

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 №2.6**  
**дисциплины «Основы программной инженерии»**

Выполнил:  
Юрьев Илья Евгеньевич  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Руководитель практики:  
Богданов С.С., ассистент кафедры  
инфокоммуникаций

---

(подпись)

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

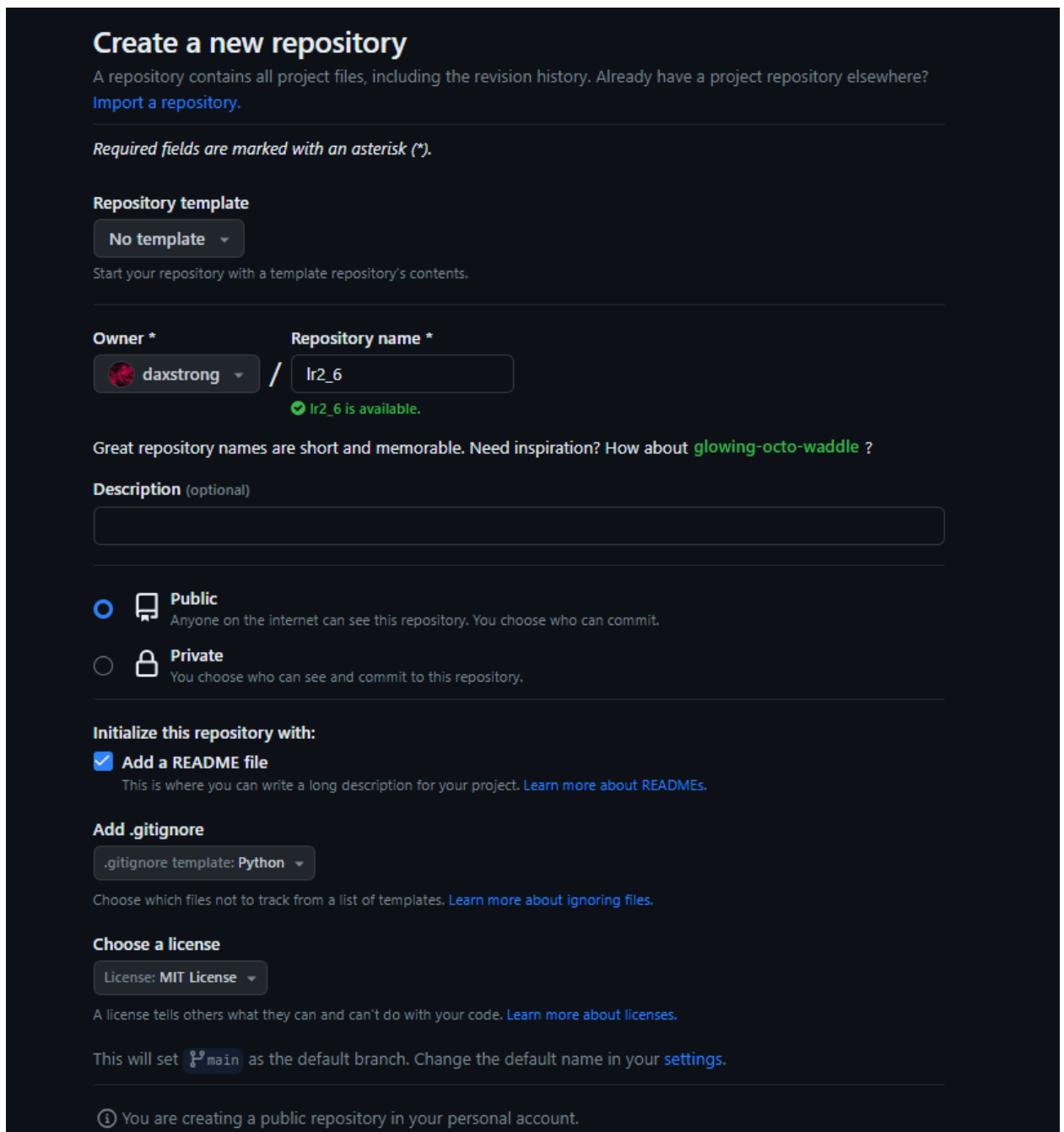
Ставрополь, 2023 г.

