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
№220дисциплины «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 4.6 Класс данных в Python.

Цель работы: приобретение навыков по работе с классами данных при написании программ с помощью языка программирования Python версии 3.x.

1. Создание нового репозитория с лицензией MIT.

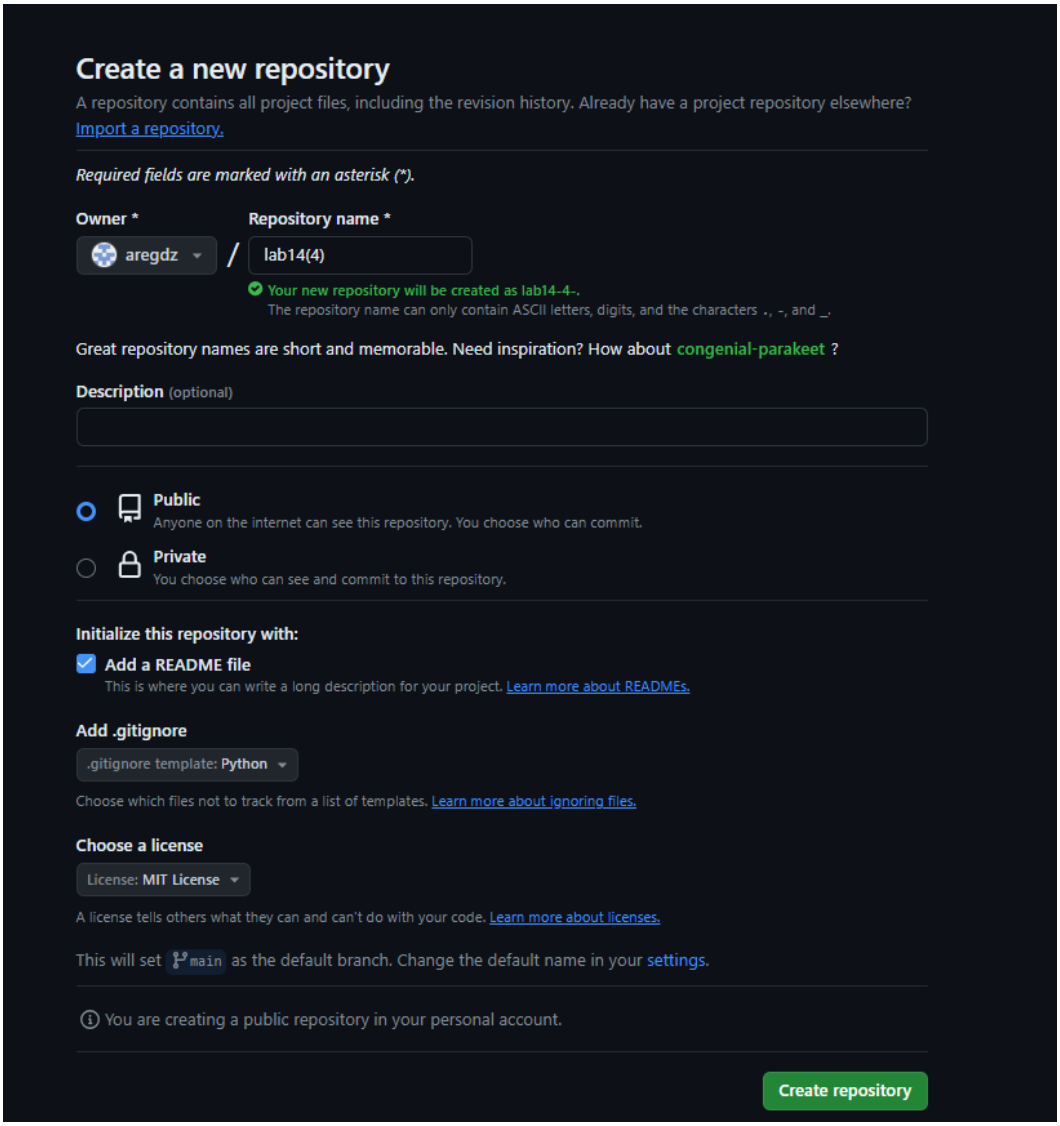

Рисунок 1 – создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```
aregd@DESKTOP-5KV9QA9 MINGW64 ~
$ cd "D:\Рабочий стол\4 семестр\опи\14"

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/14
$ git clone https://github.com/aregdz/lab14-4-.git
Cloning into 'lab14-4-'...
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/14
$
```

Рисунок 2 – клонирование репозитория

3. Дополнил файл .gitignore необходимыми инструкциями.

