ЛР2 Базовые элементы языка Python

Формат сдачи: При выполнении ЛР вам необходимо создать репозиторий на GitHub (или GitLab) с названием labs_python. В репозитории под каждую ЛР (не задание, а ЛР!) заводится отдельная папка с именим lab_[номер ЛР] (например, для данной ЛР – это lab_2). В этой папке должен находиться результат выполнения каждого задания – отдельный файл с расширением *.py. Каждый файл с выполненым задание должен иметь следующие название [номер_студента_в_общем_списке_группы]_lab_[номер_ЛР]_[номер_задания].py (например, 1_lab_2_1.py) (невыполнение этих простых правил влечет анулирование баллов за задание). Во время защиты необходимо продемонстрировать работу вашей программы, а также ответить на вопросы преподавателя. Максимальный балл за задание ставится в случае если студент правильно выполнил задание, его код соответствуют PEP-8, стиль кода соответствует pythonic стилю, задание выполнено наиболее оптимальным образом и студент прошел успешно защиту на паре.

Требования к выполнению заданий:

ARGS необходимо организавать возможность запуска выполненного задания с аргументами командной строки.

MODULE необходимо реализовать возможность использовать ваш исполняемый файл как модуль.

INTER необходимо реализовать возможность интрерактивной работы с программой (при этом все должно быть интуитивно понятно пользователю).

FILE необходимо реализовать возможность ввода и/или вывода с помощью файлов.

STATUS необходимо реализовать возможность интерактивного отображения статуса выполнения программы (например, в процентах).

CHECK необходимо реализовать проверку корректности вводимых данных/параметров.

Возможность использования дополнительных библиотек для выполнения заданий спрашивайте у лектора!

Срок сдачи: до 05.04.2019 (может меняться)

Task 1. (5 баллов). Sqrt-декомпозиция [INTER] [FILE] [CHECK]

Напишите программу, реализующую метод sqrt-декомпозици для запросов типа "сумма на отрезке массива от l до r". Массив чисел и потом запросы вводятся пользователем интерактивно или через файл.

Task 2. (5 баллов). Генератор больших файлов [ARGS] [MODULE] [STATUS] [CHECK]

Реализовать программу генерирующую текстовый файл. На вход программы поступает строка с названием выходного файла и одно (два или три) числа – Mb[, K][, L]; где Mb – размер выходного файла в МегаБайтах, K – количество слов в строке (слова разделены пробелами) и L – длина каждого слова. При этом K – tupple из двух чисел (a, b), для каждой новой строки количество слов в строке – это случайное число от a до b; по умолчанию K=(10, 100) и L – tupple из двух чисел (a, b), для каждого нового слова длина – это

случайное число от a до b; по умолчанию L=(3, 10). (если a=b, то длина строки или длина слова евляется постоянной величиной). Генерация файла останавливается, когда достигнут необходимый размер файла. Слова состоят только из букв латинского алфавита (как больших, так и маленьких)

Task 3. (5 баллов). Merge it
[ARGS] [MODULE] [STATUS] [FILE]

Реализовать программу сортировки строк файла и создание файла содержащий отсортированный файл по строкам, а каждая строка отсортирована по словам. Использовать для сортировки алгоритм Merge Sort (реализовать его самим). **Важно: у вас есть ограничение по оперативной памяти для программы - 400MB. А поступающие на сортировку файлы могут весить 1GB.**

Task 4. (5 баллов). Flatten
[ARGS] [MODULE]

Необходимо написать функцию-генератор flatten_it, которая принимает на вход любой итерируемый аргумент и линеаризует его, т.е. нужно рекурсивно проходиться по элементам из этого итерируемого объекта, которые также могут являются итерируемыми объектами, и так далее до произвольной глубины вложенности, пока не дойдем до неитерируемых "листьев". Ваша функция в итоге должна являться итератором по всем "листьям". Обратите внимание, что возможны циклы по вложенности, вы должны уметь обрабатывать такие ситуации и бросать исключение ValueError в этом случае. (Пример – на входе список [1, 2, [3, 4, 5], [6, [7, 8]]], на выходе [1, 2, 3, 4, 5, 6, 7 ,8])

Task 5. (5 баллов). Object to JSON
[MODULE] [FILE]

Реализовать функцию to_json(obj), которая на вход получает python-объект, а на выходе у неё строка в формате JSON. Если при вызове to_json был передан (или обеъкт, содержащий) тип, которого не может быть преобразован в JSON, функция выбрасывает исключение ValueError. JSON-сереализуемые объекты: dict, list, int, float, None, bool, str, tuple (рассматривается как list). **Важно: использовать функции, например: str, __repr__ и т.д. – НЕЛЬЗЯ!**

Task 6. (5 баллов). JSON to Object
[MODULE] [FILE]

Реализовать функцию from_json(text), которая возвращает python-объект соответствующий json-строке. **Важно: использовать функции, например: str, __repr__ и т.д. – НЕЛЬЗЯ! А также использовать стандартные инструменты работы с JSON НЕЛЬЗЯ!**

Task 7. (5 баллов). Числа Леонардо
[MODULE] [INTER] [ARGS] [CHECK]

Реализовать программу определения n-ого числа Леонардо. Про числа Леонардо можно прочитать по ссылке – https://en.wikipedia.org/wiki/Leonardo_number

Task 8. (5 баллов). 2^n
[MODULE] [INTER] [ARGS] [CHECK]

Написать программу, которая позволяет по заданному числу ответить на вопрос: яв-

ляется ли число точной степенью двойки. **Операцией возведения в степень пользоваться нельзя!**

Task 9. (Дополнительное). Собираем статистики по тексту [MODULE] [INTER] [ARGS] [FILE]

На вход поступают текстовые данные (использовать ввод через файл). Необходимо посчитать и вывести:

1. сколько раз повторяется каждое слово в указанном тексте
2. среднее количество слов в предложении
3. медианное количество слов в предложении
4. top-K самых часто повторяющихся буквенных N-грам (K и N имеют значения по-умолчанию 10 и 4, но должна быть возможность задавать их с клавиатуры)

При решении использовать контейнер dict() или его аналоги и встроенные операции над строками. Предусмотреть обработку знаков препинания. При тестировании использовать генератор текстового файла из Задания 2.

Task 10. (Дополнительное). Хранилище [INTER]

При запуске программа работает в интерактивном режиме и поддерживает команды:

1. add <key> [<key> . . .] - добавить один или более элементов в хранилище (если уже содержится, то не добавлять).
2. remove <key> - удалить элемент из хранилища.
3. find <key> [<key> . . .] - проверить наличие одного или более элементов в хранилище, вывести найденные.
4. list - вывести все элементы в хранилище.
5. grep <regexp> - поиск значения по регулярному выражению.
6. save и load - сохранить хранилище в файл и загрузить хранилище из файла

При решении использовать контейнер set().