

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Лабораторная работа 2.19

Работа с файловой системой в Python3 с
использованием модуля pathlib

Выполнил студент группы ИВТ-б-о-20-1

Симанский М.Ю « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____

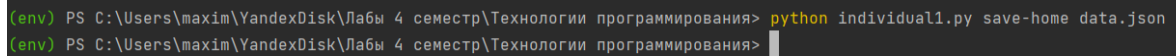
(подпись)

Цель работы: приобретение навыков по работе с файловой системой с помощью библиотеки `pathlib` языка программирования Python версии 3.x.

Задание 1

Для своего варианта лабораторной работы 2.17 добавьте возможность хранения файла данных в домашнем каталоге пользователя. Для выполнения операций с файлами необходимо использовать модуль `pathlib`.

Для решения этой задачи была создана функция сохранения в домашнем каталоге `save-home` – её выполнение отображено на рисунке 1.



```
(env) PS C:\Users\maxim\YandexDisk\Лабы 4 семестр\Технологии программирования> python individual1.py save-home data.json
(env) PS C:\Users\maxim\YandexDisk\Лабы 4 семестр\Технологии программирования> 
```

Рисунок 1 – Выполнение в консоли

Результат выполнения функции в корневом каталоге пользователя (рис. 2).

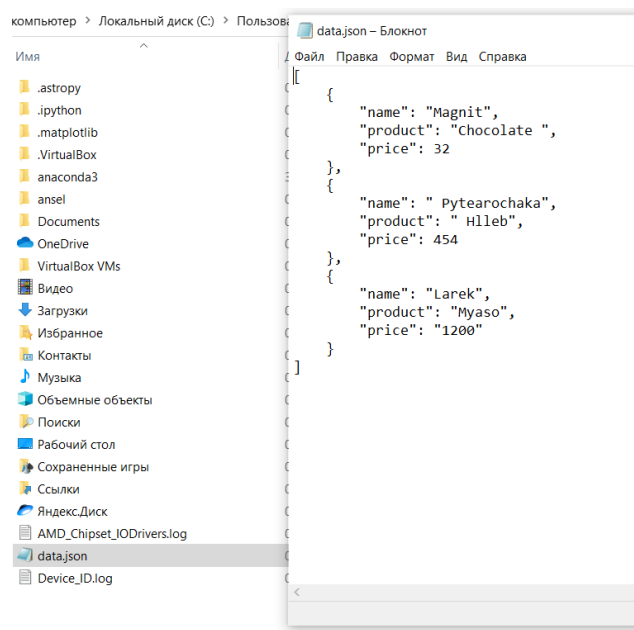


Рисунок 2 – Результат выполнения

Задание 2

Разработайте аналог утилиты tree в Linux. Используйте возможности модуля argparse для управления отображением дерева каталогов файловой системы. Добавьте дополнительные уникальные возможности в данный программный продукт. Выполнение команды отображено на рисунке 3.

```
++ ip3.exe
++ ython.exe
++ ythonw.exe
++ ndividual1.py
++ ndividual2.py
++ equirements.txt
++ $6 2.19.docx
++ a6 2.19.docx
++ a6 2.19.pdf
```

Рисунок 1 - Результат

Так же были добавлены функции отображения содержания текущей папки (рис. 2).

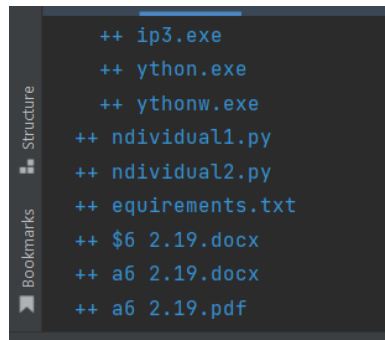


Рисунок 2 – Результат выполнения

Функция, подсчитывающая количество файлов определенного типа в текущей папке (рис. 3)

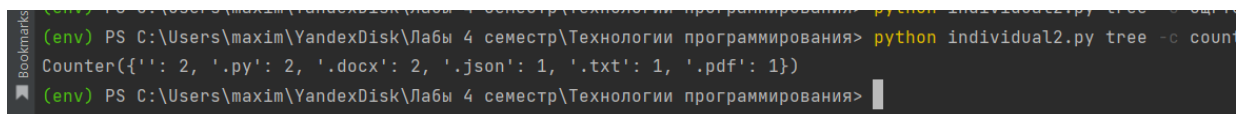


Рисунок 3 – Результат выполнения

Функция, подсчитывающая количество файлов выбранного типа (рис. 4)

