

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 РАБОТЕ №2.8**  
**дисциплины «Основы кроссплатформенного программирования»**

Выполнил:

Кондратенко Даниил Витальевич

1 курс, группа ИТС-б-о-22-1,

11.03.02 «Инфокоммуникационные  
технологии и системы связи»,

направленность (профиль)

«Инфокоммуникационные системы и  
сети», очная форма обучения

---

(подпись)

Руководитель практики:

Воронкин Р.А., канд. тех. наук, доцент,  
доцент кафедры инфокоммуникаций

---

(подпись)

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

Ставрополь, 2023 г.

*Тема:* работа с функциями в языке Python.

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

Порядок выполнения работы:

Задание 1.

Изучил теоретический материал работы, создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами.

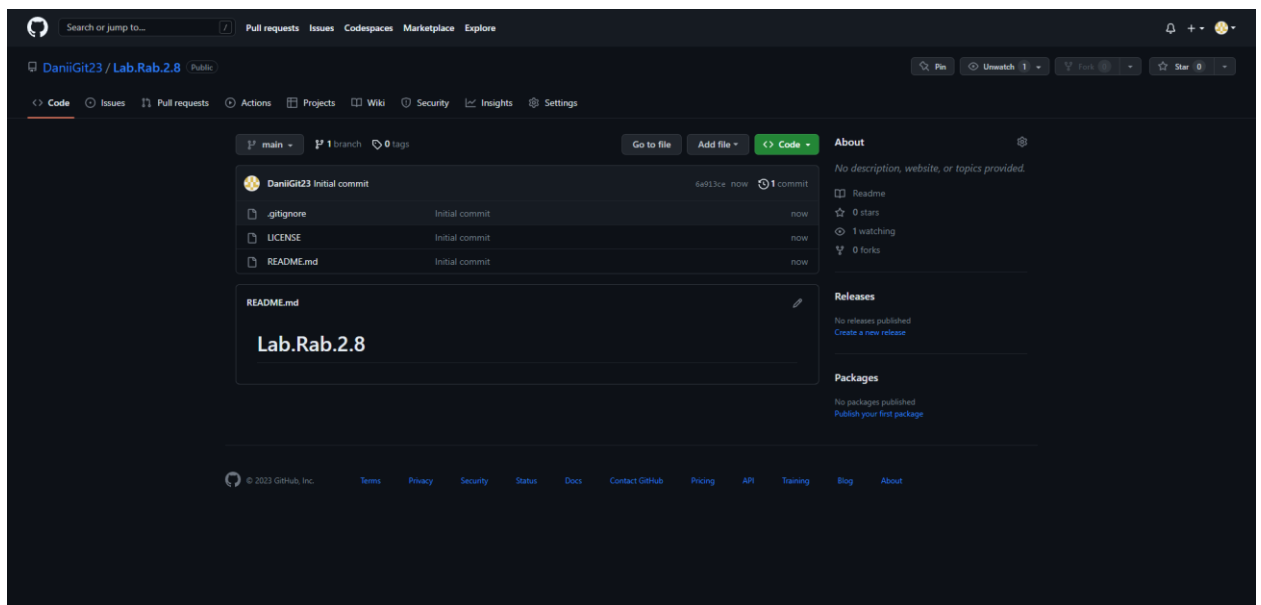


Рисунок 1 – Новый репозиторий

## Задание 2.

Проклонировал свой репозиторий на свой компьютер.

Организовал свой репозиторий в соответствии с моделью ветвления git-flow, появилась новая ветка develop.

```
C:\Users\HUAWEI>git config --global user.name "Daniil"

C:\Users\HUAWEI>git config --global user.email "kondratenko_danil.23@mail.ru"

C:\Users\HUAWEI>git clone https://github.com/DaniiGit23/Lab.Rab.2.8.git
Cloning into 'Lab.Rab.2.8'...
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.

C:\Users\HUAWEI>cd C:\Users\HUAWEI\Lab.Rab.2.8

C:\Users\HUAWEI\Lab.Rab.2.8>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] t
Hooks and filters directory? [C:/Users/HUAWEI/Lab.Rab.2.8/.git/hooks]
```

Рисунок 2 – Клонирование и модель ветвления git-flow

Реализовывал примеры и индивидуальные задания на основе ветки develop, без создания дополнительной ветки feature/(название ветки) по указанию преподавателя.

