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 №2.8
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 функциями в языке Python.

Цель работы: приобретение навыков по работе с функциями 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:

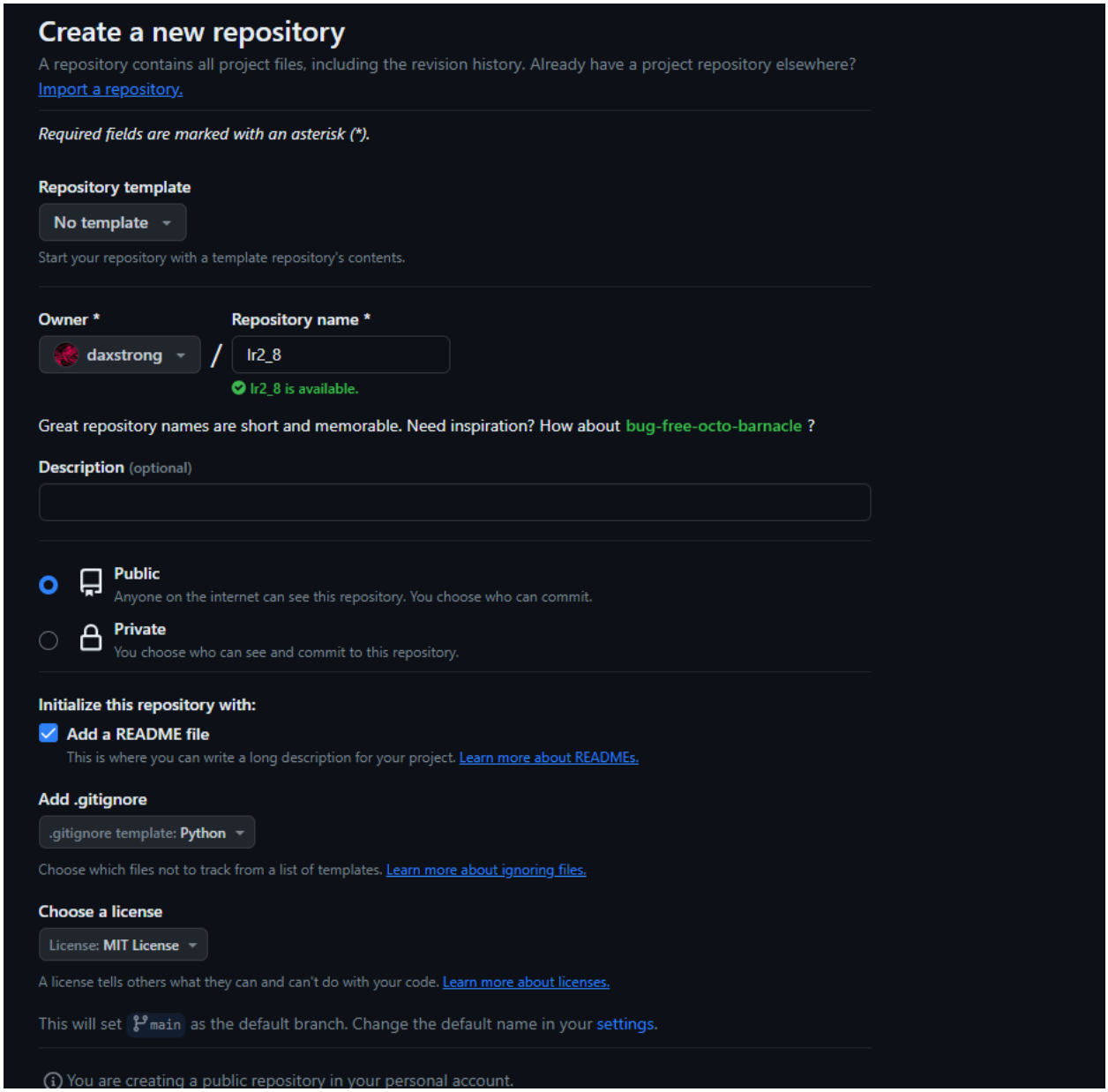


Рисунок 1 – Создание репозитория с заданными настройками

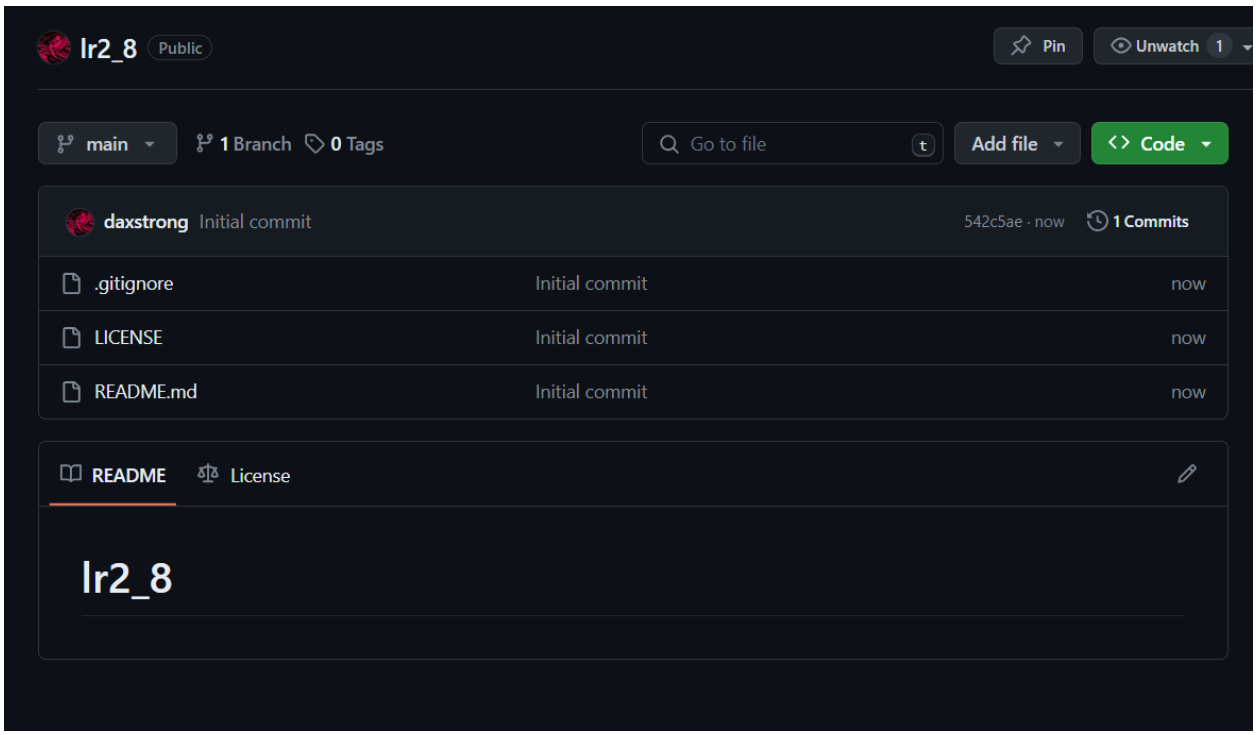


Рисунок 2 – Созданный репозиторий

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии
$ git clone https://github.com/daxstrong/lr2_8.git
Cloning into 'lr2_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.
```

Рисунок 3 – Клонирование репозитория

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_8 (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:

Листинг примера:

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

import sys
from datetime import date


def get_worker():
    """
    Запросить данные о работнике.
    """

    name = input("Фамилия и инициалы? ")
    post = input("Должность? ")
    year = int(input("Год поступления? "))
    # Создать словарь.
    return {
        'name': name,
        'post': post,
        'year': year,
    }


def display_workers(staff):
    """
    Отобразить список работников.
    """

    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)
        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )
        print(line)
    else:
        print("Список работников пуст.")


def select_workers(staff, period):
```

```
    """
    Выбрать работников с заданным стажем.
    """

    # Получить текущую дату.
    today = date.today()
    # Сформировать список работников.
    result = []
    for employee in staff:
        if today.year - employee.get('year', today.year) >= period:
            result.append(employee)
    # Возвратить список выбранных работников.
    return result


def main():
    """
    Главная функция программы.
    """

    # Список работников.
    workers = []
    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()
        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break
        elif command == 'add':
            # Запросить данные о работнике.
            worker = get_worker()
            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))
        elif command == 'list':
            # Отобразить всех работников.
            display_workers(workers)
        elif command.startswith('select '):
            # Разбить команду на части для выделения стажа.
            parts = command.split(' ', maxsplit=1)
            # Получить требуемый стаж.
            period = int(parts[1])
            # Выбрать работников с заданным стажем.
            selected = select_workers(workers, period)
            # Отобразить выбранных работников.
            display_workers(selected)
        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить работника;")
            print("list - вывести список работников;")
            print("select <стаж> - запросить работников со стажем;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)


if __name__ == '__main__':
    main()
```

