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 №5
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: аннотация типов.

Цель: приобретение навыков по работе с аннотациями типов при написании программ с помощью языка программирования Python версии 3.x. Рассмотрен вопрос контроля типов переменных и функций с использованием комментариев и аннотаций. Приведено описание PEP-ов, регламентирующих работу с аннотациями, и представлены примеры работы с инструментом mypy для анализа Python кода.

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.

Рисунок 1. Клонирование репозитория

Задание 2. Создал виртуальное окружение conda и активировал его, также установил необходимые пакеты isort, black, flake8.

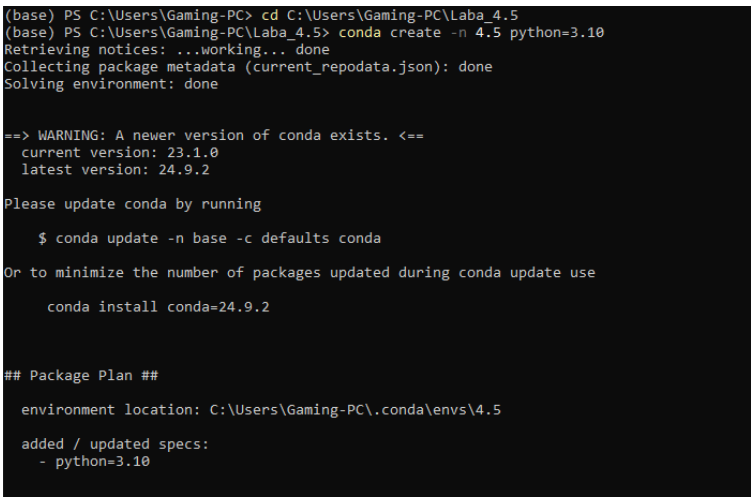

Рисунок 2. Создание виртуального окружения

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

Условие примера: Для примера 1 лабораторной работы 14 добавьте аннотации типов.

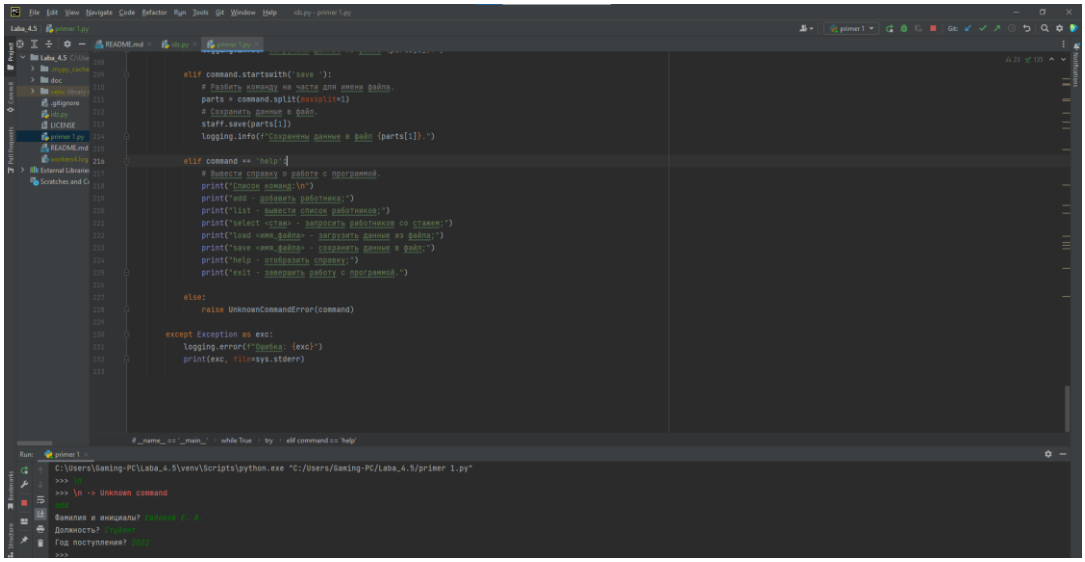

Рисунок 3. Выполнение первого примера

Задание 4.