## Задание 3.

Создал проект PyCharm в папке репозитория.

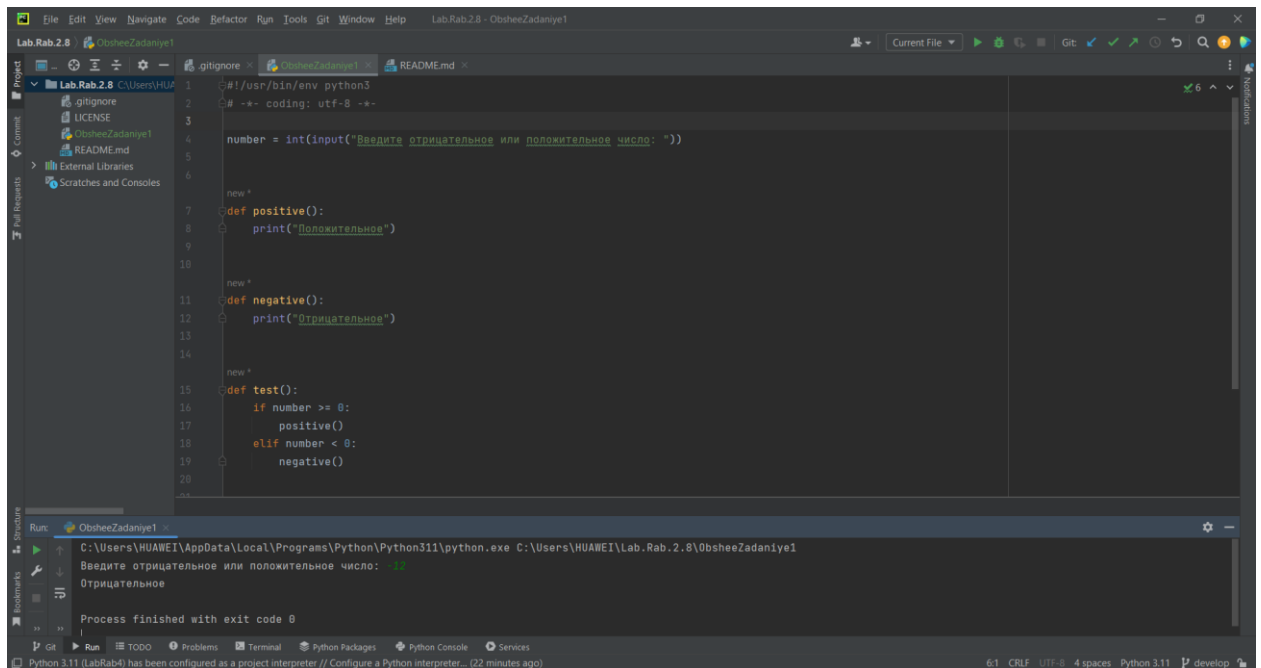
Работа с общим заданием №1.

Добавил новый файл *ObsheeZadaniye1.py*

*Условие задания:* основная ветка программы, не считая заголовков функций, состоит из двух строки кода. Это вызов функции test() и инструкции if \_\_name\_\_ == '\_\_main\_\_'. В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция positive(), тело которой содержит

команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция `negative()`, ее тело содержит выражение вывода на экран слова "Отрицательное". Понятно, что вызов `test()` должен следовать после определения функций. Однако имеет ли значение порядок определения самих функций? То есть должны ли определения `positive()` и `negative()` предшествовать `test()` или могут следовать после него?

Проверьте вашу гипотезу, поменяв объявления функций местами. Попробуйте объяснить результат.



The screenshot shows a code editor with a Python script and its execution output. The script defines three functions: `positive()`, `negative()`, and `test()`. The `test()` function calls `positive()` or `negative()` based on the input number. The output shows the program running successfully and printing "Отрицательное" (Negative).