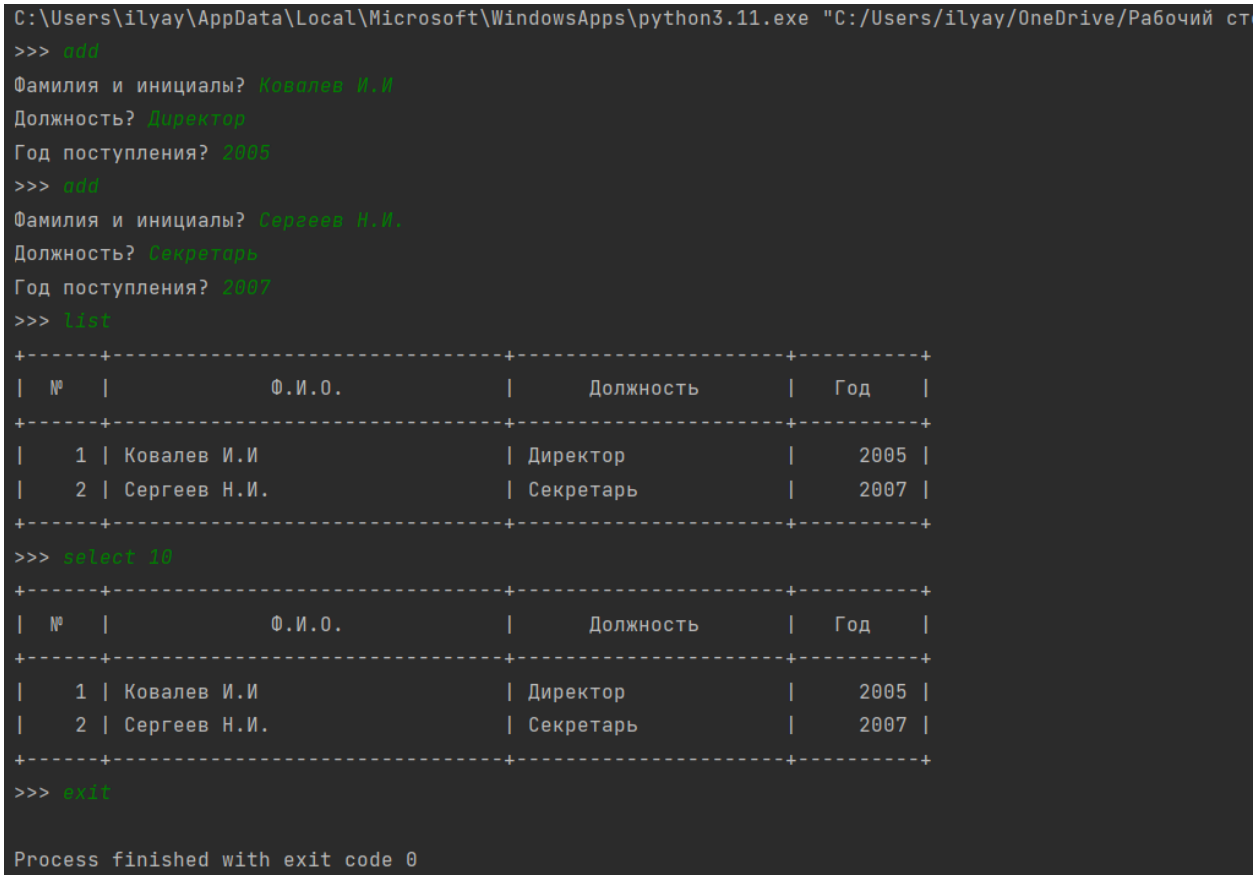


Рисунок 5 – Вывод программы

3. Решите задачу: основная ветка программы, не считая заголовков функций, состоит из двух строк 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".