```
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
```

Рисунок 4 – Файл .gitignore

```
aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/14
$ cd lab14-4-

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/14/lab14-4- (main)
$ git checkout -b develop
Switched to a new branch 'develop'

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/14/lab14-4- (develop
)
$
```

Рисунок 4 – организация ветки

```
(venv) PS D:\Рабочий стол\4 сем
Package         Version
--------------- -------
black           24.4.0
cfgv            3.4.0
click           8.1.7
colorama        0.4.6
distlib         0.3.8
pyflakes        3.2.0
PyYAML          6.0.1
setuptools      69.5.1
virtualenv      20.25.2
```

Рисунок 5 – создание виртуального окружения

4.Пример 1.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from dataclasses import dataclass, field
from datetime import date
import sys
from typing import List
import xml.etree.ElementTree as ET
@dataclass(frozen=True)
class Worker:
    name: str
    post: str
    year: int
@dataclass
class Staff:
    workers: List[Worker] = field(default_factory=lambda: [])
```

```
def add(self, name, post, year):
    self.workers.append(
        Worker(
            name=name,
            post=post,
            year=year
        )
    )
    self.workers.sort(key=lambda worker: worker.name)
def __str__(self):
    # Заголовок таблицы.
    table = []
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20,
        '-' * 8
    )
    table.append(line)
    table.append(
        '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
            "№",
            "Ф.И.О.",
            "Должность",
            "Год"
        )
    )
    table.append(line)
    # Вывести данные о всех сотрудниках.
    for idx, worker in enumerate(self.workers, 1):
        table.append(
            '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                idx,
                worker.name,
                worker.post,
                worker.year
            )
        )
    table.append(line)
    return '\n'.join(table)
def select(self, period):
    # Получить текущую дату.
    today = date.today()
    result = []
    for worker in self.workers:
        if today.year - worker.year >= period:
            result.append(worker)
    return result
def load(self, filename):
    with open(filename, 'r', encoding='utf8') as fin:
        xml = fin.read()
        parser = ET.XMLParser(encoding="utf8")
        tree = ET.fromstring(xml, parser=parser)
        self.workers = []
        for worker_element in tree:
            name, post, year = None, None, None
            for element in worker_element:
                if element.tag == 'name':
                    name = element.text
                elif element.tag == 'post':
                    post = element.text
                elif element.tag == 'year':
                    year = int(element.text)
            if name is not None and post is not None \
                    and year is not None:
                self.workers.append(
                    Worker(
                        name=name,
                        post=post,
                        year=year
                    )
```

```
                )
def save(self, filename):
    root = ET.Element('workers')
    for worker in self.workers:
        worker_element = ET.Element('worker')
        name_element = ET.SubElement(worker_element, 'name')
        name_element.text = worker.name
        post_element = ET.SubElement(worker_element, 'post')
        post_element.text = worker.post
        year_element = ET.SubElement(worker_element, 'year')
        year_element.text = str(worker.year)
        root.append(worker_element)
    tree = ET.ElementTree(root)
    with open(filename, 'wb') as fout:
        tree.write(fout, encoding='utf8', xml_declaration=True)
if __name__ == '__main__':
    # Список работников.
    staff = Staff()
    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()
        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break
        elif command == 'add':
            # Запросить данные о работнике.
            name = input("Фамилия и инициалы? ")
            post = input("Должность? ")
            year = int(input("Год поступления? "))
            # Добавить работника.
            staff.add(name, post, year)
        elif command == 'list':
            # Вывести список.
            print(staff)
        elif command.startswith('select '):
            # Разбить команду на части для выделения номера года.
            parts = command.split(maxsplit=1)
            # Запросить работников.
            selected = staff.select(parts[1])
            # Вывести результаты запроса.
            if selected:
                for idx, worker in enumerate(selected, 1):
                    print(
                        '{:>4}: {}'.format(idx, worker.name)
                    )
            else:
                print("Работники с заданным стажем не найдены.")
        elif command.startswith('load '):
            # Разбить команду на части для имени файла.
            parts = command.split(maxsplit=1)
            # Загрузить данные из файла.
            staff.load(parts[1])
        elif command.startswith('save '):
            # Разбить команду на части для имени файла.
            parts = command.split(maxsplit=1)
            # Сохранить данные в файл.
            staff.save(parts[1])
        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить работника;")
            print("list - вывести список работников;")
            print("select <стаж> - запросить работников со стажем;")
            print("load <имя_файла> - загрузить данные из файла;")
            print("save <имя_файла> - сохранить данные в файл;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)
```

