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

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
дисциплины «Объектно-ориентированное программирование»

Выполнил:
Евдаков Евгений Владимирович
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Воронкин Р. А., доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: классы данных в Python.

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

Ход работы:

Задание 1. Создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер. Организовал свой репозиторий в соответствие с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

```
C:\Users\Gaming-PC\Postman>git clone https://github.com/EvgenyEvdakov/Laba_4.6.git
Cloning into 'Laba_4.6'...
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 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 1. Клонирование репозитория

Задание 2. Создал файлы poetry.lock и pyproject.toml, также установил необходимые пакеты isort, black, flake8, mypy, pre-commit, pytest.

```
(.venv) PS C:\Users\evdak\Laba_4.6> poetry init

This command will guide you through creating your pyproject.toml config.

Package name [laba_4.6]:
Version [0.1.0]:
Description []:
Author [Evgeni <kkrutkov02@gmail.com>, n to skip]:
License []:
Compatible Python versions [^3.10]:

Would you like to define your main dependencies interactively? (yes/no) [yes]
You can specify a package in the following forms:
  - A single name (requests): this will search for matches on PyPI
  - A name and a constraint (requests@^2.23.0)
  - A git url (git+https://github.com/python-poetry/poetry.git)
  - A git url with a revision (git+https://github.com/python-poetry/poetry.git#develop)
  - A file path (../my-package/my-package.whl)
  - A directory (../my-package/)
  - A url (https://example.com/packages/my-package-0.1.0.tar.gz)

Package to add or search for (leave blank to skip):

Would you like to define your development dependencies interactively? (yes/no) [yes]
Package to add or search for (leave blank to skip):

Generated file
```

Рисунок 2. Создание виртуального окружения

Задание 3. Создал проект PyCharm в папке репозитория. Приступил к работе с примером. Добавил новый файл primer1.py.

Условие примера: Для примера 2 лабораторной работы 9 добавьте возможность работы с классами данных, а также сохранения и чтения данных в формат XML.

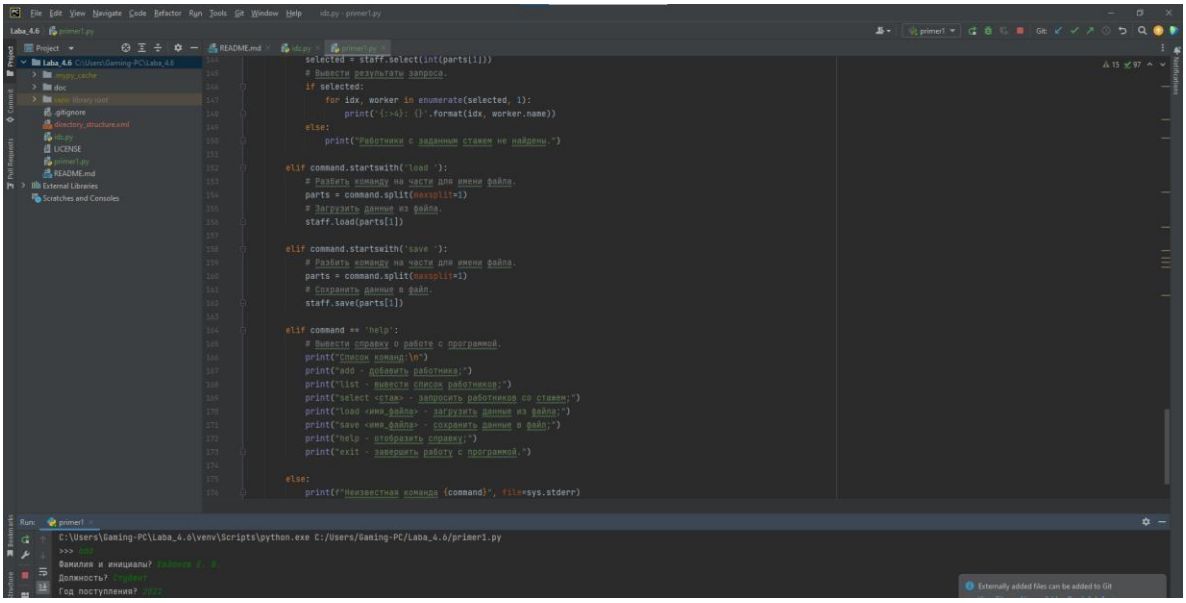

Рисунок 3. Выполнение первого примера

Задание 4.

Индивидуальное задание

Вариант 6

Создал новый файл под названием idz1.py.

Условие задания: выполнить индивидуальное задание лабораторной работы 4.5, использовав классы данных, а также загрузку и сохранение данных в формат XML.

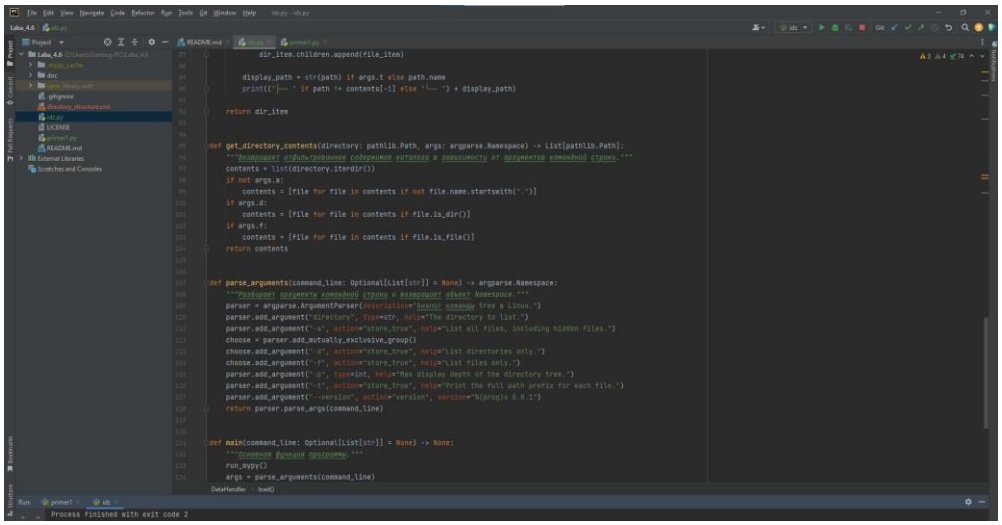

Рисунок 4. Выполнение первого индивидуального задания

Далее запустим код и проверим его на выполнение.