**Тема:** Работа с кортежами в языке Python.

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

**Ход выполнения работы:**

1. 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 (\*).*

**Repository template**

**No template** ▾

Start your repository with a template repository's contents.

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

 **daxstrong** ▾ / **lr2\_6**

✔ lr2\_6 is available.

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

**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**

**.gitignore template: Python** ▾

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

**Choose a license**

**License: MIT License** ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

📌 You are creating a public repository in your personal account.

Рисунок 1 – Создание репозитория с заданными настройками

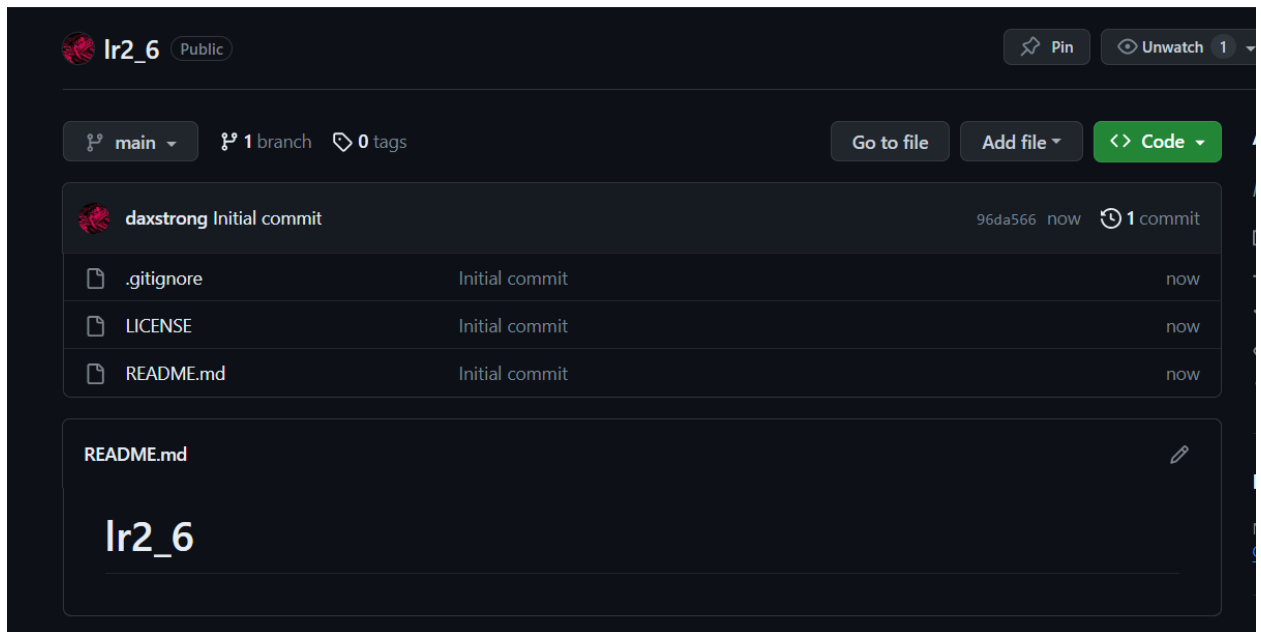


Рисунок 2 – Созданный репозиторий

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии
$ git clone https://github.com/daxstrong/lr2_6.git
Cloning into 'lr2_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
Receiving objects: 100% (5/5), done.
```

Рисунок 3 – Клонирование репозитория

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr2_6 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

Рисунок 4 – Создание ветки develop

2. Проработать примеры лабораторной работы, оформляя код согласно PEP-8:

```

example1.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5 from datetime import date
6
7 ▶ if __name__ == '__main__':
8     # Список работников.
9     workers = []
10    # Организовать бесконечный цикл запроса команд.
11    while True:
12        # Запросить команду из терминала.
13        command = input(">>> ").lower()
14        # Выполнить действие в соответствие с командой.
15        if command == 'exit':
16            break
17        elif command == 'add':
18            # Запросить данные о работнике.
19            name = input("Фамилия и инициалы? ")
20            post = input("Должность? ")
21            year = int(input("Год поступления? "))
22            # Создать словарь.
23            worker = {
24                'name': name,
25                'post': post,
26                'year': year,
27            }
28            # Добавить словарь в список.
29            workers.append(worker)
30            # Отсортировать список в случае необходимости.
31            if len(workers) > 1:
32                workers.sort(key=lambda item: item.get('name', ''))

```

```

33 elif command == 'list':
34     # Заголовок таблицы.
35     line = '+-{}-+-{}-+-{}-+-{}-+'.format(
36         '-' * 4,
37         '-' * 30,
38         '-' * 20,
39         '-' * 8
40     )
41     print(line)
42     print(
43         '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
44             "№",
45             "Ф.И.О.",
46             "Должность",
47             "Год"
48         )
49     )
50     print(line)
51     # Вывести данные о всех сотрудниках.
52     for idx, worker in enumerate(workers, 1):
53         print(
54             '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
55                 idx,
56                 worker.get('name', ''),
57                 worker.get('post', ''),
58                 worker.get('year', 0)
59             )
60         )
61     print(line)
62 elif command.startswith('select '):
63     # Получить текущую дату.
64     today = date.today()

```

```

65 # Разбить команду на части для выделения номера года.
66 parts = command.split(' ', maxsplit=1)
67 # Получить требуемый стаж.
68 period = int(parts[1])
69 # Инициализировать счетчик.
70 count = 0
71 # Проверить сведения работников из списка.
72 for worker in workers:
73     if today.year - worker.get('year', today.year) >= period:
74         count += 1
75         print(
76             '{:>4}: {}'.format(count, worker.get('name', ''))
77         )
78 # Если счетчик равен 0, то работники не найдены.
79 if count == 0:
80     print("Работники с заданным стажем не найдены.")
81 elif command == 'help':
82     # Вывести справку о работе с программой.
83     print("Список команд:\n")
84     print("add - добавить работника;")
85     print("list - вывести список работников;")
86     print("select <стаж> - запросить работников со стажем;")
87     print("help - отобразить справку;")
88     print("exit - завершить работу с программой.")
89 else:
90     print(f"Неизвестная команда {command}", file=sys.stderr)

```

Рисунок 5 — Пример 1

```
example1 x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/ilyay/
>>> add
Фамилия и инициалы? Семёнов С.С
Должность? Повар
Год поступления? 2006
>>> add
Фамилия и инициалы? Ковалёв А.С
Должность? Начальник
Год поступления? 2003
>>> list
+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+
|  1 | Ковалёв А.С              |      Начальник      |      2003      |
|  2 | Семёнов С.С              |      Повар          |      2006      |
+-----+-----+-----+
>>> select 2
1: Ковалёв А.С
2: Семёнов С.С
>>>
```

Рисунок 6 – Вывод программы (Пример 1)

3. Решим задачу: создайте словарь, связав его с переменной school, и наполните данными, которые бы отражали количество учащихся в разных классах (1а, 1б, 2б, 6а, 7в и т. п.). Внесите изменения в словарь согласно следующему: а) в одном из классов изменилось количество учащихся, б) в школе появился новый класс, с) в школе был расформирован (удален) другой класс. Вычислите общее количество учащихся в школе.

```
task1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == '__main__':
5      # Создание словаря с данными о количестве учащихся в разных классах
6      school = {
7          '1а': 25,
8          '1б': 28,
9          '2б': 30,
10         '6а': 22,
11         '7в': 26,
12     }
13
14     # Изменение количества учащихся в одном из классов
15     school['1б'] = 30
16
17     # Добавление нового класса
18     school['8г'] = 24
19
20     # Удаление класса
21     del school['2б']
22
23     # Вычисление общего количества учащихся в школе
24     total_students = sum(school.values())
25
26     print(school)
27     print(f"Общее количество учащихся в школе: {total_students}")
28
```

Рисунок 7 – Задание №1

```
task1 x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Общее количество учащихся в школе: 127
Process finished with exit code 0
```

Рисунок 8 – Вывод программы (Задание №1)

4. Решите задачу: создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод `items()`, с помощью полученного объекта `dict_items` создайте новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.