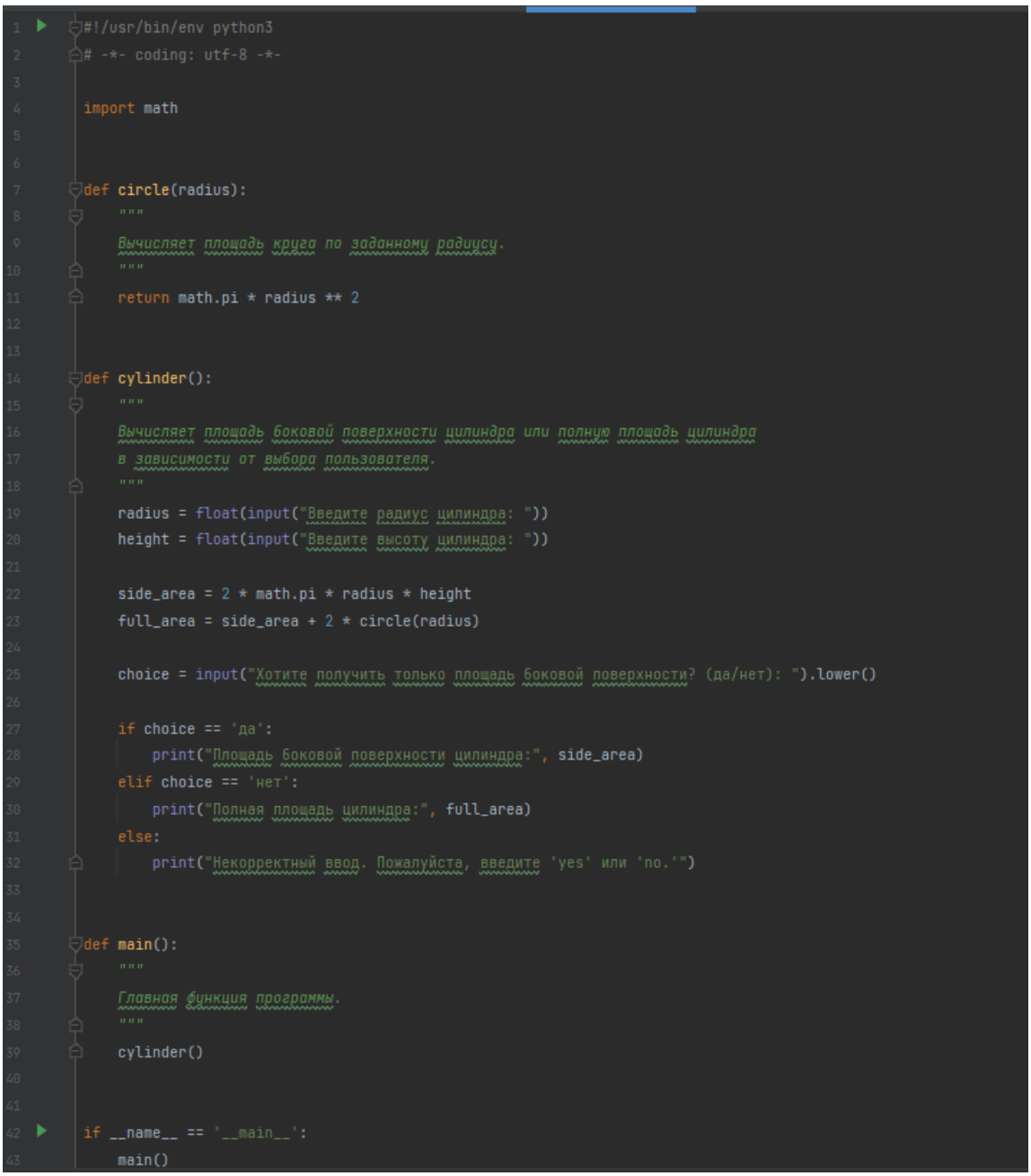
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def test():
    num = int(input("Введите целое число: "))
    if num > 0:
        positive()
    elif num < 0:
        negative()