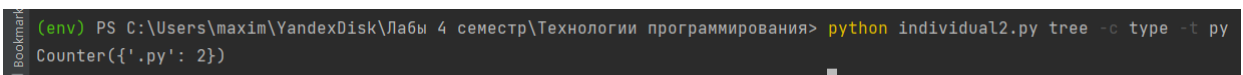
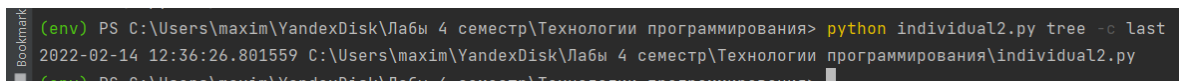


Рисунок 4 – Результат выполнения

Функция отображения последнего изменённого файла (рис. 5)



```
(env) PS C:\Users\maxim\YandexDisk\Лабы 4 семестр\Технологии программирования> python individual2.py tree -c last
2022-02-14 12:36:26.801559 C:\Users\maxim\YandexDisk\Лабы 4 семестр\Технологии программирования\individual2.py
```

Рисунок 5 – Результат выполнения

Вывод: в результате выполнения работы были приобретены навыки по работе с файловой системой с помощью библиотеки `pathlib` языка программирования Python версии 3.x.

Ответы на контрольные вопросы

1. До Python 3.4 работа с путями файловой системы осуществлялась либо с помощью методов строк: `>>> path.split('\\', maxsplit=1)[0]`, либо с помощью модуля `os.path` : `os.path.isfile(os.path.join(os.path.expanduser('~'), 'realpython.txt'))`
2. Что регламентирует PEP 428? - Модуль `pathlib` был введен в Python 3.4 (PEP 428) для решения этих проблем. Он объединяет необходимые функции в одном месте и делает его доступным через методы и свойства простого в использовании объекта `Path`
3. Все, что вам действительно нужно знать, это класс `pathlib.Path` . Есть несколько разных способов создания пути. Прежде всего, существуют `classmethods` наподобие `cwd()` (текущий рабочий каталог) и `.home()` (домашний каталог вашего пользователя).
4. Свойство `PurePath.parents` представляет собой неизменяемую последовательность, обеспечивающую доступ к логическим предкам пути.
5. Традиционно для чтения или записи файла в Python использовалась встроенная функция `open()` . Это все еще верно, поскольку функция `open()` может напрямую использовать объекты `Path` .
6. Для простого чтения и записи файлов в библиотеке `pathlib` есть несколько удобных методов:

- `.read_text()` : открыть путь в текстовом режиме и вернуть содержимое в виде строки.

- `.read_bytes()` : открыть путь в двоичном/байтовом режиме и вернуть содержимое в виде

- строки байтов.

- `.write_text()` : открыть путь и записать в него строковые данные.

- `.write_bytes()` : открыть путь в двоичном/байтовом режиме и записать в него данные.

7. Различные части пути удобно доступны как свойства. Основные примеры включают в себя:

- `.name` : имя файла без какого-либо каталога

- `.parent` : каталог, содержащий файл, или родительский каталог, если путь является

- каталогом

- `.stem` : имя файла без суффикса

- `.suffix` : расширение файла

- `.anchor` : часть пути перед каталогами

8. Через `pathlib` вы также получаете доступ к базовым операциям на уровне файловой системы, таким как перемещение, обновление и даже удаление файлов. По большей части эти методы не выдают предупреждение и не ждут подтверждения, прежде чем информация или файлы будут потеряны. Будьте осторожны при использовании этих методов. Чтобы переместить файл, используйте `.replace()` . Обратите внимание, что если место назначения уже существует, `.replace()` перезапишет его. К сожалению, `pathlib` явно не поддерживает безопасное перемещение файлов.

9. Есть несколько разных способов перечислить много файлов. Самым простым является метод `.iterdir()` , который перебирает все файлы в данном каталоге. Комбинируется `.iterdir()` с классом `collection.Counter` для подсчета количества файлов каждого типа в текущем каталоге.

10. Определяется функция `tree()` , которая будет печатать визуальное

дерево, представляющее иерархию файлов, с корнем в данном каталоге. Здесь мы также хотим перечислить подкаталоги, поэтому мы используем метод `.rglob()`.

11. Последний пример покажет, как создать уникальное нумерованное имя файла на основе шаблона. Сначала укажите шаблон для имени файла с местом для счетчика. Затем проверьте существование пути к файлу, созданного путем соединения каталога и имени файла (со значением счетчика).

12. Ранее мы отмечали, что когда мы создавали экземпляр `pathlib.Path`, возвращался либо объект `WindowsPath`, либо `PosixPath`. Тип объекта будет зависеть от операционной системы, которую вы используете. Эта функция позволяет довольно легко писать кроссплатформенный код. Можно явно запросить `WindowsPath` или `PosixPath`, но вы будете ограничивать свой код только этой системой без каких-либо преимуществ. Такой конкретный путь не может быть использован в другой системе: