Лабораторная работа № 9

Тема: Работа с файлами. Библиотека **os**.

Цель работы: научиться работать с файлами в Python.

Работа с файлами в Python включает в себя несколько основных шагов, используя встроенные функции и методы. Вот краткое руководство:

Открытие файла:
Для открытия файла используется функция open(). Эта функция принимает путь к файлу и режим открытия (например, чтение, запись и т.д.).

```
# Пример открытия файла для чтения
file = open('example.txt', 'r')
```

Чтение данных из файла:
Используйте методы чтения, такие как read(), readline(), или readlines() для получения данных из файла.

```
content = file.read()          # Чтение всего содержимого файла
line = file.readline()         # Чтение одной строки
lines = file.readlines()       # Чтение всех строк в виде списка
```

Запись данных в файл:
Для записи данных в файл используйте методы записи, такие как write().

```
with open('example.txt', 'w') as file:
    file.write('Hello, World!\n')
    file.write('Another line.')
```

Закрытие файла:
Важно закрывать файл после завершения работы с ним с помощью метода close().

```
file.close()
```

Контекстный менеджер with:
Рекомендуется использовать контекстный менеджер with для автоматического закрытия файла после выполнения блока кода.

```
with open('example.txt', 'r') as file:
    content = file.read()
```

Режимы открытия файла:

'r': Чтение (по умолчанию).
'w': Запись (если файл не существует, создает новый; если существует, обрезает).
'a': Добавление (если файл не существует, создает новый; если существует, добавляет в конец).
'b': Двоичный режим.
Работа с бинарными файлами:
Для работы с бинарными файлами используйте режим 'b' вместе с методами для чтения/записи бинарных данных (read(), write()).

```python
with open('binary_data.bin', 'rb') as binary_file:
    data = binary_file.read()
```

Библиотека **os**.

Библиотека **os** в Python предоставляет множество функций для работы с операционной системой, включая манипуляции с файлами и директориями. Вот несколько основных функций, касающихся работы с файлами и директориями:

```python
import os

# получение текущей директории
current_directory = os.getcwd()

# Смена текущей директории:
os.chdir('/путь/к/новой/директории')

# Создание директории:
os.mkdir('/путь/новая_директория')

# Создание директории и всех промежуточных директорий (если их нет):
os.makedirs('/путь/к/новой/директории/и/всех/промежуточных')

# Удаление файла:
os.remove('/путь/к/файлу')

# Удаление директории:
os.rmdir('/путь/к/директории')

# Удаление директории и всех её содержимых (включая вложенные директории):
os.removedirs('/путь/к/директории/и/всем/ее/содержимым')

# Переименование файла или директории:
os.rename('/старое/имя', '/новое/имя')
```

```python
#Получение списка файлов и директорий в текущей директории:
file_list = os.listdir()

# Проверка существования файла или директории:
exists = os.path.exists('/путь/к/файлу_или_директории')

# Проверка, является ли путь директорией:
is_directory = os.path.isdir('/путь/к/директории')

# Получение абсолютного пути:
absolute_path = os.path.abspath('относительный/путь')
```

Практика.

Задание 1.
В отдельном файле (recipes.txt) хранится писок рецептов в следующем формате:

Название блюда
Количество ингредиентов в блюде
Название ингредиента | Количество | Единица измерения
Название ингредиента | Количество | Единица измерения
...
Пример:

Омлет
3
Яйцо | 2 | шт
Молоко | 100 | мл
Помидор | 2 | шт

Утка по-пекински
4
Утка | 1 | шт
Вода | 2 | л
Мед | 3 | ст.л
Соевый соус | 60 | мл

Запеченный картофель
3
Картофель | 1 | кг
Чеснок | 3 | зубч
Сыр гауда | 100 | г

Фахитос
5
Говядина | 500 | г
Перец сладкий | 1 | шт
Лаваш | 2 | шт
Винный уксус | 1 | ст.л
Помидор | 2 | шт

Должен получится словарь вида:

```
cook_book = {
  'Омлет': [
    {'ingredient_name': 'Яйцо', 'quantity': 2, 'measure': 'шт.'},
    {'ingredient_name': 'Молоко', 'quantity': 100, 'measure': 'мл'},
    {'ingredient_name': 'Помидор', 'quantity': 2, 'measure': 'шт'}
    ],
  'Утка по-пекински': [
    {'ingredient_name': 'Утка', 'quantity': 1, 'measure': 'шт'},
    {'ingredient_name': 'Вода', 'quantity': 2, 'measure': 'л'},
    {'ingredient_name': 'Мед', 'quantity': 3, 'measure': 'ст.л'},
    {'ingredient_name': 'Соевый соус', 'quantity': 60, 'measure': 'мл'}
    ],
  'Запеченный картофель': [
    {'ingredient_name': 'Картофель', 'quantity': 1, 'measure': 'кг'},
    {'ingredient_name': 'Чеснок', 'quantity': 3, 'measure': 'зубч'},
    {'ingredient_name': 'Сыр гауда', 'quantity': 100, 'measure': 'г'},
    ]
```

}

Задача №2

Нужно написать функцию, которая на вход принимает список блюд из **cook_book** и количество персон для кого мы будем готовить

get_shop_list_by_dishes(dishes, person_count)

На выходе мы должны получить словарь с названием ингредиентов и его количества для блюда. Например, для такого вызова

get_shop_list_by_dishes(['Запеченный картофель', 'Омлет'], 2)

Должен быть следующий результат:

```
{
  'Картофель': {'measure': 'кг', 'quantity': 2},
  'Молоко': {'measure': 'мл', 'quantity': 200},
  'Помидор': {'measure': 'шт', 'quantity': 4},
  'Сыр гауда': {'measure': 'г', 'quantity': 200},
  'Яйцо': {'measure': 'шт', 'quantity': 4},
  'Чеснок': {'measure': 'зубч', 'quantity': 6}
}
```

Обратите внимание, что ингредиенты могут повторяться

Задача №3

В папке (sorted) лежит некоторое количество файлов. Считайте, что их количество и имена вам заранее известны, необходимо объединить их в один по следующим правилам:

1. Содержимое исходных файлов в результирующем файле должно быть отсортировано по количеству строк в них (то есть первым нужно записать файл с наименьшим количеством строк, а последним - с наибольшим)
2. Содержимое файла должно предваряться служебной информацией на 2-х строках: имя файла и количество строк в нем

Пример Даны файлы: 1.txt

Строка номер 1 файла номер 1
Строка номер 2 файла номер 1
2.txt

Строка номер 1 файла номер 2
Итоговый файл:

2.txt
1
Строка номер 1 файла номер 2
1.txt
2
Строка номер 1 файла номер 1
Строка номер 2 файла номер 1