def positive():
    print("Положительное")


def negative():
    print("Отрицательное")


if __name__ == '__main__':
    test()
```

Рисунок 6 – Задание №1

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Введите целое число: 23
Положительное
```

Рисунок7 – Вывод программы (Задание №1)

4. Решите следующую задачу: 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().



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

import math


def circle(radius):
    """
    Вычисляет площадь круга по заданному радиусу.
    """
    return math.pi * radius ** 2


def cylinder():
    """
    Вычисляет площадь боковой поверхности цилиндра или полную площадь цилиндра
    в зависимости от выбора пользователя.
    """
    radius = float(input("Введите радиус цилиндра: "))
    height = float(input("Введите высоту цилиндра: "))

    side_area = 2 * math.pi * radius * height
    full_area = side_area + 2 * circle(radius)

    choice = input("Хотите получить только площадь боковой поверхности? (да/нет): ").lower()

    if choice == 'да':
        print("Площадь боковой поверхности цилиндра:", side_area)
    elif choice == 'нет':
        print("Полная площадь цилиндра:", full_area)
    else:
        print("Некорректный ввод. Пожалуйста, введите 'yes' или 'no.'")


def main():
    """
    Главная функция программы.
    """
    cylinder()


if __name__ == '__main__':
    main()
```

Рисунок 8 – Задание №2

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Введите радиус цилиндра: 6
Введите высоту цилиндра: 8
Хотите получить только площадь боковой поверхности? (да/нет): нет
Полная площадь цилиндра: 527.7875658030853
```

Рисунок 9 – Вывод программы (Задание №2)

5. Решите следующую задачу: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.

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

def multiply_until_zero():
    """
    Считывает числа с клавиатуры и перемножает их до ввода 0.
    Возвращает полученное произведение.
    """
    product = 1
    while True:
        num = float(input("Введите число (для завершения введите 0): "))
        if num == 0:
            break
        product *= num
    return product


def main():
    """
    Главная функция программы.
    """
    result = multiply_until_zero()
    print(f"Произведение введенных чисел: {result}")


if __name__ == '__main__':
    main()
```

Рисунок 10 – Задание №3

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Введите число (для завершения введите 0): 2
Введите число (для завершения введите 0): 3
Введите число (для завершения введите 0): 4
Введите число (для завершения введите 0): 0
Произведение введенных чисел: 24.0
```

Рисунок 11 – Вывод программы (Задание №3)

6. Решите следующую задачу: 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.