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

number = int(input("Введите отрицательное или положительное число: "))

def positive():
    print("Положительное")

def negative():
    print("Отрицательное")

def test():
    if number >= 0:
        positive()
    elif number < 0:
        negative()
```

Run: ObsheeZadaniye1

C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\Lab.Rab.2.8\ObsheeZadaniye1

Введите отрицательное или положительное число: -12

Отрицательное

Process finished with exit code 0

Рисунок 3 – Результат программы общего задания №1

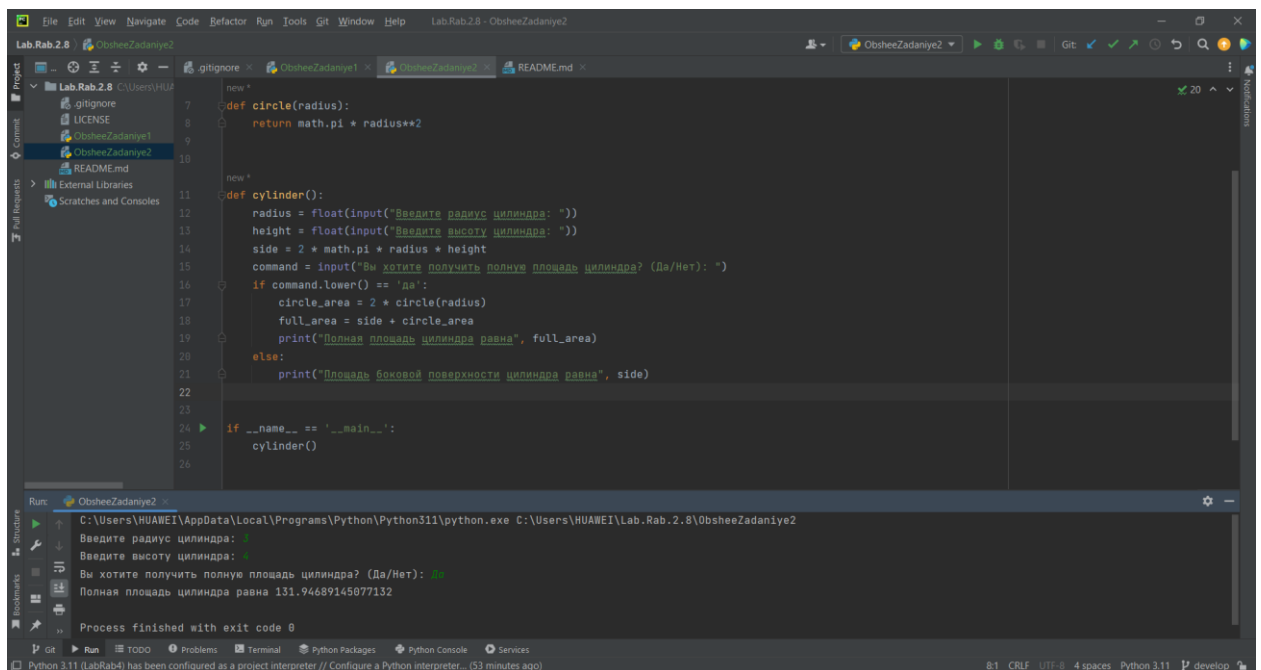
Порядок определения функций не имеет значения, поэтому функции `positive()` и `negative()` могут следовать после вызова функции `test()`. Это связано с тем, что в Python функции создаются целиком при первом выполнении программы, а не в момент определения. При вызове функции `test()` все функции уже будут определены и могут быть использованы.

Задание 4.

Работа с общим заданием №2.

Добавил новый файл *ObsheeZadaniye2.py*

*Условие задания:* в основной ветке программы вызывается функция `cylinder()`, которая вычисляет площадь цилиндра. В теле `cylinder()` определена функция `circle()`, вычисляющая площадь круга по формуле . В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле , или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции `circle()`.



