

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
дисциплины «Анализ данных»
Вариант №2

Выполнила:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических
наук, доцент, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Работа с файловой системе в Python3 с использованием модуля pathlib

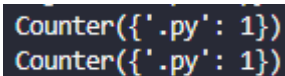
Цель: приобретение навыков по работе с файловой системой с помощью библиотеки pathlib языка программирования Python версии 3.x.

Ход работы

Подсчет файлов

```
import pathlib
import collections

print(collections.Counter(p.suffix for p in pathlib.Path.cwd().iterdir()))
print(collections.Counter(p.suffix for p in
pathlib.Path.cwd().glob('*.p*')))
```



```
Counter({' .py': 1})
Counter({' .py': 1})
```

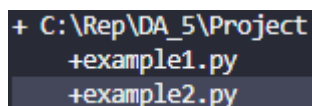
Рисунок 1. Результат работы программы

Показать дерево каталогов

```
import pathlib

def tree(directory):
    print(f'+ {directory}')
    for path in sorted(directory.rglob('*')):
        depth = len(path.relative_to(directory).parts)
        spacer = '    ' * depth
        print(f'{spacer}+{path.name}')

print(tree(pathlib.Path.cwd()))
```



```
+ C:\Rep\DA_5\Project
+example1.py
+example2.py
```

Рисунок 2. Результат работы программы

Найти последний измененный файл

```
from datetime import datetime
import pathlib
```

```
directory = pathlib.Path.cwd()
time, file_path = max((f.stat().st_mtime, f) for f in directory.iterdir())
print(datetime.fromtimestamp(time), file_path)
```

2024-04-02 13:03:23.842609 C:\Rep\DA_5\Project\example3.py

Рисунок 3. Результат работы программы

Создать уникальное имя файла

```
import pathlib

def unique_path(directory, name_pattern):
    counter = 0
    while True:
        counter += 1
        path = directory/name_pattern.format(counter)
        if not path.exists():
            return path

path = unique_path(pathlib.Path.cwd(), 'test{:03d}.txt')
print(path)
```

C:\Rep\DA_5\Project\test001.txt

Рисунок 4. Результат работы программы

Индивидуальное задание

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`.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pathlib import Path
import argparse
import json
import os.path
```

```

def add_student(staff, surname, group_number, grades):
    """
    Добавить данные о студенте.
    """
    staff.append(
        {
            "surname": surname,
            "group_number": group_number,
            "grades": grades
        }
    )

    return staff


def display_students(staff):
    """
    Отобразить список студентов.
    """
    # Проверить, что список студентов не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 14
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^14} |'.format(
                "№",
                "Ф.И.О.",
                "Группа",
                "Оценки"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>14} |'.format(
                    idx,
                    student.get('surname', ''),
                    student.get('group_number', ''),

```

```

        ', '.join(str(el) for el in student.get('grades')[0])
    )
    )
    print(line)

else:
    print("Список студентов пуст.")

def select_students(staff):
    # Сформировать список студентов, имеющих оценки 4 и 5.
    result = []
    for student in staff:
        if all(int(grade) >= 4 for grade in student['grades'][0]):
            result.append(student)

    # Возвратить список выбранных студентов.
    return result

def save_students(file_name, staff):
    """
    Сохранить всех учеников в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_students(file_name):
    """
    Загрузить всех учеников из файла JSON.
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)

def main(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The data file name"
    )

```

```
# Создать основной парсер командной строки.
parser = argparse.ArgumentParser("students")
parser.add_argument(
    "--version",
    action="version",
    version="% (prog)s 0.1.0"
)

subparsers = parser.add_subparsers(dest="command")

# Создать субпарсер для добавления студента.
add = subparsers.add_parser(
    "add",
    parents=[file_parser],
    help="Add a new student"
)
add.add_argument(
    "-sn",
    "--surname",
    action="store",
    type=str,
    required=True,
    help="The student's surname"
)
add.add_argument(
    "-gn",
    "--group_number",
    action="store",
    type=int,
    help="The student's group"
)
add.add_argument(
    "-g",
    "--grades",
    action="store",
    nargs='+',
    type=list,
    required=True,
    help="grades"
)

# Создать субпарсер для отображения всех студентов.
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Display all students"
```

```

)

# Создать субпарсер для выбора студентов.
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select the students"
)

# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)

home = Path.home()
file_path = home / args.filename

# Загрузить всех студентов из файла, если файл существует.
is_dirty = False
if os.path.exists(file_path):
    students = load_students(file_path)
else:
    students = []

# Добавить студента.
if args.command == "add":
    students = add_student(
        students,
        args.surname,
        args.group_number,
        args.grades
    )
    is_dirty = True

# Отобразить всех студентов.
elif args.command == "display":
    display_students(students)

# Выбрать требуемых студентов.
elif args.command == "select":
    selected = select_students(students)
    display_students(selected)