Индивидуальное задание

Вариант 6

Создал новый файл под названием idz1.py.

Условие задания: выполнить индивидуальное задание 2 лабораторной работы 2.19, добавив аннтотации типов. Выполнить проверку программы с помощью утилиты mypy.

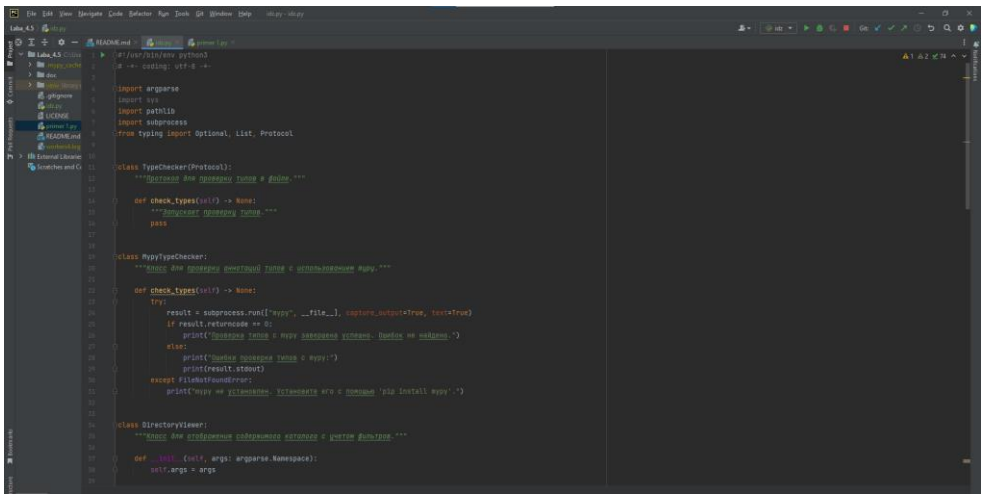

Рисунок 4. Выполнение первого индивидуального задания

Далее запустим код и проверим его на выполнение.

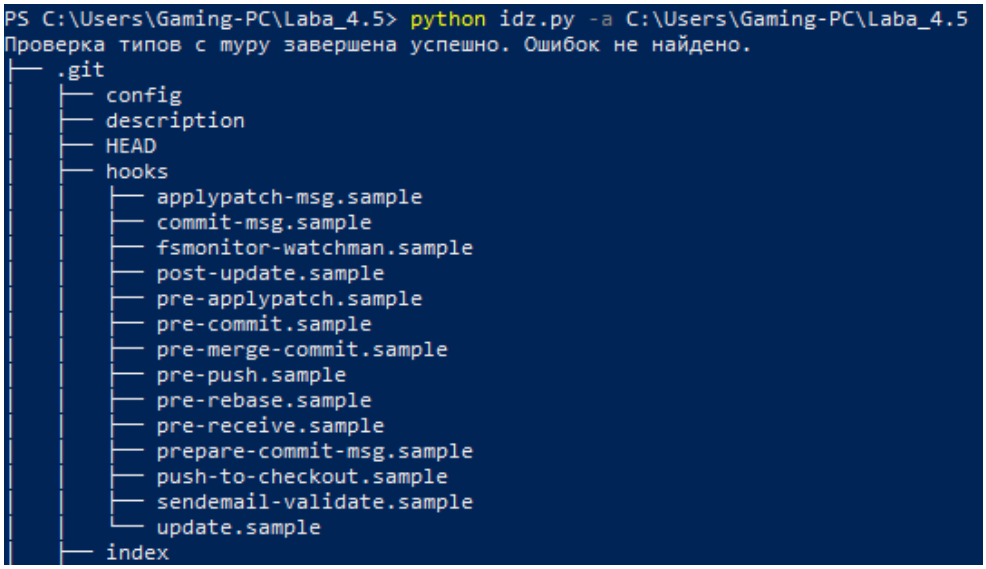

Рисунок 5. Выполнение программы

Задание 5.