```
def circle(radius):  
    return math.pi * radius**2  
  
def cylinder():  
    radius = float(input("Введите радиус цилиндра: "))  
    height = float(input("Введите высоту цилиндра: "))  
    side = 2 * math.pi * radius * height  
    command = input("Вы хотите получить полную площадь цилиндра? (Да/Нет): ")  
    if command.lower() == 'да':  
        circle_area = 2 * circle(radius)  
        full_area = side + circle_area  
        print("Полная площадь цилиндра равна", full_area)  
    else:  
        print("Площадь боковой поверхности цилиндра равна", side)  
  
if __name__ == '__main__':  
    cylinder()
```

Run: C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\Lab.Rab.2.8\ObsheeZadaniye2

Введите радиус цилиндра: 1  
Введите высоту цилиндра: 1  
Вы хотите получить полную площадь цилиндра? (Да/Нет): да  
Полная площадь цилиндра равна 131.94689145077132

Process finished with exit code 0

Рисунок 4 – Результат программы общего задания №2

Зафиксировал данные изменения.

Задание 5.

Работа с общим заданием №2.

Добавил новый файл *ObsheeZadaniye3.py*

*Условие задания:* напишите программу, в которой определены следующие четыре функции:

1. Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.

2. Функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое `True`. Если нельзя – `False`.
3. Функция `str_to_int()` имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число.
4. Функция `print_int()` имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

В основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция вернула `True`, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.

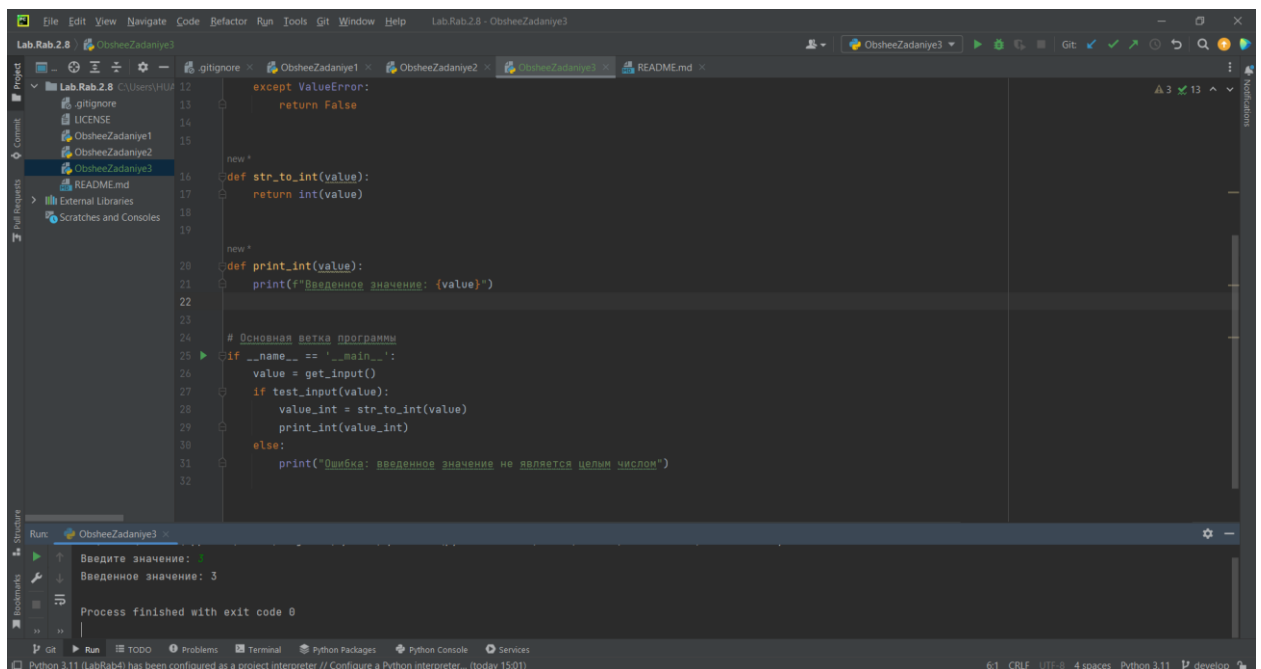


Рисунок 5 – Результат программы общего задания №3

Зафиксировал данные изменения.

Задание 6.

Работа с пример №1.

Добавил новый файл *primer1.py*

*Условие задания:* для примера 1 лабораторной работы 2.6, оформить каждую команду в виде вызова отдельной функции.

## Что было в 2.6.

**Пример 1.** *Использовать словарь, содержащий следующие ключи: фамилия и инициалы работника; название занимаемой должности; год поступления на работу. Написать программу, выполняющую следующие действия:*

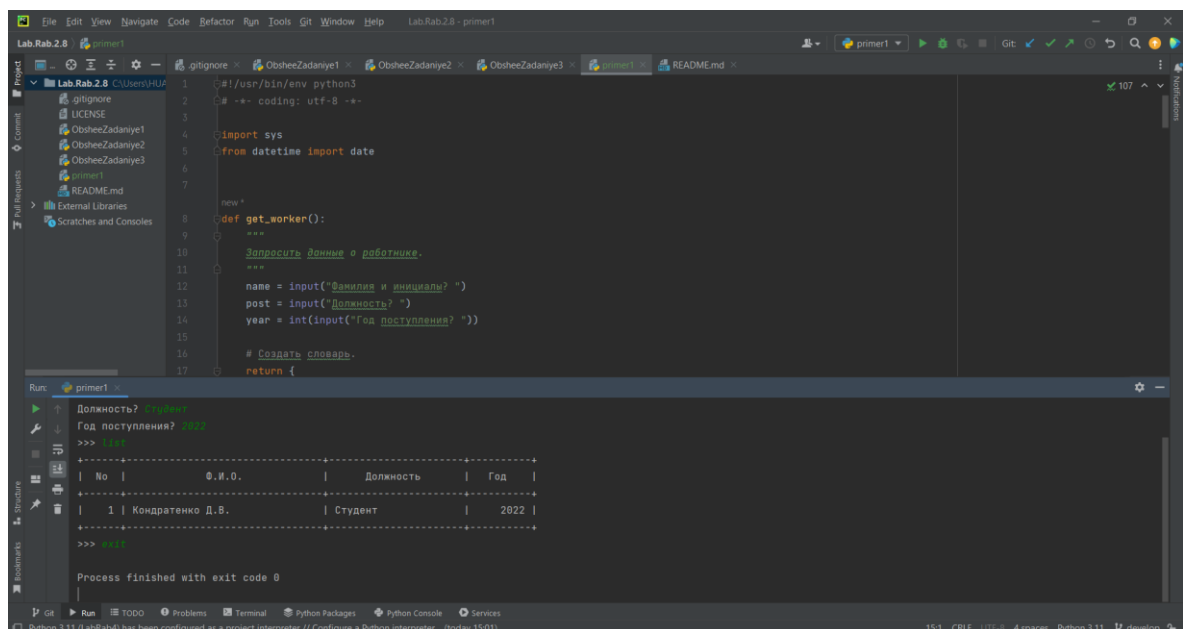
- ввод с клавиатуры данных в список, состоящий из заданных словарей;
- записи должны быть размещены по алфавиту;
- вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры;
- если таких работников нет, вывести на дисплей соответствующее сообщение.

**Решение:** Определим следующие ключи для словарей:

- *name* - фамилия и инициалы работника;
- *post* - название занимаемой должности;
- *year* - год поступления.

Введем следующие команды для работы со списком словарей в интерактивном режиме:

- *add* - запросить информацию о сотруднике с клавиатуры и добавить в список, поддерживая список в отсортированном состоянии;
- *list* - вывести на экран содержимое списка словарей;
- *select* - вывести на дисплей фамилий работников, чей стаж работы в организации превышает заданное значение, при этом это значение должно быть аргументом команды *select* и отделено от нее пробелом;
- *help* - вывести на дисплей список команд с описанием;
- *exit* - завершить работу программы.



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

import sys
from datetime import date

new = 1
def get_worker():
    """
    Запросить данные о работнике.
    """
    name = input("Фамилия и инициалы? ")
    post = input("Должность? ")
    year = int(input("Год поступления? "))

    # Создать словарь.
    return {
```

Run: primer1

```
>>> list
Должность? студент
Год поступления? 2022
>>> list
-----
| No |      О.И.О.      | Должность | Год |
-----
|  1 | Кондратенко Д.В. | Студент   | 2022 |
-----
>>> list
Process finished with exit code 0
```

Рисунок 6 – Результат программы примера №1

Зафиксировал данные изменения.

