Лабораторная работа №2

Основные элементы языка Python

Темы:

1. Типы данных.
2. Строки и форматирование.
3. Ввод вывод.
4. Модули.
5. Обработка ошибок.
6. Классы и метаклассы.
7. Генераторы и итераторы.
8. Декораторы.
9. Дескрипторы.

Критерии приема задач:

- Решение поставленной задачи.
- Хорошее оформление и читаемость кода.
- Теория по теме задач.

Требования:

- Каждое задание - отдельный модуль, который можно вызывать как файл или импортировать и использовать как библиотеку.
- Там где это нужно, создать свои классы исключения для сигнализтварония о специфических ошибках или использовать уже существующие. Обрабатывать ошибки вызываемых библиотек.
- В заданиям нельзя просто использовать встроенные аналоги запрашиваемых элементов из языка или библиотек надо реализовывать самостоятельно. Пример: в json преобразователе нельзя использовать встроенный модуль json.

Основные задания:

1. Сортировка слиянием во внешней памяти.
   На вход поступает файл с числами в произвольном формате. Размер файла должен быть достаточно большим, чтобы на обычную сортировку слиянием не хватало оперативной памяти. На выходе получается файл с этими числами, упорядоченными по возрастанию. Промежуточные результаты хранить во временных файлах (рекомендуется модуль tempfile). Временные файлы после использования удаляются.
   Создать большой файл с числами можно таким скриптом
   ```
   import random
   with open('numbers.txt', 'w') as f:
       f.writelines('{}\n'.format(random.randint(-1000000, 1000000)) for _ in range(500000000))
   ```
2. Свой преобразователь в JSON. Реализовать функцию to_json(obj), которая на вход получает python объект, а на выходе у неё строка в формате JSON.
3. Класс “n мерный вектор”. У этого класса должны быть определены все естественные для вектора операции - сложение, вычитание, умножение на константу и скалярное произведение, сравнение на равенство. Кроме этого должны быть операции вычисления длины, получение элемента по индексу, а также строковое представление.
4. Декторатор @cached, который сохраняет значение функции при каждом вызове. Если функция вызвана повторно с теми же аргументами, то возвращается сохраненное значение, а функция не вычисляется.
5. Юниттесты. Использовать любой фреймворк для тестирования (unittest, nose, pytest). Использовать модуль coverage для оценки покрытия кода тестами. На каждое основное задание написать 2+ тестов (корректная работа, некорректная работа).

Дополнительные задания:

1. Свой преобразователь из JSON. Реализовать функцию from_json(text), которая возвращает pythonобъект соответствующий jsonстроке. Не использовать стандартные инструменты работы с JSON. (4 балла)
2. Синглтон. Реализовать шаблон проектирования Singleton, который можно применять на произвольный класс. Разработать самостоятельно, как этот инструмент будет применяться к целевому классу (например, модифицировать исходный класс или изменять способ вызова конструктора). (2 балла)
3. Реализовать свой устанавливаемый (setup.py) пакет со всеми заданиями этой лабораторной, разбитой на модули. Описать зависимости, указать скрипты для запуска заданий из каждого пункта. (2 балла)