После выполнения работы на ветке develop, слил ее с веткой main и отправил изменения на удаленный сервер. Создал файл envirement.yml и деактивировал виртуальное окружение.

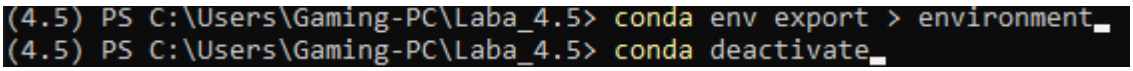

Рисунок 6. Деактивация ВО

Ссылка: https://github.com/EvgenyEvdakov/Laba_4.5

Ответы на контрольные вопросы:

1. **Для чего нужны аннотации типов в языке Python?**

Аннотации типов помогают сделать код более понятным, читаемым и проверяемым. Они позволяют разработчикам указывать, какого типа данные ожидаются для переменных, параметров и возвращаемых значений функций. Это упрощает чтение кода, облегчает отладку и работу над проектом в команде, а также позволяет инструментам статического анализа (например, mypy) выявлять возможные ошибки типов до запуска программы.

2. **Как осуществляется контроль типов языке Python?**

Python — это язык с динамической типизацией, где типы данных определяются и проверяются во время выполнения программы. Аннотации

типов в Python не влияют на исполнение кода и не заставляют язык следить за соответствием типов. Однако контроль типов можно выполнять с помощью сторонних инструментов, таких как mypy, Pyright, Pylint и других анализаторов, которые проверяют соответствие типов с учетом аннотаций и помогают выявить ошибки.

3. **Какие существуют предложения по усовершествованию Python для работы с аннотациями типов?**

Python постоянно развивается, и сообщество активно предлагает улучшения для работы с типами. Несколько важных предложений:

- PEP 484 — Введение стандартных аннотаций типов и базовых коллекций (например, List, Dict).
- PEP 585 — Позволяет использовать встроенные типы коллекций (например, list[int], dict[str, int]), начиная с Python 3.9.
- PEP 563 — Отложенная аннотация типов с использованием строк для улучшения производительности.
- PEP 563 и PEP 649 — Внесли изменения в правила вычисления аннотаций, позволяя указывать типы в виде строк для их отложенной интерпретации.
- PEP 544 — Протоколы, которые обеспечивают поддержку структурной типизации в Python.
- PEP 604 — Введение операторов объединения типов (int | str для обозначения Union).

4. **Как осуществляется аннотирование параметров и возвращаемых значений функций?**

Для аннотирования параметров и возвращаемых значений функции используют следующую синтаксис:

```
def функция(параметр: Тип) -> ТипВозвращаемогоЗначения:
    # Тело функции
    pass
```

5. **Как выполнить доступ к аннотациям функций?**

Доступ к аннотациям функции можно получить через специальное свойство __annotations__, которое содержит аннотации параметров и возвращаемого значения функции в виде словаря.

6. **Как осуществляется аннотирование переменных в языке Python?**

Для аннотирования переменных в Python используют двоеточие (:) после имени переменной и указывают тип данных. Python не заставляет придерживаться указанных типов, но аннотации делают код более понятным и позволяют инструментам проверки типов выполнять анализ типов.

7. **Для чего нужна отложенная аннотация в языке Python?**

Отложенная аннотация (введенная в PEP 563) позволяет использовать строки для указания типов в аннотациях, чтобы отложить их интерпретацию до момента, когда они будут реально использоваться. Это полезно для улучшения производительности, особенно когда типы зависят от других модулей, которые еще не загружены. Она также решает проблемы с циклическими зависимостями типов.

Вывод: приобрел навыки по работе с аннотациями типов при написании программ с помощью языка программирования Python версии 3.x. Рассмотрен вопрос контроля типов переменных и функций с использованием комментариев и аннотаций. Приведено описание PEP-ов, регламентирующих работу с аннотациями, и представлены примеры работы с инструментом mypy для анализа Python кода.