## Задание 7.

Выполнение индивидуального задания.

Создал файл под названием *individual.py*

Оформить в виде функций индивидуальное задание из лабораторной работы №2.6.

*Условие задание (2.6):* 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.

Та же программа только в виде функций.

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

def input_data():
    """
    Функция для ввода данных в список из словарей заданной структуры
    """
    data = []
    while True:
        person = {}

        person['surname'] = input('Введите фамилию: ')
        person['name'] = input('Введите имя: ')
        person['phone_number'] = input('Введите номер телефона: ')

        birthdate = input('Введите дату рождения через пробел (день месяц год): ')
        birthdate = birthdate.split()
        birthdate = [int(i) for i in birthdate]
        person['birthdate'] = birthdate

        data.append(person)

        add_more = input('Хотите добавить еще одного человека? (да/нет) ')
        if add_more.lower() == 'нет':
            break

    # Сортировка по фамилии и имени
    data = sorted(data, key=lambda x: (x['surname'], x['name']))

    return data

def get_birthdays(data):
    """
    Функция для вывода информации о людях, чьи дни рождения приходятся на
    месяц, значение которого введено с клавиатуры
    """
```



```

month = int(input('Введите месяц: '))

birthdays = []
for person in data:
    if person['birthdate'][1] == month:
        birthdays.append(person)

if len(birthdays) == 0:
    print('Нет людей с таким днем рождения')
else:
    print('Люди, у которых день рождения в выбранном месяце:')
    for person in birthdays:
        print(f"{person['surname']} {person['name']}: "
              f"{person['birthdate'][0]}.{person['birthdate'][1]}.{person['birthdate'][2]}")

def main():
    """
    Основная функция
    """
    data = input_data()
    get_birthdays(data)

if __name__ == '__main__':
    main()

```

```

Run: ind x
C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\Lab.Rab.2.8\ind
Введите фамилию: Кондратенко
Введите имя: Даниил
Введите номер телефона: 12-14-64
Введите дату рождения через пробел (день месяц год): 23 01 2004
Хотите добавить еще одного человека? (да/нет) да
Введите фамилию: Иванова
Введите имя: Николай
Введите номер телефона: 12-14-74
Введите дату рождения через пробел (день месяц год): 12 01 2003
Хотите добавить еще одного человека? (да/нет) нет
Введите месяц: 01
Люди, у которых день рождения в выбранном месяце:
Иванов Николай: 12.1.2003
Кондратенко Даниил: 23.1.2004
Process finished with exit code 0

```

Рисунок 7 – Результат программы

Зафиксировал данные изменения.

## Задание 8.

Слил ветку develop с веткой main и отправил на удаленный сервер.

```
C:\Users\HUAWEI\Lab.Rab.2.8>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)

C:\Users\HUAWEI\Lab.Rab.2.8>git merge develop
Merge made by the 'ort' strategy.
ObsheeZadaniye1 | 23 ++++++++
ObsheeZadaniye2 | 25 ++++++++
ObsheeZadaniye3 | 31 ++++++++
individual       | 63 ++++++++
primer1         | 136 ++++++++
5 files changed, 278 insertions(+)
create mode 100644 ObsheeZadaniye1
create mode 100644 ObsheeZadaniye2
create mode 100644 ObsheeZadaniye3
create mode 100644 individual
create mode 100644 primer1

C:\Users\HUAWEI\Lab.Rab.2.8>git push
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 8 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (22/22), 5.07 KiB | 2.54 MiB/s, done.
Total 22 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (8/8), completed with 1 local object.
To https://github.com/DaniiGit23/Lab.Rab.2.8.git
 6a913ce..663af33  main -> main