```
task2.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     original_dict = {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
6
7     # Получение объекта dict_items
8     dict_items = original_dict.items()
9
10    # Создание "обратного" словаря
11    inverted_dict = {value: key for key, value in dict_items}
12
13    print("Обратный словарь:")
14    print(inverted_dict)
15
```

Рисунок 9 – Задание №2

```
task2 x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Обратный словарь:
{'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}
Process finished with exit code 0
```

Рисунок 10 – Вывод программы (Задание №2)

## 5. Выполним индивидуальное задание:

12. Использовать словарь, содержащий следующие ключи: фамилия, имя; номер телефона; дата рождения (список из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть размещены по алфавиту; вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение.

### Листинг программы:

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

import sys
from datetime import datetime

if __name__ == '__main__':
    people = []
```

```

while True:
    command = input(">>> ").lower()

    match command:
        case 'exit':
            break
        case 'add':
            last_name = input("Фамилия: ")
            first_name = input("Имя: ")
            phone_number = input("Номер телефона: ")
            birthdate_str = input("Дата рождения (в формате ДД.ММ.ГГГГ): ")
            birthdate = datetime.strptime(birthdate_str, "%d.%m.%Y")

            person = {
                'фамилия': last_name,
                'имя': first_name,
                'номер телефона': phone_number,
                'дата рождения': birthdate,
            }

            people.append(person)
            people.sort(key=lambda x: x['фамилия'])

        case 'list':
            line = f'+-{"-" * 25}+-{"-" * 15}+-{"-" * 25}+'
            print(line)
            print(f"| {'Фамилия':^25} | {'Имя':^15} | {'Дата рождения':^25} |")

            for person in people:
                print(line)
                print(f"| {person['фамилия']:^25} | {person['имя']:^15} | {person['дата рождения'].strftime('%d.%m.%Y'):^25} |")
                print(line)

        case command if command.startswith('select '):
            month_to_search = int(command.split(' ')[1])
            found = False

            print(f"Люди с днем рождения в месяце {month_to_search}:")
            for person in people:
                if person['дата рождения'].month == month_to_search:
                    print(
                        f"Фамилия: {person['фамилия']}, Имя: {person['имя']}, Дата
рождения: {person['дата рождения'].strftime('%d.%m.%Y')}"
                    )
                    found = True

            if not found:
                print("Нет людей с днем рождения в указанном месяце.")

        case 'help':
            print("Список команд:\n")
            print("add - добавить информацию о человеке;")
            print("list - вывести список всех людей;")
            print("select <месяц> - вывести людей с днем рождения в указанном
месяце;")

            print("exit - завершить работу с программой.")

        case _:
            print(f"Неизвестная команда {command}", file=sys.stderr)

```

```
individual x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/ilyay/OneDrive/Рабочий стол
>>> add
Фамилия: Юрьев
Имя: Илья
Номер телефона: 878743283
Дата рождения (в формате ДД.ММ.ГГГГ): 02.03.2004
>>> add
Фамилия: Ковалев
Имя: Сергей
Номер телефона: 878372422
Дата рождения (в формате ДД.ММ.ГГГГ): 11.11.1999
>>> list
+-----+-----+-----+
|      Фамилия      |      Имя      |      Дата рождения      |
+-----+-----+-----+
|      Ковалев      |      Сергей    |      11.11.1999         |
+-----+-----+-----+
|      Юрьев        |      Илья      |      02.03.2004         |
+-----+-----+-----+
>>> select 11
Люди с днем рождения в месяце 11:
Фамилия: Ковалев, Имя: Сергей, Дата рождения: 11.11.1999
```