# Сохранить данные в файл, если список студентов был изменен.
if is_dirty:
    save_students(file_path, students)

if __name__ == "__main__":

```

```
main()

(base) PS C:\Rep\DA_5\Project> python indv1.py add data.json --surname="Третьяков К.Б" -gn=3 --grades="44454"
(base) PS C:\Rep\DA_5\Project> python indv1.py add data.json --surname="Журавлёва А.В" -gn=1 --grades="55545"
(base) PS C:\Rep\DA_5\Project> python indv1.py display data.json
+-----+-----+-----+-----+
| № | Ф.И.О. | Группа | Оценки |
+-----+-----+-----+-----+
| 1 | Михайлова С.А | 2 | 3, 3, 4, 3, 4 |
| 2 | Третьяков К.Б | 3 | 4, 4, 4, 5, 4 |
| 3 | Журавлёва А.В | 1 | 5, 5, 5, 4, 5 |
+-----+-----+-----+-----+
(base) PS C:\Rep\DA_5\Project> 
```

Рисунок 5. Результат работы программы

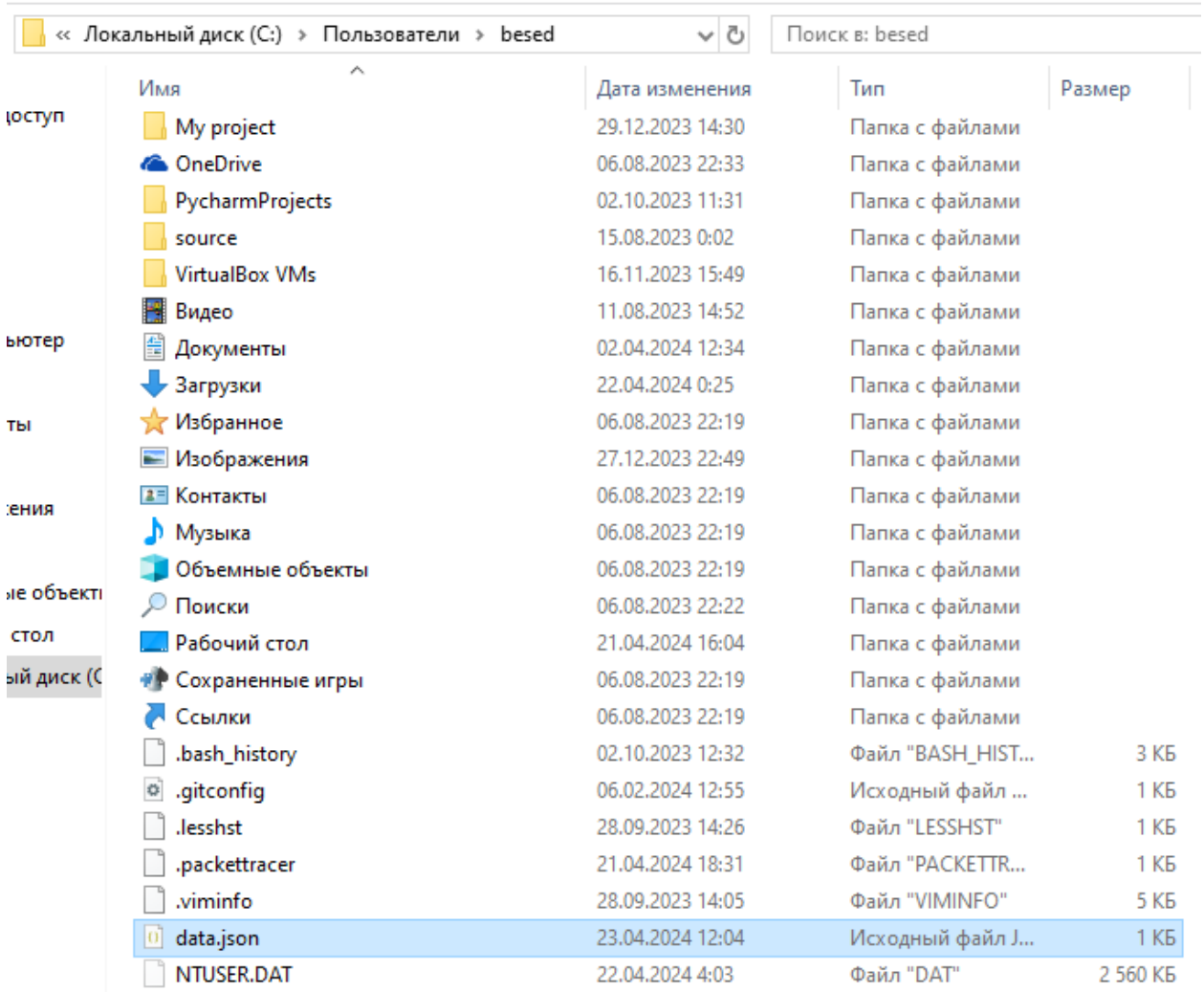


Рисунок 6. Файл данных в домашнем каталоге

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.

