Погружение в Python (семинары)

Задание 1. Работа с основными данными

Напишите функцию, которая получает на вход директорию и рекурсивно обходит её и все вложенные директории. Результаты обхода сохраните в файлы json, csv и pickle. Для дочерних объектов указывайте родительскую директорию. Для каждого объекта укажите файл это или директория. Для файлов сохраните его размер в байтах, а для директорий размер файлов в ней с учётом всех вложенных файлов и директорий. Соберите из созданных на уроке и в рамках домашнего задания функций пакет для работы с файлами разных форматов.

Подсказка № 1

Для рекурсивного обхода используйте функцию os.walk(). Эта функция генерирует имена файлов и директорий в указанной директории и ее поддиректориях. Внутри цикла можно разделять файлы и директории и собирать информацию о них.

Подсказка № 2

Используйте os.path.getsize() для определения размера файла. Эта функция возвращает размер файла в байтах. Для директорий вы можете использовать рекурсивный обход для вычисления общего размера всех вложенных файлов.

Подсказка № 3

Для сбора информации о каждом объекте создайте словарь. Словарь должен содержать такие ключи, как `'name'`, `'path'`, `'type'`, `'size'`, и `'parent'`. Используйте `os.path.basename()` для получения имени родительской директории.

Подсказка № 4

Сохраняйте данные в разные форматы с помощью соответствующих библиотек. Используйте `json.dump()` для JSON, `csv.DictWriter()` для CSV и `pickle.dump()` для Pickle.

Эталонное решение:

```
import os

import json

import csv
```

```
import pickle


def get_size(path):
    """Возвращает размер файла или директории."""
    if os.path.isfile(path):
        # Если путь - файл, возвращаем его размер
        return os.path.getsize(path)
    elif os.path.isdir(path):
        total_size = 0
        # Если путь - директория, рекурсивно вычисляем размер всех файлов в директории
        for dirpath, _, filenames in os.walk(path):
            for filename in filenames:
                file_path = os.path.join(dirpath, filename)
                total_size += os.path.getsize(file_path)
        return total_size


def traverse_directory(directory):
    """Рекурсивно обходит директорию и возвращает информацию о файлах и директориях."""
    result = []
    # Обход директории с помощью os.walk, который возвращает корневую директорию, поддиректории и файлы
    for root, dirs, files in os.walk(directory):
        for name in dirs + files:
            path = os.path.join(root, name)
            is_dir = os.path.isdir(path)
            size = get_size(path)
```

```
            parent = os.path.basename(root)

            # Добавление информации о текущем объекте в список
результата

            result.append({

                'name': name,

                'path': path,

                'type': 'directory' if is_dir else 'file',

                'size': size,

                'parent': parent

            })

    return result


def save_to_json(data, filename):

    """Сохраняет данные в формате JSON."""

    with open(filename, 'w') as json_file:

        json.dump(data, json_file, indent=4)


def save_to_csv(data, filename):

    """Сохраняет данные в формате CSV."""

    with open(filename, 'w', newline='') as csv_file:

        writer = csv.DictWriter(csv_file, fieldnames=['name', 'path',
'type', 'size', 'parent'])

        writer.writeheader()

        writer.writerows(data)


def save_to_pickle(data, filename):

    """Сохраняет данные в формате Pickle."""
```

```
    with open(filename, 'wb') as pickle_file:
        pickle.dump(data, pickle_file)


def main(directory):
    """Основная функция, которая выполняет обход директории и
сохраняет результаты."""
    data = traverse_directory(directory)


    save_to_json(data, 'directory_info.json')
    save_to_csv(data, 'directory_info.csv')
    save_to_pickle(data, 'directory_info.pkl')


if __name__ == "__main__":
    # Замените 'your_directory' на путь к вашей директории
    main('your_directory')
```

Задача 2. Объединение данных из нескольких JSON файлов

Напишите скрипт, который объединяет данные из нескольких JSON файлов в один. Каждый файл содержит список словарей, описывающих сотрудников компании (имя, фамилия, возраст, должность). Итоговый JSON файл должен содержать объединённые списки сотрудников из всех файлов.

Пример: У вас есть три файла `employees1.json`, `employees2.json`, `employees3.json`. Нужно объединить их в один файл `all_employees.json`.

Подсказка № 1

Используйте функцию `glob.glob()` для поиска всех JSON файлов в указанной директории.

Подсказка № 2

Откройте каждый JSON файл с помощью json.load() и добавьте данные в общий список. Функция `json.load()` позволяет прочитать содержимое JSON файла и преобразовать его в Python объект. Используйте `list.extend()` для объединения данных.

Подсказка № 3

Сохраните объединенные данные в новый JSON файл с помощью json.dump(). После объединения данных, используйте `json.dump()` для записи списка в новый JSON файл.

Эталонное решение:

```
import json
import glob


def merge_json_files(input_files, output_file):
    """Объединяет данные из нескольких JSON файлов в один."""
    merged_data = []  # Список для хранения объединенных данных
    for file in input_files:
        try:
            with open(file, 'r') as f:
                data = json.load(f)  # Чтение данных из файла
                merged_data.extend(data)  # Добавление данных в
общий список
        except json.JSONDecodeError:
            print(f"Ошибка чтения JSON файла: {file}")


    with open(output_file, 'w') as f:
        json.dump(merged_data, f, indent=4)  # Сохранение
объединенных данных в новый файл


if __name__ == "__main__":
```

```
    # Получаем все JSON файлы в текущей директории

    json_files = glob.glob('employees*.json')

    merge_json_files(json_files, 'all_employees.json')
```

Задача 3. Агрегирование данных из CSV файла

Напишите скрипт, который считывает данные из JSON файла и сохраняет их в CSV файл. JSON файл содержит данные о продуктах (название, цена, количество на складе). В CSV файле каждая строка должна соответствовать одному продукту.

Пример: Из файла `products.json` нужно создать `products.csv`.

Подсказка № 1

Используйте json.load() для чтения данных из JSON файла. Функция `json.load()` позволяет загрузить данные из JSON файла в виде Python объекта, например, списка словарей.

Подсказка № 2

Используйте `csv.DictWriter` для записи данных в CSV файл. Функция `csv.DictWriter` позволяет записывать данные в CSV файл, где каждый словарь из списка становится одной строкой в CSV.

Подсказка № 3

Обеспечьте правильное управление строками в CSV файле. При записи в CSV файл используйте параметр `newline=''` в `open()`, чтобы избежать дополнительных пустых строк между записями на Windows.

Эталонное решение:

```
import json

import csv



def json_to_csv(json_file, csv_file):

    """Превращает данные из JSON файла в CSV файл."""

    # Чтение данных из JSON файла

    with open(json_file, 'r') as f:
```

```
        data = json.load(f)  # Загрузка данных из JSON



    # Проверка корректности формата данных

    if not isinstance(data, list) or not all(isinstance(item, dict)
for item in data):

        raise ValueError("Некорректный формат данных в JSON файле")



    # Запись данных в CSV файл

    with open(csv_file, 'w', newline='') as f:

        fieldnames = data[0].keys()  # Получение заголовков из
ключей первого словаря

        writer = csv.DictWriter(f, fieldnames=fieldnames)

        writer.writeheader()  # Запись заголовков

        writer.writerows(data)  # Запись данных



if __name__ == "__main__":

    json_to_csv('products.json', 'products.csv')
```

Задача 4. Агрегирование данных из CSV файла

Напишите скрипт, который считывает данные из CSV файла, содержащего информацию о продажах (название продукта, количество, цена за единицу), и подсчитывает общую выручку для каждого продукта. Итог должен быть сохранён в новом CSV файле.
Пример: Из файла sales.csv нужно создать файл total_sales.csv, где для каждого продукта будет указана общая выручка.

Подсказка № 1

Используйте csv.DictReader для чтения данных из исходного CSV файла. `csv.DictReader` позволяет читать строки CSV файла как словари, где ключи соответствуют заголовкам столбцов.

Подсказка № 2

Создайте словарь для хранения выручки по каждому продукту. Используйте продукт в качестве ключа и выручку в качестве значения. Убедитесь, что добавляете выручку при встрече одинакового продукта.

Подсказка № 3

Используйте csv.DictWriter для записи данных в новый CSV файл. Запишите итоговые данные в новый файл, указывая заголовки столбцов и записывая итоговую выручку для каждого продукта.

Подсказка № 4

Преобразуйте данные в числовые типы для корректного вычисления выручки. Убедитесь, что данные из CSV преобразованы в целые или вещественные числа, чтобы корректно производить арифметические операции.

Эталонное решение:

```
import csv



def calculate_total_sales(input_file, output_file):

    sales_totals = {}  # Словарь для хранения общей выручки по
каждому продукту



    # Чтение данных из исходного CSV файла

    with open(input_file, 'r') as f:

        reader = csv.DictReader(f)



        for row in reader:

            product = row['название продукта']

            quantity = int(row['количество'])  # Преобразование
количества в целое число

            price_per_unit = float(row['цена за единицу'])  #
Преобразование цены за единицу в вещественное число

            total_sales = quantity * price_per_unit  # Вычисление
общей выручки
```

```
            if product in sales_totals:
                sales_totals[product] += total_sales  # Добавляем к существующей выручке
            else:
                sales_totals[product] = total_sales  # Создаем новую запись в словаре

    # Запись итоговых данных в новый CSV файл
    with open(output_file, 'w', newline='') as f:
        fieldnames = ['название продукта', 'общая выручка']
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writeheader()

        for product, total_sales in sales_totals.items():
            writer.writerow({'название продукта': product, 'общая выручка': total_sales})

if __name__ == "__main__":
    calculate_total_sales('sales.csv', 'total_sales.csv')
```

Задача 5. Конвертация CSV в JSON с изменением структуры данных

Напишите скрипт, который считывает данные из CSV файла и сохраняет их в JSON файл с другой структурой. CSV файл содержит данные о книгах (название, автор, год издания). В JSON файле данные должны быть сгруппированы по авторам, а книги каждого автора должны быть записаны как список.

Пример: Из файла `books.csv` нужно создать файл `books_by_author.json`, где книги сгруппированы по авторам.

Подсказка № 1

Используйте csv.DictReader для чтения данных из CSV файла. Эта функция читает данные из CSV файла и преобразует каждую строку в словарь, где ключи соответствуют заголовкам столбцов.

Подсказка № 2

Создайте словарь, где ключи будут авторами, а значения — списками книг. Используйте словарь для группировки книг по авторам. Для каждого автора создавайте список книг, который будет заполняться по мере чтения CSV файла.

Подсказка № 3

Преобразуйте данные в формат JSON с помощью json.dump(). После того как данные сгруппированы, используйте `json.dump()` для записи данных в файл JSON. Убедитесь, что данные имеют нужный формат и структуру.

Эталонное решение:

```
import csv
import json


def convert_csv_to_json(input_file, output_file):
    books_by_author = {}  # Словарь для хранения книг по авторам

    # Чтение данных из CSV файла
    with open(input_file, 'r') as f:
        reader = csv.DictReader(f)

        for row in reader:
            author = row['автор']
            book = {
                'название': row['название'],
                'год издания': row['год издания']
            }
```

```
            if author in books_by_author:
                books_by_author[author].append(book)  # Добавляем
книгу к существующему автору
            else:
                books_by_author[author] = [book]  # Создаем новый
список книг для нового автора

    # Запись данных в JSON файл
    with open(output_file, 'w') as f:
        json.dump(books_by_author, f, indent=4)  # Сохраняем данные
в JSON формате

if __name__ == "__main__":
    convert_csv_to_json('books.csv', 'books_by_author.json')
```