C:\Users\HUAWEI\Lab.Rab.2.8>
```

Рисунок 8 – Слияние веток

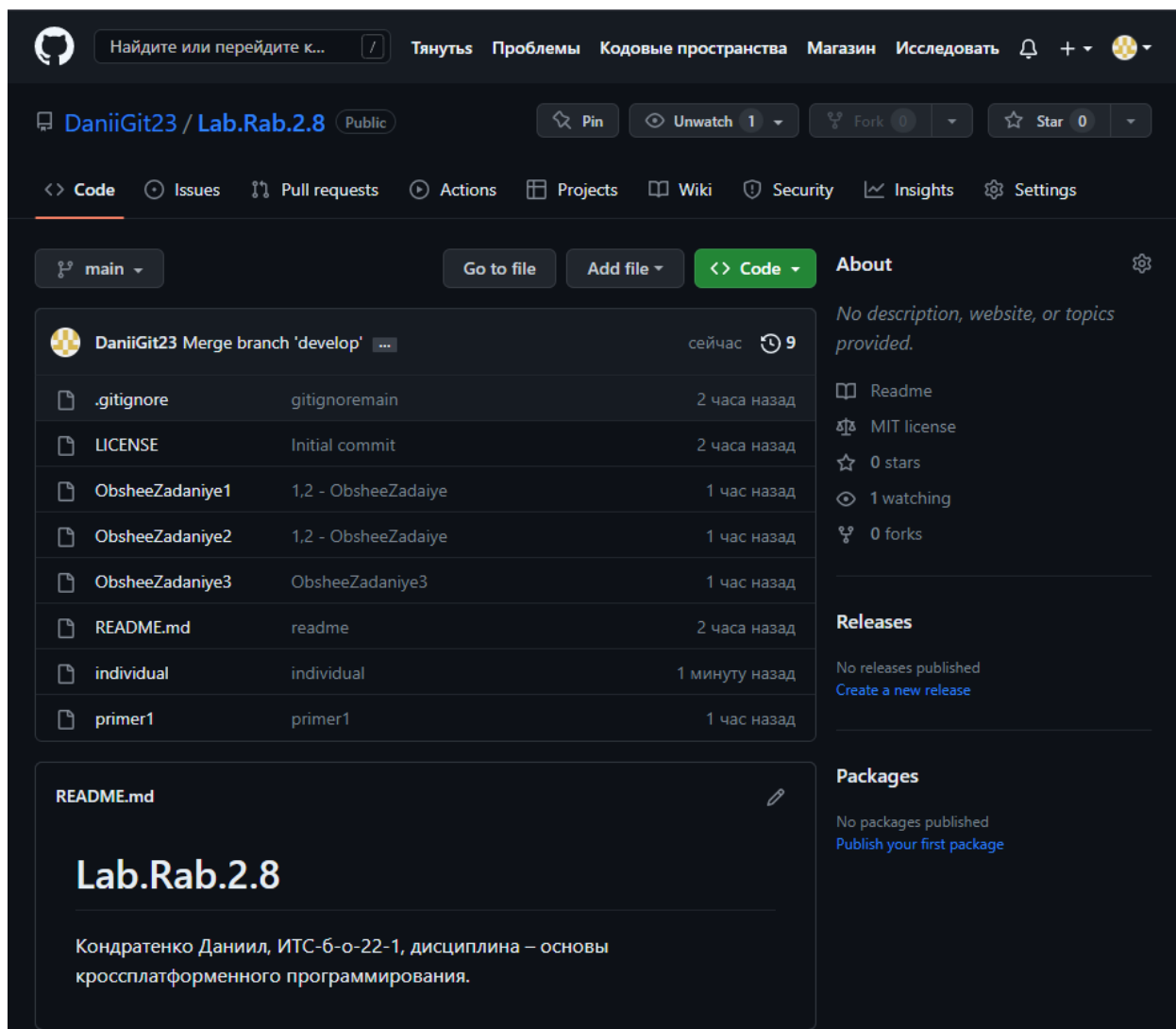


Рисунок 9 – Удаленный сервер

Ссылка на репозиторий: <https://github.com/DaniiGit23/Lab.Rab.2.8.git>

Контрольные вопросы:

1) Каково назначение функций в языке программирования Python?

Функции в языке программирования Python используются для группировки набора инструкций в одну логическую единицу, которая может быть вызвана из любой части программы. Использование функций позволяет избежать повторного кода, улучшить читаемость программы, сделать код более организованным и легким для тестирования и сопровождения. Также функции могут быть использованы для создания модулей, которые могут быть использованы в различных проектах.

2) Каково назначение операторов def и return?

Оператор `def` в языке программирования Python используется для определения функции. Функция представляет собой блок кода с именем, который может вызываться из других частей программы. Оператор `return` внутри функции позволяет вернуть значение из функции.

Проще говоря, оператор `def` используется для создания функций, а оператор `return` - для возвращения результатов работы функции, которые могут быть использованы в другой части программы. Функции позволяют разбить код на более мелкие логические блоки, повторно использовать код, упрощать чтение и понимание программы в целом.

3) Каково назначение локальных и глобальных переменных при написании функций в Python?

Локальные переменные - это переменные, которые определены внутри функции и могут быть использованы только внутри этой функции. Они не видны за ее пределами и существуют только во время выполнения функции.

Глобальные переменные - это переменные, которые определены вне функции и могут быть использованы в любой части программы, включая функции. Они существуют в течение всего времени работы программы.

Назначение локальных переменных - это изолировать код функции от других частей программы, чтобы избежать изменений переменных из других частей программы, которые могут негативно повлиять на работу функции.

Назначение глобальных переменных - это обеспечить доступ к данному объекту из любой части программы. Однако, существует опасность перезаписи глобальных переменных, и использование глобальных переменных следует использовать с осторожностью.

4) Как вернуть несколько значений из функции Python?

В Python можно вернуть несколько значений из функции, используя кортеж. Для этого возвращаемые значения перечисляются через запятую внутри круглых скобок. В выбираемые значения можно обратиться по индексу.

