решении задачи запрещается использовать сторонние библиотеки.

Задача №3

По адресу `95.165.90.137:9235/api/random_strings/{num-strings}` находится некий API-сервис. В ответ на запрос GET он присылает JSON, в котором по ключу `strings` лежит массив строк. Количество строк в массиве равно `num-strings`. Массив обладает случайной вложенностью, то есть содержит не сами строки, а другие массивы случайной вложенности, также содержащие строки. Ваша задача — написать **генератор**, "выпрямляющий" этот список. Например, если назвать его `string_gen`, то следующий код:

```
arr = ["a", "b", ["c"], [["d", "e"], "f"]
for string in string_gen(arr):
    print(string)
```

должен вывести:

```
a
b
```

```
c
d
e
f
```

Затем нужно значения из этого генератора сохранить сначала в список, и в множество. Пользуясь известными вам средствами профилирования памяти и времени, замерьте среднее время добавления в коллекцию и память, занимаемую коллекцией, для `num-strings`, равного 10, 1000 и 100000.

Пользуясь своими знаниями об устройстве контейнеров `list` и `set`, объясните разницу полученных результатов в комментарии к программе.