

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
№220 дисциплины «Основы программной инженерии»

Выполнил:

Джараян Арег Александрович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

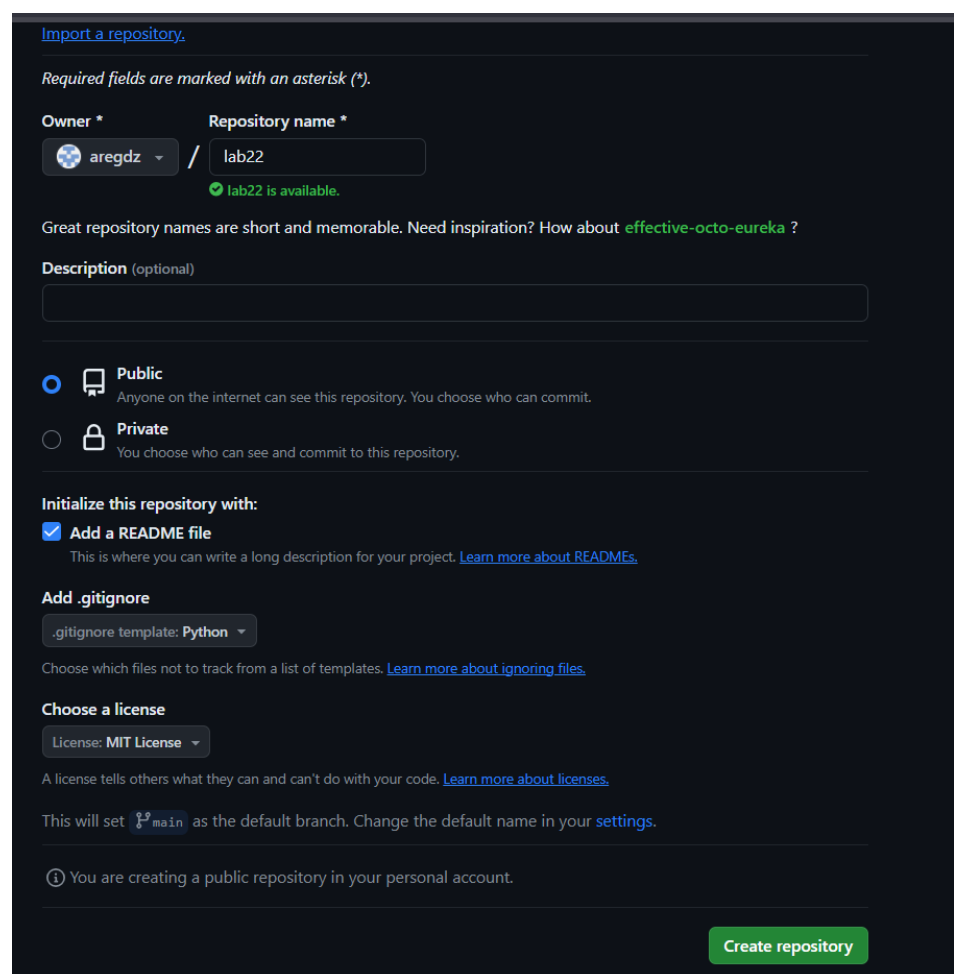
Ставрополь, 2024 г.

Тема: Лабораторная работа 2.19. Работа с файловой системе в Python3 с использованием модуля pathlib

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

Ход работы.

1. Создание нового репозитория с лицензией MIT.



The screenshot shows the GitHub 'Create repository' form. At the top, there is a link 'Import a repository.' and a note 'Required fields are marked with an asterisk (*)'. The 'Owner' field is set to 'aregdz' and the 'Repository name' field is 'lab22', with a green checkmark indicating 'lab22 is available.'. Below this is a suggestion for repository names: 'Great repository names are short and memorable. Need inspiration? How about effective-octo-eureka?'. The 'Description' field is optional and empty. The 'Visibility' section has 'Public' selected, with the description 'Anyone on the internet can see this repository. You choose who can commit.' and 'Private' as an alternative. The 'Initialize this repository with:' section has 'Add a README file' checked, with a note 'This is where you can write a long description for your project. Learn more about READMEs.' The 'Add .gitignore' section has a dropdown set to '.gitignore template: Python'. The 'Choose a license' section has a dropdown set to 'License: MIT License', with a note 'A license tells others what they can and can't do with your code. Learn more about licenses.' At the bottom, there is a note 'This will set main as the default branch. Change the default name in your settings.' and an information icon with the text 'You are creating a public repository in your personal account.' A green 'Create repository' button is at the bottom right.