```
import os
```



```

import argparse

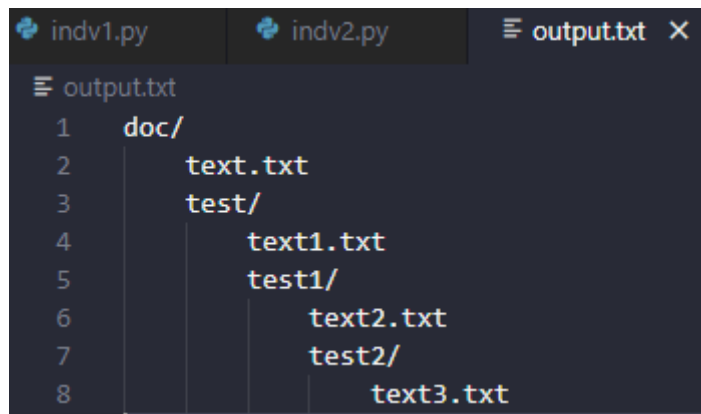
def list_files(startpath, output_file=None):
    for root, dirs, files in os.walk(startpath):
        level = root.replace(startpath, '').count(os.sep)
        indent = ' ' * 4 * (level)
        if output_file:
            with open(output_file, 'a') as f:
                f.write('{}{}/\n'.format(indent, os.path.basename(root)))
                subindent = ' ' * 4 * (level + 1)
                for file in files:
                    f.write('{}{}/\n'.format(subindent, file))
        else:
            print('{}{}/'.format(indent, os.path.basename(root)))
            subindent = ' ' * 4 * (level + 1)
            for file in files:
                print('{}{}/'.format(subindent, file))

def main():
    parser = argparse.ArgumentParser(
        description='Analog of the tree utility in Linux')
    parser.add_argument('directory', nargs='?', default='.',
                        help='Directory to display')
    parser.add_argument(
        '-o', '--output', help='Output file to save the tree structure')
    args = parser.parse_args()

    if args.output:
        list_files(args.directory, args.output)
    else:
        list_files(args.directory)

if __name__ == '__main__':
    main()

```

A screenshot of a code editor window with three tabs: 'indv1.py', 'indv2.py', and 'output.txt'. The 'output.txt' tab is active, showing a file tree structure. The tree starts with 'doc/' on line 1. Under 'doc/' is 'text.txt' on line 2. Under 'text.txt' is 'test/' on line 3. Under 'test/' is 'text1.txt' on line 4. Under 'text1.txt' is 'test1/' on line 5. Under 'test1/' is 'text2.txt' on line 6. Under 'text2.txt' is 'test2/' on line 7. Under 'test2/' is 'text3.txt' on line 8. The lines are numbered 1 through 8 on the left side of the editor.

```
1 doc/
2   text.txt
3   test/
4     text1.txt
5     test1/
6       text2.txt
7       test2/
8         text3.txt
```

Рисунок 7. Результат работы программы, сохранённый в отдельном файле

Контрольные вопросы:

1. До Python 3.4 для работы с файловой системой использовались модули `os` и `os.path`.
2. PEP 428 регламентирует добавление модуля `pathlib` в стандартную библиотеку Python.
3. Для создания путей средствами модуля `pathlib` используется метод `Path()`.
4. Для получения пути дочернего элемента файловой системы с помощью модуля `pathlib` можно использовать методы `joinpath()` или `/`.
5. Для получения пути к родительским элементам файловой системы с помощью модуля `pathlib` используется метод `parent`.
6. Операции с файлами с помощью модуля `pathlib` выполняются через методы этого модуля, такие как `open()`, `rename()`, `unlink()` и другие.
7. Для выделения компонентов пути файловой системы с помощью модуля `pathlib` используются различные свойства объекта `Path`, например, `name`, `stem`, `suffix`.
8. Для перемещения и удаления файлов с помощью модуля `pathlib` используются методы `rename()` и `unlink()` соответственно.
9. Для подсчета файлов в файловой системе можно использовать методы модуля `pathlib` в сочетании с циклами.
10. Для отображения дерева каталогов файловой системы можно использовать рекурсивные функции и методы модуля `pathlib`.

11. Для создания уникального имени файла можно использовать модуль `uuid` для генерации уникальных идентификаторов.

12. Основное отличие в использовании модуля `pathlib` для различных операционных систем заключается в том, что он автоматически обрабатывает различия в путях к файлам между операционными системами, что делает код более переносимым.

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