Листинг программы:

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

def get_input():
    """
    Запрашивает у пользователя ввод значения и возвращает введенную строку.
    """
    user_input = input("Введите значение: ")
    return user_input


def test_input(value):
    """
    Проверяет, можно ли преобразовать значение в целое число.
    Возвращает True, если возможно, в противном случае - False.
    """
    try:
        int(value)
        return True
    except ValueError:
        return False


def str_to_int(value):
    """
    Преобразует переданное значение в целое число.
    Возвращает полученное целое число.
    """
    return int(value)
```

```python
def print_int(value):
    """
    Выводит переданное значение на экран.
    """
    print(value)


def main():
    """
    Главная функция программы.
    """
    # Получаем ввод от пользователя
    user_value = get_input()

    # Проверяем, можно ли преобразовать введенное значение в целое число
    if test_input(user_value):
        # Если возможно, преобразуем строку в целое число
        int_value = str_to_int(user_value)
        # Выводим полученное целое число на экран
        print_int(int_value)
    else:
        print("Невозможно преобразовать введенное значение в целое число.")


if __name__ == '__main__':
    main()
```

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/ilyay/OneDrive/Рабочий
Введите значение: 543653561
543653561
```

Рисунок 12 – Вывод программы с числом

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/ilyay/OneDrive/Рабочий
Введите значение: dydgyg
Невозможно преобразовать введенное значение в целое число.
```

Рисунок 13 – Вывод программы со строкой

7. Выполним индивидуальные задания:

Листинг программы:

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

import sys
from datetime import datetime


def exit_program():
    """
    Выход из программы.
    """
    sys.exit()


def add_person(people):
```

```
    """
    Добавление информации о человеке.
    """
    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'])


def list_people(people):
    """
    Вывод списка всех людей.
    """
    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)


def select_people_by_month(people, month_to_search):
    """
    Вывод людей с днем рождения в указанном месяце.
    """
    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.")


def help_info():
    """
    Вывод справочной информации о командах.
    """
    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.")
```

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

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

        match command:
            case 'exit':
                exit_program()
            case 'add':
                add_person(people)
            case 'list':
                list_people(people)
            case command if command.startswith('select '):
                month_to_search = int(command.split(' ')[1])
                select_people_by_month(people, month_to_search)
            case 'help':
                help_info()
            case _:
                print(f"Неизвестная команда {command}", file=sys.stderr)
```

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/ilyay/OneDrive/Рабочий стол
>>> add
Фамилия: Ковалев
Имя: Дмитрий
Номер телефона: 7837984798
Дата рождения (в формате ДД.ММ.ГГГГ): 02.03.2000
>>> add
Фамилия: Смирнов
Имя: Александр
Номер телефона: 798728738
Дата рождения (в формате ДД.ММ.ГГГГ): 12.12.1999
>>> list
+--------------------------+----------------+--------------------------+
|         Фамилия          |       Имя      |       Дата рождения      |
+--------------------------+----------------+--------------------------+
|         Ковалев          |     Дмитрий    |        02.03.2000        |
+--------------------------+----------------+--------------------------+
|         Смирнов          |    Александр   |        12.12.1999        |
+--------------------------+----------------+--------------------------+
>>>
```

Рисунок 14 – Вывод программы

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

```
$ git log
commit 7ded39683a84f69f6deb0f5437aaf99c56285830 (HEAD -> develop)
Author: dexstrong <ilya.yurev.04@inbox.ru>
Date:   Mon Dec 18 19:44:09 2023 +0300

    Финальные изменения

commit 11ce381375f45cea3206ef4758c16aed547adf4d
Author: dexstrong <ilya.yurev.04@inbox.ru>
Date:   Sun Dec 17 22:09:53 2023 +0300

    Добавлена программа для определения знака числа

commit 542c5ae4e6745fbfb8b5baff596eab311e66cf84 (origin/main, origin/HEAD, main)
Author: Ilya Yurev <112946692+daxstrong@users.noreply.github.com>
Date:   Sat Dec 16 20:01:48 2023 +0300

    Initial commit
```