Рисунок 6 – пример 1

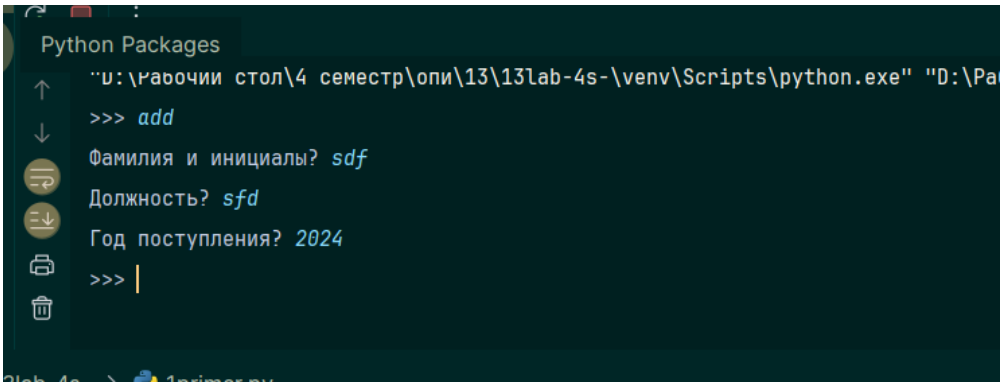

Рисунок 7 – выполнение примера 1

5.Задание 1. Выполнить индивидуальное задание лабораторной работы 4.5, использовав классы данных, а также загрузку и сохранение данных в формат XML.

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


import argparse
from dataclasses import dataclass, field
from typing import List
from pathlib import Path
import json
import xml.etree.ElementTree as ET

@dataclass
class Flight:
    destination: str
    departure_date: int
    aircraft_type: str

@dataclass
class FlightManager:
    file_path: Path
    flights: List[Flight] = field(default_factory=list)

    def add_flight(self, destination: str, departure_date: int, aircraft_type: str) ->
None:
        self.flights.append(Flight(destination, departure_date, aircraft_type))

    def display_flights(self) -> None:
        if self.flights:
            header_line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20,
"-" * 8)
            print(header_line)
            print("| {:^4} | {:^30} | {:^20} | {:^8} |".format("No", "Destination",
"Departure Date", "Aircraft Type"))
            print(header_line)
            for idx, flight in enumerate(self.flights, 1):
                print("| {:>4} | {:<30} | {:<20} | {:>8} |".format(idx,
flight.destination, flight.departure_date, flight.aircraft_type))
                print(header_line)
        else:
            print("No flights found.")
```

```
    def select_flights(self, date: int) -> List[Flight]:
        return [flight for flight in self.flights if flight.departure_date == date]

    def save_flights_json(self) -> None:
        with open(self.file_path, 'w', encoding='utf-8') as fout:
            json_dump = [flight.__dict__ for flight in self.flights]
            json.dump(json_dump, fout, ensure_ascii=False, indent=4)

    def load_flights_json(self) -> None:
        with open(self.file_path, 'r', encoding='utf-8') as fin:
            data = json.load(fin)
            self.flights = [Flight(**flight_data) for flight_data in data]

    def save_flights_xml(self) -> None:
        root = ET.Element('flights')
        for flight in self.flights:
            flight_element = ET.SubElement(root, 'flight')
            ET.SubElement(flight_element, 'destination').text = flight.destination
            ET.SubElement(flight_element, 'departure_date').text =
str(flight.departure_date)
            ET.SubElement(flight_element, 'aircraft_type').text = flight.aircraft_type
        tree = ET.ElementTree(root)
        with open(self.file_path, 'wb') as fout:
            tree.write(fout, encoding='utf-8', xml_declaration=True)

def main(command_line=None) -> 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()
    file_path = home_dir / args.filename

    flight_manager = FlightManager(file_path)

    if args.command == "add":
        flight_manager.add_flight(args.destination, int(args.departure_date),
args.aircraft_type)

