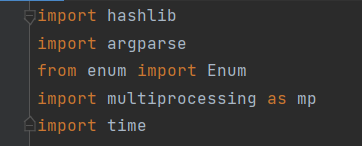
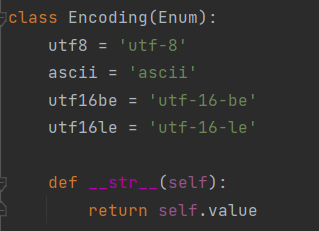
Руководство по использованию утилиты «Восстановление паролей»

Общее описание программы:

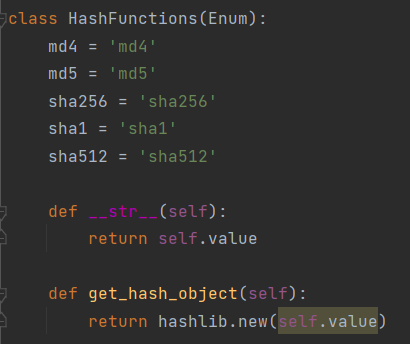
* Подключаем необходимые библиотеки для обработки текстового файла с паролями, файла с хешами, генерации хеш-значений, парсинга аргументов, параллельной обработки данных и замера времени работы:



* Создаем класс с описанием используемых кодировок, которые являются атрибутами класса. Наследуем этот класс от суперкласса Enum, чтобы иметь возможность работать с перечислениями.



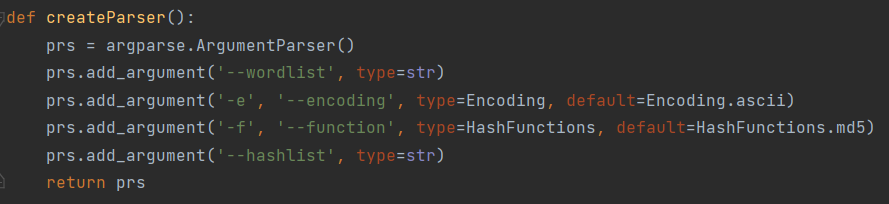
* Создаем класс с описанием хэш-функций, которые являются атрибутами класса. Наследуем этот класс от суперкласса Enum, чтобы иметь возможность работать с перечислениями, а также создаем метод-конструктор для применения определенного алгоритма хеширования.



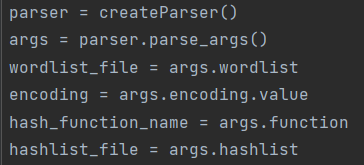
* Определяем количество используемых рабочих процессов:



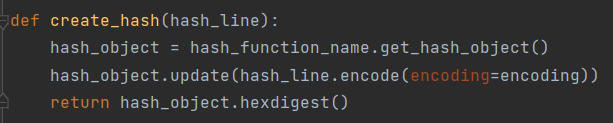
* Создаем функцию для обработки параметров командной строки: создаем экземпляр класса ArgumentParser, добавляем в него информацию об ожидаемых параметрах, возвращаем созданный объект



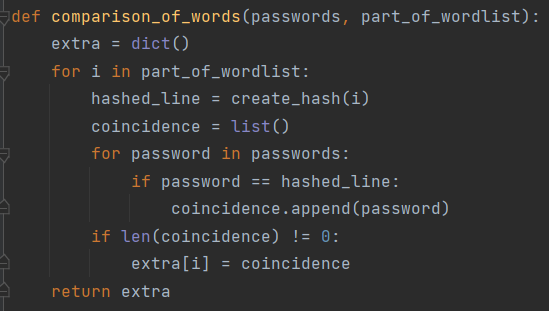
* Разбираем командную строку на параметры, и используем эти параметры в основной части программы



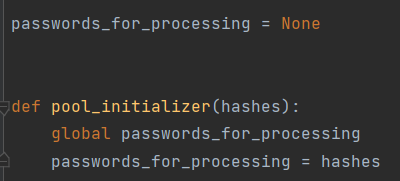
* Создаем функцию, которая принимает на входе строку с исходным паролем из файла и с помощью методов, определенных ранее, высчитывает хеш-значение и возвращает строку Hex, что и является хешем:



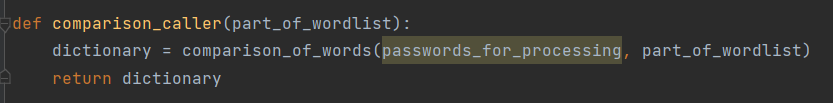
* Создаем вспомогательную функцию, производящую сравнение блока паролей-кандидатов с хеш-значениями из другого файла, а также возвращающая найденную пару ключ-значение в случае совпадения:



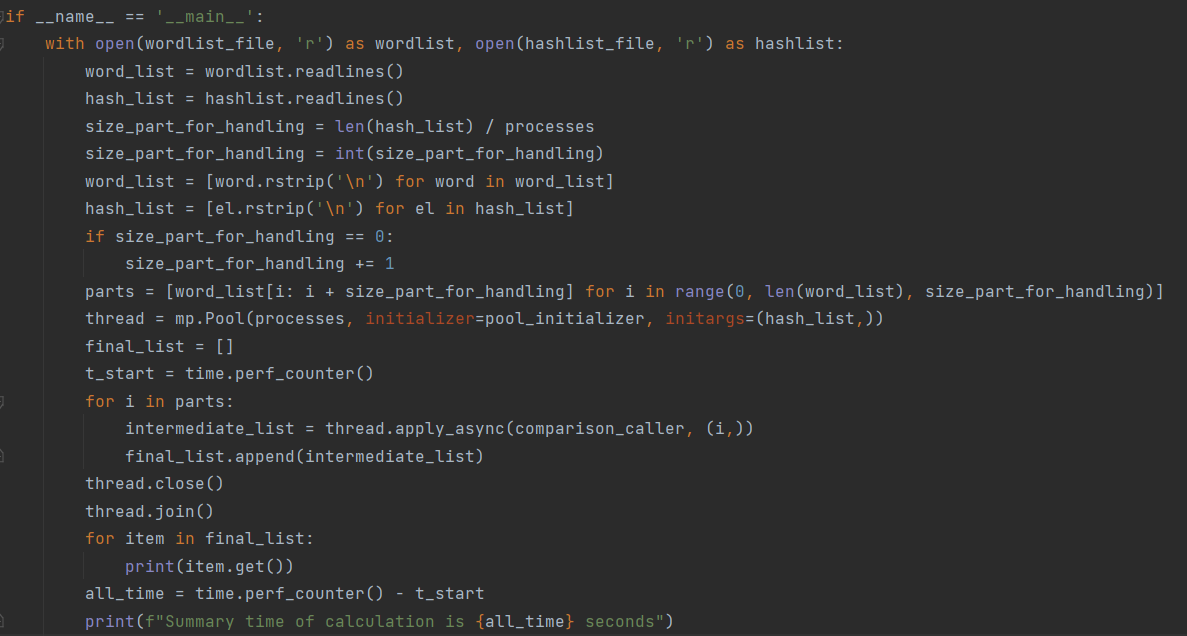
* Определяем глобальную переменную, отвечающую за список хеш-значений из входного файла, с которым мы сравниваем блоки наших паролей-кандидатов. Она используется при создании объекта, управляющего пулом рабочих процессов:



* Создаем вспомогательный вызов функции сравнения для передачи ее в параметры метода асинхронной обработки данных объекта Pool:



* В основной функции открываем оба входных файла на считывание, преобразуем данные обоих файлов в список, определяем размер блока, который будем обрабатывать параллельно на нескольких ядрах, и с помощью объекта Pool, реализующего асинхронное выполнение задач:



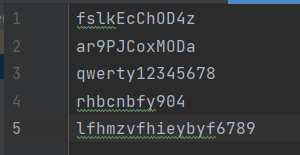
Тестовые примеры и сценарии использования:

1. На вход подается текстовый файл с 5 паролями, файл с хеш-значениями, количество которых = 10. Так как он был сгенерирован утилитой-генератором хешей, первые 5 хеш-значений соответствуют нашим 5 паролям, остальные псевдослучайны. Кодировка – utf-8, режим шифрования – md5. Проверим корректность работы программы:

Запускаем утилиту с параметрами командной строки:



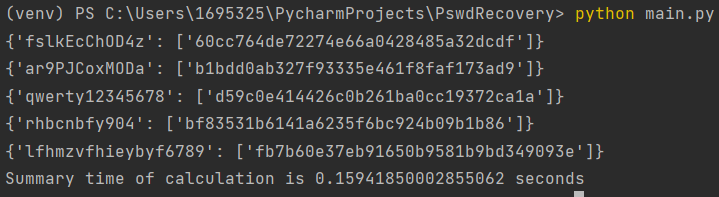
Исходный файл с паролями:



Исходный файл с хешами:



Результат выполнения программы:



Результат полностью соответствует ожиданиям.

1. Анализ скорости работы утилиты:

* Исходный файл с паролями содержит 100665 слов
* Исходный файл с хешами содержит 254387 слов
* Время работы утилиты = 185 секунд
* Количество кандидат в секунду = 544

Данные о процессоре:

Intel Core i7 10750H

6 ядер

12 потоков