#### 5) Какие существуют способы передачи значений в функцию?

В языке программирования Python значения могут быть переданы в функцию несколькими способами:

- позиционные аргументы (передача аргументов в порядке их следования);
- именованные аргументы (передача аргументов с указанием их имени);
- аргументы по умолчанию (передача аргументов со значениями по умолчанию);
- распаковывание списков и словарей.

#### 6) Как задать значение аргументов функции по умолчанию?

Для того, чтобы задать значение аргументов функции по умолчанию в Python, нужно указать это значение после имени аргумента в определении функции.

#### 7) Каково назначение lambda-выражений в языке Python?

Lambda-выражения в языке Python представляют собой способ создания анонимных функций без явного определения имени функции. Они могут использоваться как аргументы встроенных функций, таких как `filter()`, `map()` и `reduce()`.

В lambda-выражениях объединяются три элемента: аргументы, оператор-разделитель и тело функции.

#### 8) Как осуществляется документирование кода согласно PEP257?

Документирование кода в Python оформляется с помощью docstring'ов (строки документации), которые помещаются сразу после объявления функции, класса, метода или модуля. Для того чтобы оформить документацию в соответствии с PEP257, используют следующие рекомендации:

1. Docstring должен начинаться с однострочного описания того, что делает объект (функция, класс и т.д.). Это описание следует начинать с заглавной буквы и заканчивать точкой.

2. За однострочным описанием должна следовать пустая строка.

3. Если это функция или метод, то следует указать, какие аргументы она принимает, какие они должны быть по типу и за что они отвечают.

4. Если функция или метод что-то возвращает, то это также должно быть указано в документации.

5. Если объект имеет какие-то особенности или ограничения, то их нужно описать.

6. Если есть примеры использования, то их нужно привести.

9) В чем особенность однострочных и многострочных форм строк документации?

Однострочная форма документации заключается в одном ряду и применяется для краткого описания. Она начинается со знака # и двух пробелов, специально размещенных после знака #. Многострочная форма документации позволяет вставлять более детальные описания. Она начинается и заканчивается тройными кавычками и предоставляет возможность размещения внутри описания более одного абзаца. Эта форма чаще применяется при описании функций и модулей.

*Вывод:* приобретение навыков по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.