Рисунок 11 – Вывод программы (Индивидуальное задание)

6. Зафиксируем проделанные изменения, сольем ветки и отправим на удаленный репозиторий:

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_6 (develop)
$ git log
commit 392a290245732682f8f986f20289e53437aa52f4 (HEAD -> develop)
Author: dexstrong <ilya.yurev.04@inbox.ru>
Date: Tue Dec 5 00:52:31 2023 +0300

    final changes

commit 96da566edae9f0f10eb385fedd89f61110b5fba8 (origin/main, origin/HEAD, main)
Author: Ilya Yurev <112946692+daxstrong@users.noreply.github.com>
Date: Mon Dec 4 22:21:29 2023 +0300

    Initial commit
```

Рисунок 12 – Коммиты ветки develop во время выполнения лабораторной работы

```

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_6 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr2_6 (main)
$ git merge develop
Updating 96da566..392a290
Fast-forward
 .idea/.gitignore | 8 +++
 .idea/inspectionProfiles/profiles_settings.xml | 6 ++
 .idea/lr2_6.iml | 8 +++
 .idea/misc.xml | 4 ++
 .idea/modules.xml | 8 +++
 .idea/vcs.xml | 6 ++
 example1.py | 90 +++++++++++++++++++++++++++++++++++++
 individual.py | 66 +++++++++++++++++++++++++++++++++++++
 task1.py | 26 ++++++
 task2.py | 14 +++++
10 files changed, 236 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/lr2_6.iml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 example1.py
create mode 100644 individual.py
create mode 100644 task1.py
create mode 100644 task2.py

```

Рисунок 13 – Слияние веток main и develop

```

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr2_6 (main)
$ git push origin main
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (14/14), 4.60 KiB | 4.60 MiB/s, done.
Total 14 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/daxstrong/lr2_6.git
 96da566..392a290 main -> main

```

Рисунок 14 – Отправка изменений на удаленный репозиторий

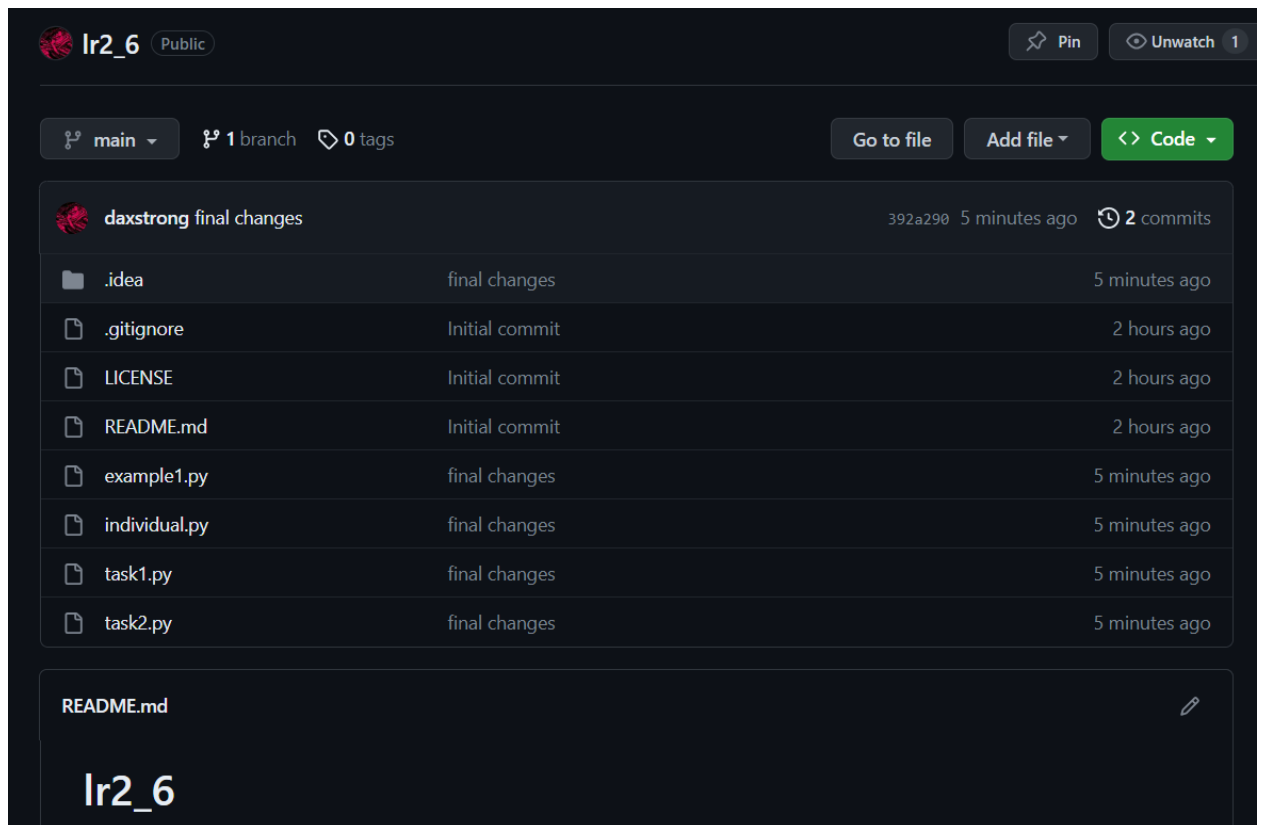


Рисунок 15 – Изменения удаленного репозитория

## Ответы на контрольные вопросы:

### 1. Что такое словари в языке Python?

Словари (dictionaries) в Python – это структуры данных, которые хранят коллекцию пар ключ-значение. Они предоставляют эффективный способ хранения и доступа к данным. Ключи словаря должны быть уникальными и неизменяемыми, а значения могут быть любого типа.

### 2. Может ли функция len() быть использована при работе со словарями?

Да, функция len() может быть использована для определения количества элементов (пар ключ-значение) в словаре. Например:

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
```

```
length = len(my_dict)
```

```
print(length) # Выведет: 3
```