```
PS C:\Users\Gaming-PC\Laba_4.5> cd C:\Users\Gaming-PC\Laba_4.6
PS C:\Users\Gaming-PC\Laba_4.6> python idz.py -a C:\Users\Gaming-PC\Laba_4.6
Проверка типов с муру завершена успешно. Ошибок не найдено.
├── config
├── description
├── HEAD
├── applypatch-msg.sample
├── commit-msg.sample
├── fsmonitor-watchman.sample
├── post-update.sample
├── pre-applypatch.sample
├── pre-commit.sample
├── pre-merge-commit.sample
├── pre-push.sample
├── pre-rebase.sample
├── pre-receive.sample
├── prepare-commit-msg.sample
├── push-to-checkout.sample
├── sendemail-validate.sample
└── update.sample
├── hooks
├── index
└── exclude
├── info
├── HEAD
└── main
├── heads
└── HEAD
└── origin
└── remotes
└── refs
├── logs
└── 9de29bb2d1d6434b8b29ae775ad8c2e48c5391
├── e6
├── info
```

Рисунок 5. Выполнение программы

Задание 5.

После выполнения работы на ветке develop, слил ее с веткой main и отправил изменения на удаленный сервер.

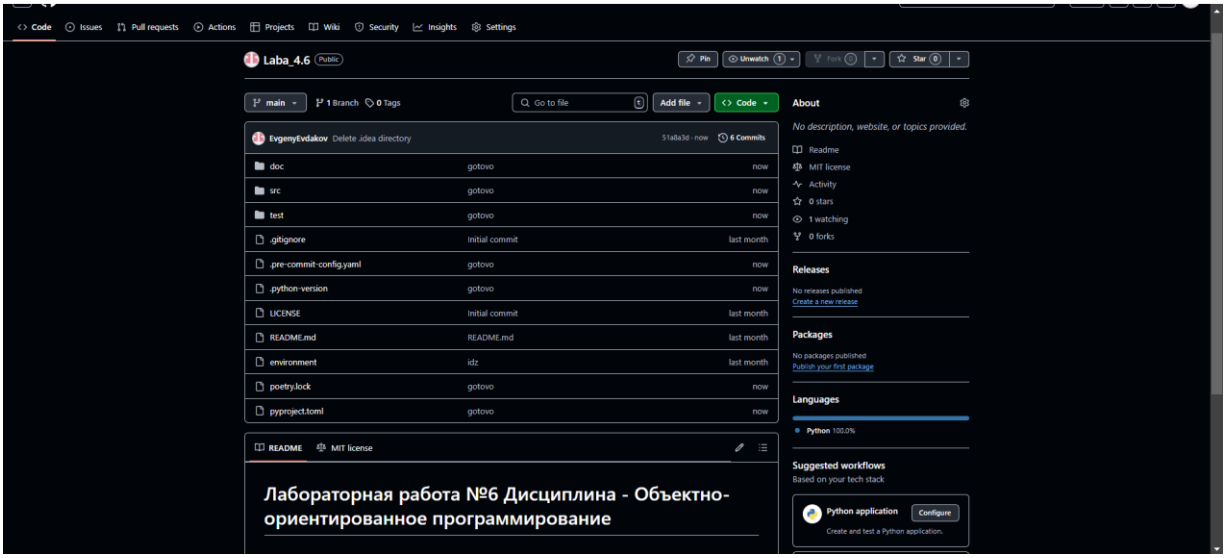

Рисунок 6. Готовый репозиторий

Ссылка: https://github.com/EvgenyEvdakov/Laba_4.6

Ответы на контрольные вопросы:

1. **Как создать класс данных в языке Python?**

Классы данных (или датаклассы) в Python можно создать с помощью декоратора @dataclass из модуля dataclasses. Декоратор @dataclass автоматически добавляет полезные методы в класс, такие как __init__, __repr__, __eq__, и другие, что облегчает работу с данными.

2. **Какие методы по умолчанию реализует класс данных?**

Декоратор @dataclass автоматически добавляет следующие методы, если они не были определены вручную:

- _init__: Конструктор, который инициализирует поля, используя аргументы, переданные при создании объекта.
- _repr__: Представление объекта в виде строки, удобно для его просмотра в консоли.
- _eq_: Метод сравнения, позволяющий сравнивать два объекта на равенство по значению их полей.
- __lt__, __le__, __gt__, __ge__: Методы для выполнения операций сравнения <, <=, >, >=, если в @dataclass указана опция order=True.
- _hash__: Если класс помечен как неизменяемый, также будет добавлен метод __hash__ для использования объектов в качестве ключей словаря или элементов множества.

3. **Как создать неизменяемый класс данных?**

Чтобы создать неизменяемый класс данных, нужно использовать параметр frozen=True в декораторе @dataclass. Этот параметр запрещает изменение полей после создания объекта, делая его неизменяемым (immutable).

Вывод: приобрел навыки по работе с классами данных при написании программ с помощью языка программирования Python версии 3.x.