Рисунок 15 – Коммиты ветки develop во время выполнения лабораторной работы

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_8 (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_8 (main)
$ git merge develop
Updating 542c5ae..7ded396
Fast-forward
 .idea/.gitignore                                |   8 ++
 .idea/inspectionProfiles/profiles_settings.xml |   6 ++
 .idea/lr2_8.iml                                 |   8 ++
 .idea/misc.xml                                  |   4 +
 .idea/modules.xml                               |   8 ++
 .idea/vcs.xml                                   |   6 ++
 ex1.py                                          | 126 ++++++++++++++++++++++++++
 individual.py                                   |  97 ++++++++++++++++++++
 task1.py                                        |  21 +++++
 task2.py                                        |  43 +++++++++
 task3.py                                        |  27 ++++++
 task4.py                                        |  57 ++++++++++++
 12 files changed, 411 insertions(+)
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
 create mode 100644 .idea/lr2_8.iml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 ex1.py
 create mode 100644 individual.py
 create mode 100644 task1.py
 create mode 100644 task2.py
 create mode 100644 task3.py
 create mode 100644 task4.py
```

Рисунок 16 – Слияние веток main и develop

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr2_8 (main)
$ git push origin main
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 12 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (18/18), 6.33 KiB | 6.33 MiB/s, done.
Total 18 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/daxstrong/lr2_8.git
   542c5ae..7ded396  main -> main
```