### 3. Какие методы обхода словарей Вам известны?

В Python есть несколько способов обхода словарей:

Цикл for для перебора ключей или элементов словаря.

Методы .keys(), .values() и .items() для получения ключей, значений или пар ключ-значение в виде итерируемых объектов.

### 4. Какими способами можно получить значения из словаря по ключу?

Для получения значения из словаря по ключу можно использовать:

Оператор доступа к элементу по ключу (my\_dict[key]), который вернет значение, связанное с данным ключом.

Метод .get(key), который вернет значение по ключу или None, если ключ отсутствует.

### 5. Какими способами можно установить значение в словаре по ключу?

Чтобы установить значение в словаре по ключу:

Просто присвойте значение ключу: my\_dict[key] = value.

Используйте метод .update() для обновления значений или добавления новых пар ключ-значение.

### 6. Что такое словарь включений?

Словарь включений (dictionary comprehensions) – это способ создания нового словаря с помощью компактного синтаксиса, используя циклы и условия. Например:

```
squares = {x: x*x for x in range(5)} # Создание словаря с квадратами чисел от 0 до 4
```

#### 7. Функция zip() и примеры ее использования.

Функция zip() в Python используется для объединения элементов из нескольких итерируемых объектов в один. Например:

```
names = ['Alice', 'Bob', 'Charlie']
ages = [25, 30, 35]
combined = zip(names, ages) # Объединение имен и возрастов в пары
combined_list = list(combined) # Преобразование объекта zip в список пар
```

#### 8. Модуль datetime и его функционал по работе с датой и временем.

Модуль datetime в Python предоставляет классы для работы с датой, временем и интервалами. Включает классы datetime, date, time и timedelta, позволяющие создавать, обрабатывать и оперировать датами и временем, вычислять разницу между датами, форматировать вывод и многое другое. Например:

```
from datetime import datetime, timedelta
current_time = datetime.now() # Получение текущей даты и времени
future_time = current_time + timedelta(days=7) # Добавление 7 дней к текущей дате
```