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

    elif args.command == "select":
        selected_flights = flight_manager.select_flights(int(args.date))
        flight_manager.display_flights(selected_flights)

    if args.command in ("add", "display", "select"):
        if file_path.suffix == '.json':
            flight_manager.save_flights_json()
```

```
        elif file_path.suffix == '.xml':
            flight_manager.save_flights_xml()


if __name__ == "__main__":
    main()
```

Рисунок 8 – Задание 1

```
flights add: error: the following arguments are required: filename
(venv) PS D:\Рабочий стол\4 семестр\опи\14> python 1zadanie.py add 1zadanie.json -d "Paris" -dd 20240212 -at "Airbus A320"
(venv) PS D:\Рабочий стол\4 семестр\опи\14> python 1zadanie.py add 1zadanie.json -d "Paris" -dd 20240212 -at "Airbus A320"
(venv) PS D:\Рабочий стол\4 семестр\опи\14>
```

Рисунок 9 – пример выполнения примера 2

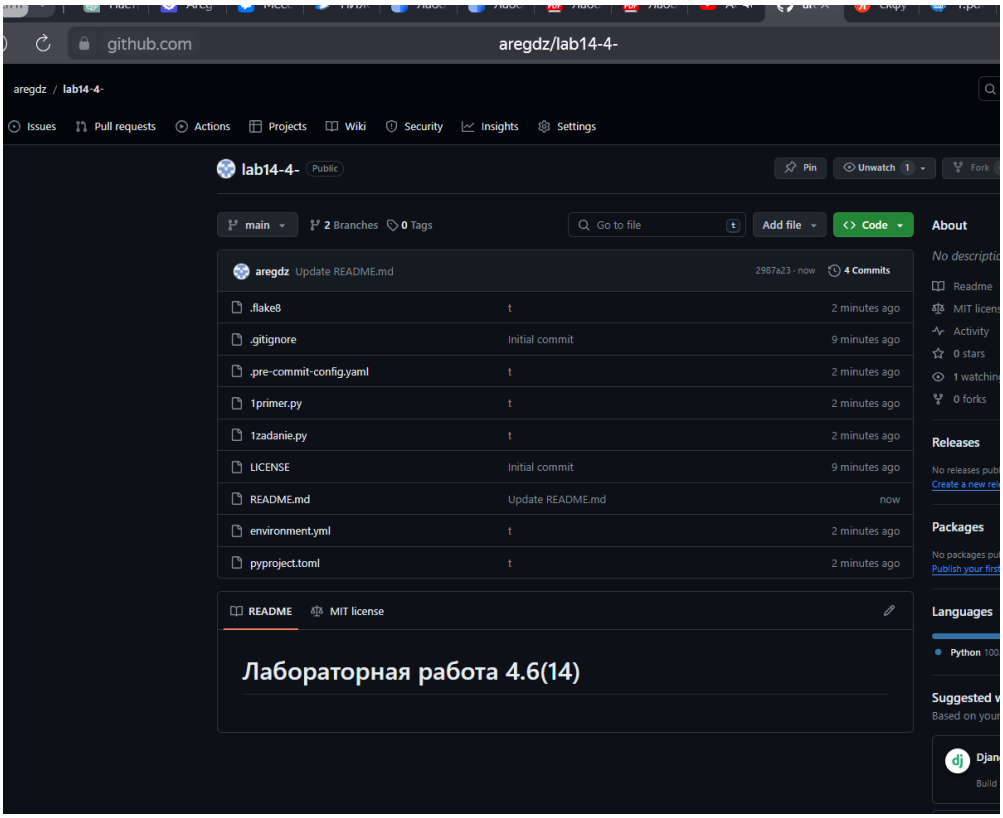

Рисунок 10 - фиксация изменений

Контрольные вопросы:

1. Чтобы создать класс данных в языке Python, используйте ключевое слово class, за которым следует имя класса, а затем двоеточие :. Внутри класса вы можете определить атрибуты и методы.

2. По умолчанию класс данных в Python наследует все методы от базового класса object. Эти методы включают в себя __init__() для инициализации объекта, __str__() для представления объекта в виде строки, __eq__() для сравнения объектов на равенство и т. д. Вы также можете определить свои собственные методы в классе для выполнения необходимых операций.

3. Чтобы создать неизменяемый класс данных (immutable data class) в Python, можно использовать декоратор @dataclass из модуля dataclasses с аргументом frozen=True.