Рисунок 1 – создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```

$ cd "D:\Рабочий стол\4 семестр\опи\lab22"

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/lab22
$ git clone https://github.com/aregdz/lab22.git
Cloning into 'lab22'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/lab22
$

```

Рисунок 2 – клонирование репозитория

3. Дополнил файл .gitignore необходимыми инструкциями.

```

1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/

```

Рисунок 4 – Файл .gitignore

```
aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/lab22/lab22 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/lab22/lab22 (develop)
$ |
```

Рисунок 4 – организация ветки

```
(venv) PS D:\Рабочий стол\4 семестр\опи\lab22\lab22> pip install -r requirements.txt
Collecting black==24.2.0 (from -r requirements.txt (line 1))
  Obtaining dependency information for black==24.2.0 from https://files.pythonhosted.org/packages/3e/58/etadata
  Using cached black-24.2.0-cp312-cp312-win_amd64.whl.metadata (74 kB)
Collecting cfgv==3.4.0 (from -r requirements.txt (line 2))
  Obtaining dependency information for cfgv==3.4.0 from https://files.pythonhosted.org/packages/c5/55/5
  Using cached cfgv-3.4.0-py2.py3-none-any.whl.metadata (8.5 kB)
Collecting click==8.1.7 (from -r requirements.txt (line 3))
  Obtaining dependency information for click==8.1.7 from https://files.pythonhosted.org/packages/00/2e/
```

Рисунок 5 – создание виртуального окружения

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

import argparse
import json
from pathlib import Path

def add_flight(flights, destination, departure_date, aircraft_type):

    flights.append(
        {
            "destination": destination,
            "departure_date": departure_date,
            "aircraft_type": aircraft_type,
        }
    )

    return flights

def display_flights(flights):

    if flights:
        line = "+-{}-+-{}-+-{}-+-{}-+ ".format(
            "-" * 4, "-" * 30, "-" * 20, "-" * 8
        )
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^8} | ".format(
                "No", "Destination", "Departure Date", "Aircraft Type"
            )
        )
        print(line)
```

```

        for idx, flight in enumerate(flights, 1):
            print(
                "| {:>4} | {:<30} | {:<20} | {:>8} |".format(
                    idx,
                    flight.get("destination", ""),
                    flight.get("departure_date", ""),
                    flight.get("aircraft_type", ""),
                )
            )
            print(line)
        else:
            print("List of flights is empty.")

def select_flights(flights, date):
    result = []
    for flight in flights:
        if flight.get("departure_date") == date:
            result.append(flight)
    return result

def save_flights(file_path, flights):
    with open(file_path, "w", encoding="utf-8") as fout:
        json.dump(flights, fout, ensure_ascii=False, indent=4)

def load_flights(file_path):
    with open(file_path, "r", encoding="utf-8") as fin:
        return json.load(fin)

def main(command_line=None):
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename", action="store", help="The data file name"
    )

    parser = argparse.ArgumentParser("flights")
    parser.add_argument(
        "--version", action="version", version="% (prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")

    add = subparsers.add_parser(
        "add", parents=[file_parser], help="Add a new flight"
    )
    add.add_argument(
        "-d",
        "--destination",
        action="store",
        required=True,
        help="Destination of the flight",
    )
    add.add_argument(
        "-dd",
        "--departure_date",
        action="store",
        required=True,
        help="Departure date of the flight",
    )
    add.add_argument(
        "-at",
        "--aircraft_type",
        action="store",
        required=True,
        help="Aircraft type of the flight",
    )

```

```

_ = subparsers.add_parser(
    "display", parents=[file_parser], help="Display all flights"
)

select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select flights by departure date",
)
select.add_argument(
    "-D",
    "--date",
    action="store",
    required=True,
    help="Departure date to select flights",
)

args = parser.parse_args(command_line)

home_dir = Path.home() # Получаем домашний каталог пользователя
file_path = home_dir / args.filename # Путь к файлу данных в домашнем каталоге

is_dirty = False
if file_path.exists():
    flights = load_flights(file_path)
else:
    flights = []

if args.command == "add":
    flights = add_flight(
        flights, args.destination, args.departure_date, args.aircraft_type
    )
    is_dirty = True

elif args.command == "display":
    display_flights(flights)

elif args.command == "select":
    selected = select_flights(flights, args.date)
    display_flights(selected)

if is_dirty:
    save_flights(file_path, flights)

if __name__ == "__main__":
    main()

