

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 №4.6**  
**дисциплины «Объектно-ориентированное программирование»**

Выполнил:  
Магомедов Имран Борисович  
3 курс, группа «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., доцент департамента  
цифровых, робототехнических систем и  
электроники

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2024 г.

## Тема: Классы данных в Python.

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

### Методика выполнения работы


1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*


**Owner \*** **Repository name \***


 m4g0med0v / OOP\_LR\_7

✔ OOP\_LR\_7 is available.

Great repository names are short and memorable. Need inspiration? How about [laughing-octo-enigma](#) ?

**Description** (optional)

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

3. Выполните клонирование созданного репозитория.

```
~/ncfu git clone https://github.com/m4g0med0v/00P_LR_7.git
Клонирование в «00P_LR_7»...
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)
Получение объектов: 100% (5/5), готово.
```

4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
~/ncfu/00P_LR_7 main git switch develop
Переключились на ветку «develop»
```

5. Проработайте примеры лабораторной работы.

```
example_1.py U x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 import sys
6 import xml.etree.ElementTree as ET
7 from dataclasses import dataclass, field
8 from datetime import date
9 from typing import List
10
11
12 @dataclass(frozen=True)
13 class Worker:
14     name: str
15     post: str
16     year: int
17
18
19 @dataclass
20 class Staff:
21     workers: List[Worker] = field(default_factory=lambda: [])
22
23     def add(self, name, post, year):
24         self.workers.append(Worker(name=name, post=post, year=year))
```

6. Выполните индивидуальное задание. Приведите в отчете скриншоты работы программ решения индивидуального задания.

## Задание 1.

Выполнить индивидуальное задание лабораторной работы 4.5, используя классы данных, а также загрузку и сохранение данных в формат XML.

```
1 import argparse
2 import logging
3 import os
4 import sqlite3
5 import xml.etree.ElementTree as ET
6 from dataclasses import asdict, dataclass
7 from pathlib import Path
8 from typing import List, Optional, Tuple
9
10
11 class ConnectError(Exception): ...
12
13
14 # Настройка логгирования
15 logging.basicConfig(
16     level=logging.INFO,
17     datefmt="%Y-%m-%d %H:%M:%S",
18     format=(
19         "[%asctime)s.%(msecs)03d] %(module)10s:%(lineno)-3d"
20         " %(levelname)-7s - %(message)s"
21     ),
22 )
23
24
25 @dataclass
26 class Train:
27     number: str
28     destination: str
29     time: str
30     station_name: str
```

```
~/ncfu/00P_LR_7 git develop ?4 python src/individual_task_1.py add -d Stavropol -n 003D -t "15:00" -s Dagestan
[2024-11-21 15:45:19.205] individual_task_1:59 INFO - Соединение с базой данных успешно установлено.
[2024-11-21 15:45:19.208] individual_task_1:109 INFO - Добавлен поезд №003D, пункт назначения: Stavropol, время отправления: 15:00.
[2024-11-21 15:45:19.208] individual_task_1:166 INFO - Поиск поезда №003D завершен.
Поезд №003D в Stavropol добавлен.

~/ncfu/00P_LR_7 git develop ?4 python src/individual_task_1.py save-xml data/trains.xml
[2024-11-21 15:45:22.474] individual_task_1:59 INFO - Соединение с базой данных успешно установлено.
[2024-11-21 15:45:22.475] individual_task_1:138 INFO - Список поездов успешно извлечен.
[2024-11-21 15:45:22.475] individual_task_1:74 INFO - Данные сохранены в файл data/trains.xml
Данные сохранены в файл data/trains.xml.

~/ncfu/00P_LR_7 git develop ?4 python src/individual_task_1.py load-xml data/trains.xml
[2024-11-21 15:45:25.474] individual_task_1:59 INFO - Соединение с базой данных успешно установлено.
[2024-11-21 15:45:25.475] individual_task_1:84 INFO - Данные загружены из файла data/trains.xml
[2024-11-21 15:45:25.475] individual_task_1:117 ERROR - Поезд с номером 003D уже существует.
Ошибка: поезд с номером 003D уже существует.
Данные из файла data/trains.xml загружены в базу данных.
```

## 7. Зафиксируйте сделанные изменения в репозитории.

```

~/ncfu/OOP_LR_7 git develop +6 ?1 git commit -m "add all changes"
check yaml.....Passed
fix end of files.....Passed
trim trailing whitespace.....Passed
ruff.....Passed
ruff-format.....Passed
mypy.....Passed
[develop 471e925] add all changes
6 files changed, 592 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 examples/example_1.py
create mode 100644 requirements.txt
create mode 100644 src/individual_task_1.py
create mode 100644 tests/run_tests.py
create mode 100644 tests/test_individual_task_1.py

```

8. Выполните слияние ветки для разработки с веткой main / master.

```

~/ncfu/OOP_LR_7 main ?1 git merge develop
Обновление 63dbc8b..471e925
Fast-forward
 .pre-commit-config.yaml | 24 ++++++ для {filename}")
 examples/example_1.py | 162 ++++++
 requirements.txt | 1 +
 src/individual_task_1.py | 268 ++++++
 tests/run_tests.py | 15 ++++++
 tests/test_individual_task_1.py | 122 ++++++
 6 files changed, 592 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 examples/example_1.py
create mode 100644 requirements.txt
create mode 100644 src/individual_task_1.py
create mode 100644 tests/run_tests.py
create mode 100644 tests/test_individual_task_1.py

```

9. Отправьте сделанные изменения на сервер GitHub.

```

~/ncfu/OOP_LR_7 main +1 ?1 git push
Перечисление объектов: 12, готово.
Подсчет объектов: 100% (12/12), готово.
При сжатии изменений используется до 12 потоков
Сжатие объектов: 100% (8/8), готово.
Запись объектов: 100% (11/11), 6.80 КиБ | 1.70 МиБ/с, готово.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/m4g0med0v/OOP_LR_7.git
63dbc8b..471e925 main -> main

```

## Контрольные вопросы

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

Для создания класса данных нужно импортировать декоратор `dataclass` из модуля `dataclasses` и использовать его:

```
1 from dataclasses import dataclass
2
3 @dataclass
4 class Person:
5     name: str
6     age: int
7     city: str = "Unknown" # Можно задавать значения по умолчанию
```

При создании объекта класса `Person` автоматически создается конструктор, принимающий аргументы для всех полей:

```
8
9 person = Person(name="Alice", age=30)
10 print(person) # Person(name='Alice', age=30, city='Unknown')
```

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

Декоратор `@dataclass` генерирует следующие методы, если их не определить вручную:

- `__init__`: Конструктор, инициализирующий поля класса.
- `__repr__`: Текстовое представление объекта (удобно для отладки).
- `__eq__`: Метод сравнения объектов (по значениям полей).
- `__hash__`: Хэш-функция (если класс помечен как неизменяемый).
- `__post_init__`: Специальный метод, который вызывается после `__init__`, если требуется дополнительная инициализация.

## 3. Как создать неизменяемый класс данных?

Для создания неизменяемого (immutable) класса данных нужно использовать параметр `frozen=True` в декораторе `@dataclass`. Это сделает объект класса неизменяемым: при попытке изменить значение поля будет вызвано исключение `FrozenInstanceError`.