Рисунок 17 – Отправка изменений на удаленный репозиторий

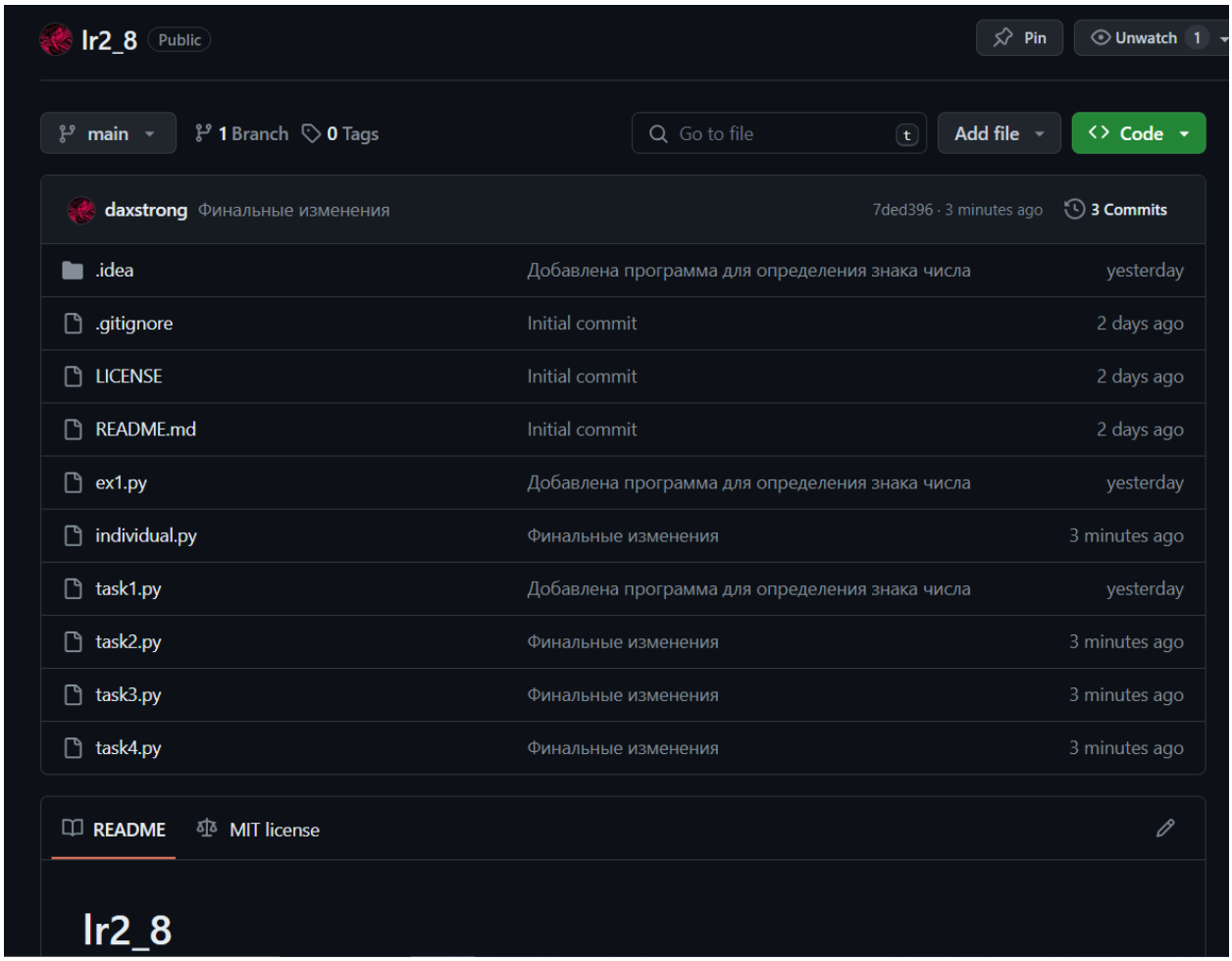


Рисунок 18 – Изменения удаленного репозитория

Ответы на контрольные вопросы:

1. Что такое множества в языке Python?

Множество (set) в Python – это неупорядоченная коллекция уникальных элементов. Они используются для выполнения операций над уникальными элементами без дублирования.

2. Как осуществляется создание множеств в Python?

Множество можно создать, используя фигурные скобки {} и перечислив элементы множества через запятую. Например:

```
my_set = {1, 2, 3, 4}
```

3. Как проверить присутствие/отсутствие элемента в множестве?

Для проверки присутствия элемента в множестве можно использовать оператор in. Например:

```
my_set = {1, 2, 3, 4}
print(3 in my_set) # Выведет: True
print(5 not in my_set) # Выведет: True
```

4. Как выполнить перебор элементов множества?

Можно использовать цикл for для перебора элементов множества:

```
my_set = {1, 2, 3, 4}
for element in my_set:
 print(element)
```

5. Что такое set comprehension?

Set comprehension – это способ создания множества с использованием компактного синтаксиса, аналогичного списочным включениям. Например:

```
my_set = {x for x in range(10) if x % 2 == 0} # Создание множества четных чисел от 0 до 9
```

6. Как выполнить добавление элемента во множество?

Для добавления элемента в множество используется метод .add().

Например:

```
my_set = {1, 2, 3}
my_set.add(4)
```

7. Как выполнить удаление одного или всех элементов множества?

Методы .remove() и .discard() используются для удаления одного элемента, а метод .clear() - для удаления всех элементов множества. Например:

my_set = {1, 2, 3, 4}

my_set.remove(3)

my_set.discard(5) # Если элемент отсутствует, discard() не вызывает ошибку

my_set.clear() # Очистка множества

8. Как выполняются основные операции над множествами: объединение, пересечение, разность?

Объединение: set1.union(set2) или оператор |

Пересечение: set1.intersection(set2) или оператор &

Разность: set1.difference(set2) или оператор -

9. Как определить, что некоторое множество является надмножеством или подмножеством другого множества?

Методы .issubset() и .issuperset() используются для определения того, чтобы определить является ли одно множество подмножеством или надмножеством другого соответственно.

10. Каково назначение множеств frozenset?

frozenset – это неизменяемая версия множества. Однажды созданное frozenset не может быть изменено, но оно может быть использовано в качестве ключа в словарях или как элемент в другом множестве.

11. Как осуществляется преобразование множеств в строку, список, словарь?

Для преобразования множества в список можно использовать list(my_set). Преобразование в строку: str(my_set).

Множество не может быть преобразовано непосредственно в словарь, но можно создать словарь из множества ключей.