```

Задание 1

```

import argparse
from pathlib import Path

COLORS = {
    'RESET': '\033[0m',
    'BOLD': '\033[1m',
    'DIR': '\033[94m', # Синий
    'EXEC': '\033[92m', # Зеленый
    'FILE': '\033[0m', # Сброс цвета
    'SIZE': '\033[93m' # Желтый
}

def format_size(size):
    """Форматирование размера файла/каталога для вывода."""
    if size < 1024:
        return f"{size} B"
    elif size < 1024 * 1024:
        return f"{size / 1024:.2f} KB"

```

```

elif size < 1024 * 1024 * 1024:
    return f"{size / 1024 / 1024:.2f} MB"
else:
    return f"{size / 1024 / 1024 / 1024:.2f} GB"

def tree(directory, max_depth=None, ignore=None, depth=0):
    """Отображение дерева каталогов."""
    directory_path = Path(directory) # Преобразуем путь к каталогу в объект Path

    for item in sorted(directory_path.iterdir()): # Итерируемся по содержимому
        каталога
        if ignore and item.name in ignore:
            continue

        prefix = '| ' * (depth - 1) + '|-- ' if depth > 0 else ''
        print(prefix, end='')

        if item.is_dir(): # Проверяем, является ли элемент каталогом
            print(f"{COLORS['DIR']}{COLORS['BOLD']}{item.name}{COLORS['RESET']}")
            if max_depth is None or depth < max_depth:
                tree(item, max_depth, ignore, depth + 1) # Рекурсивно вызываем функцию
        для каталога
        else:
            size = item.stat().st_size
            print(f"{COLORS['FILE']}{item.name}{COLORS['RESET']}
            ({COLORS['SIZE']}{format_size(size)}{COLORS['RESET']})")

def main():
    parser = argparse.ArgumentParser(description='Отображение дерева каталогов с
    использованием библиотеки pathlib')
    parser.add_argument('directory', nargs='?', default='.', help='Каталог для
    отображения дерева (по умолчанию текущий каталог)')
    parser.add_argument('-d', '--max-depth', type=int, help='Максимальная глубина
    отображения дерева')
    parser.add_argument('-i', '--ignore', nargs='+', help='Игнорируемые каталоги или
    файлы')
    args = parser.parse_args()

    directory = args.directory
    max_depth = args.max_depth
    ignore = args.ignore

    tree(directory, max_depth, ignore)

if __name__ == '__main__':
    main()

```

Задание 2

Контрольные вопросы:

1. Средства для работы с файловой системой до Python 3.4: До появления модуля `pathlib` в Python 3.4 основными средствами для работы с файловой системой были модули `os`, `os.path`, и `shutil`. Модуль `os` предоставлял функции для взаимодействия с операционной системой, `os.path` использовался для манипуляций с путями файлов, а `shutil` предлагал высокоуровневые операции с файлами, такие как копирование и удаление.

2. PEP 428 — "The pathlib module — object-oriented filesystem paths": PEP 428 описывает добавление модуля `pathlib` в стандартную библиотеку Python, который предлагает классы представления файловых системных путей с внутренней поддержкой многих удобных методов и функций. Это объектно-ориентированный подход к работе с путями.

3. Пути создаются путем инстанцирования объектов класса `Path`.

4. Для получения дочернего элемента (например, файла в каталоге) используйте оператор `/`.

5. Получение пути к родительским элементам с помощью `pathlib`: Для получения родительского каталога используйте свойство `.parent` или `.parents`.

6. Операции осуществляются через методы объекта `Path`, такие как `Path.read_text()` для чтения содержимого файла или `Path.write_text()` для записи текста в файл.

7. С помощью свойства `.parts` можно получить кортеж всех компонентов пути.

8. Метод `Path.rename(target)` используется для перемещения (или переименования) файла, а метод `Path.unlink()` для его удаления.

9. С помощью метода `Path.glob(pattern)` может быть выполнен подсчет файлов, соответствующих шаблону.