Структура файла данных:

$

Question lines

….

Question Lines

%

Answer Lines

….

Answer Lines

@

Data Line: **right\_count** (Int): Number of repetitions without errors <Space> **last\_date** (DD.MM.YYYY) of the last repetition

$

.

.

.

& - Finish of the payload, further lines - comments

И вопрос и ответ должны иметь возможность быть в файле (двух файлах). Вопрос выводится на экран из самого начала файла, забирая верхние строки в тройных кавычках или между /\* и \*/ или хранится в основном файле данных или в виде картинки или в виде отдельного файла. В качестве ответа выводится сам файл в «блокноте» или картинка или ответ из основного файла.

Реализуем примитивный, но реальный алгоритм

1. В ходе сеанса поддерживаем два списка:
   1. **items –** список всех содержащихся в файле данных вопросов (пока так, в дальнейшем возможно ограничимся только «активными» вопросами – нужно для записи данных в конце сеанса.

**assignments** – список заданий текущего сеанса

* 1. Структура **item:** (список в первой версии, потом преобразуем в класс:
     1. question: str (multiline)
     2. answer: str (multiline)
     3. right count (int):
     4. last date year (int)
     5. last date month (int)
     6. last date day (int)

1. Считываем весь файл с данными в **items** (надо будет добавить обработку ошибок – чтобы если ошибка в текущем unit – читать до следующего $
2. По ходу считывания - Если **date** + **pauses**[**right\_count**] >= Today – включаем **next\_item** в пул вопросов (**assignments**) увеличиваем количчество вопросов сеанса **total\_assignments**
3. Включаем в пул не более **chunk\_count** вопросов, поддерживаем список активных в данном тестировании вопросов **assigned\_items**, содержащий [индекс вопроса в общем списке вопросов **item\_ndx**, задаем ли вопрос при следующем проходе списка **still\_active** 1 / 0, первоначально все – 1]
4. Начинам тестирование. **solved\_assignments = 0**
5. При неправильном ответе сбрасываем значение **right\_count** в ноль
6. Если достигнуто **right\_answers** правильных ответов в течение сеанса и неправильных ответов не было – добавляем к **right\_count** 1, и исключаем Item из Assignments (ставим **still\_active** в **assigned\_items** в 0), увеличиваем **solved\_assignments** на единицу.
7. Если был неправильный ответ, устанавливаем **right\_count** в 0
8. Если достигнуто **right\_after\_wrong\_answers** – просто исключаем Item из Assignments (ставим **still\_active** в **assigned\_items** в 0), увеличиваем **solved\_assignments** на единицу.
9. После завершения ответа на каждый вопрос данные в **items** должны быть в состоянии, готовом для записи в файл данных.
10. Продолжаем так до того момента, пока остались неотвеченные до конца вопросы или до того момента, пока не будет ручной брейк.

Структура списка **assigned\_items [index, still\_active, total\_answers, continuous\_right\_answers]**

Соответственно начальная структура файла конфигурации:

1. Файл для чтения / записи (MyAnkiData.txt)
2. Размер изучаемой порции, Chunk (количество вопросов) (30)
3. Необходимое количество правильных ответов в течение сеанса – с нуля (1)
4. Необходимое количество правильных ответов в течение сеанса – после неправильного ответа (например 1 или 2)
5. Кортеж pauses = (0, 1, 2, 4, 8, 16, 32, 64,) (Например, такой) – кортеж для Питона