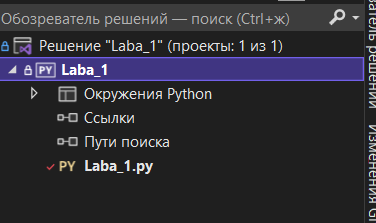
**Лабораторная работа №1**

Создание консольного приложения в IDE Visual Studio. Работа с системой контроля версий

**Цель**: изучение IDE Visual Studio, создание простейшего приложения на Python, подключение Git.

**Выполнение работы:**

Создали проект:



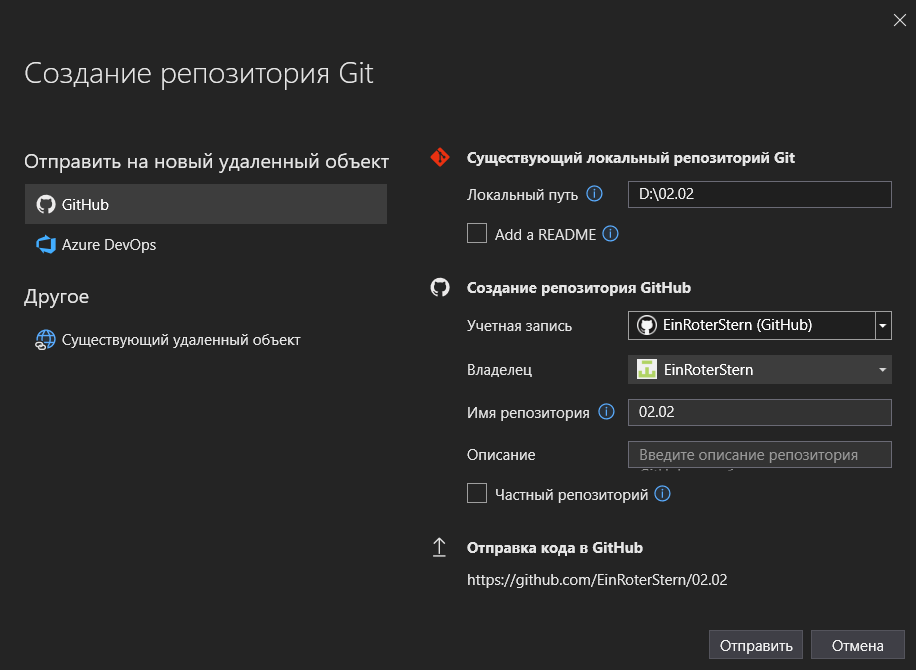
Запускаем код

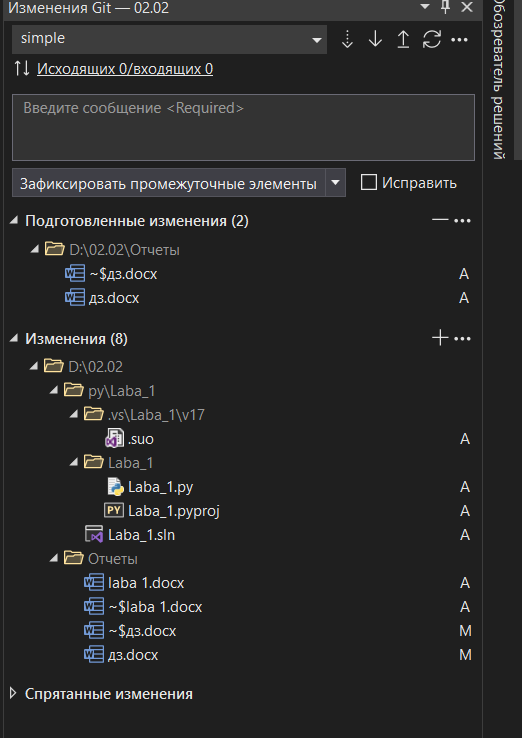


Результат работы:

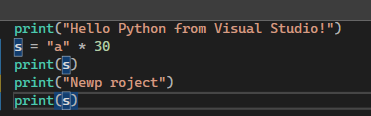


Для того, чтобы файлы приложения отслеживались системой контроля версий Git, выбрать **Git** **Changes** и нажать **Create Git Repository**:

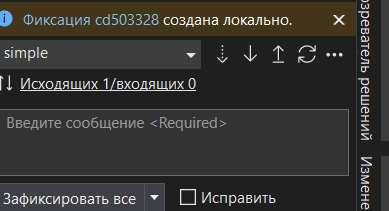


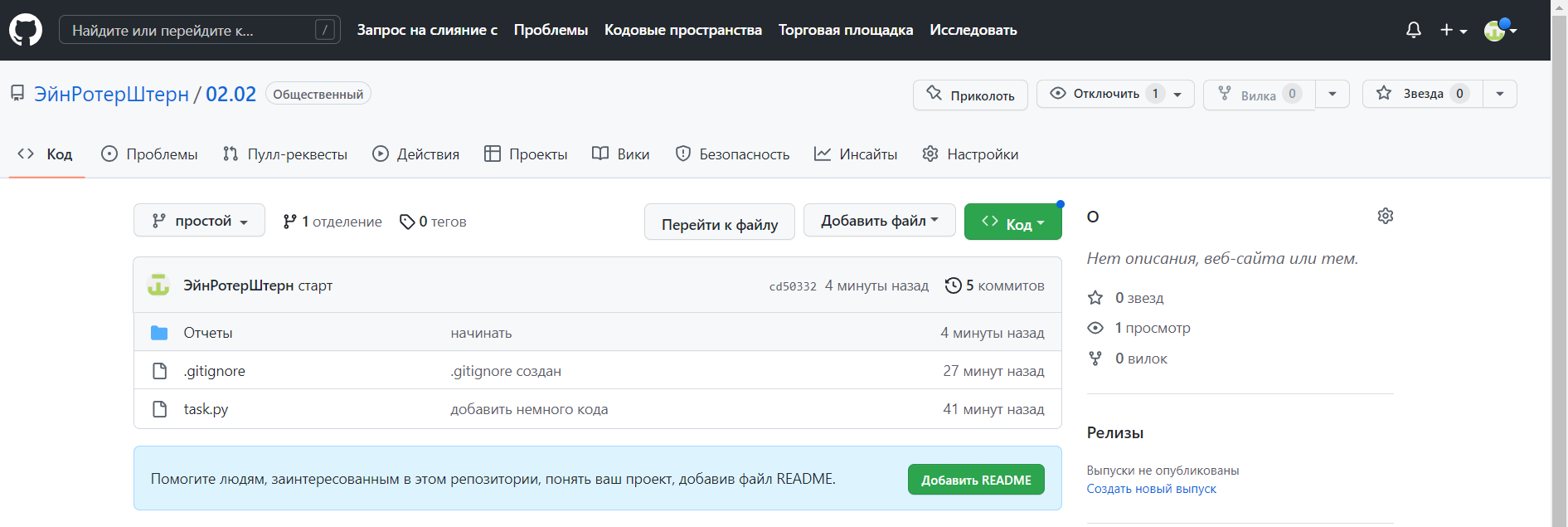


Далее добавить несколько строк кода и выполнить **коммит** с комментарием (например, «Start»):



В заголовке Git Changes высветится сообщение о создании коммита. Если вдруг потребуется изменить комментарий к нему, то можно это сделать, выбрав опцию **Amend**:







Добавить строки кода одной из решаемых ранее по своему варианту задач на Python. Выполнить коммит. Просмотреть историю коммитов, нажав на цифру со стрелкой вверх:

print("Даны две строки в одной список в другой символы, нужно из первой строки убрать все символы которые написаны во второй")

l = "Медведь, паскаль, дерево, программа"

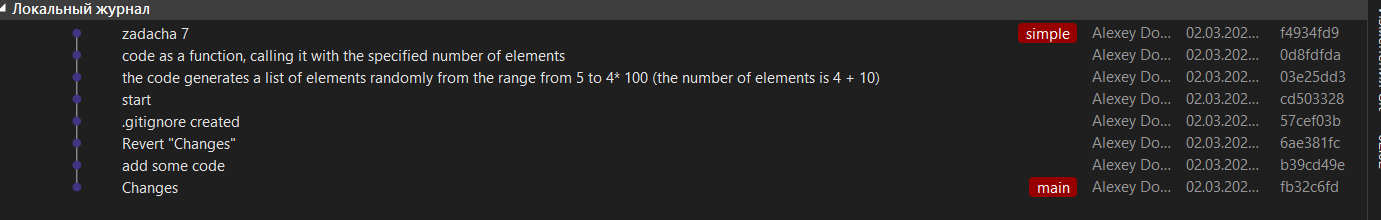
l2 = ","

print("\nстрока - "+ "["+l+"]" + "\nсимволы - "+"["+l2+"]")

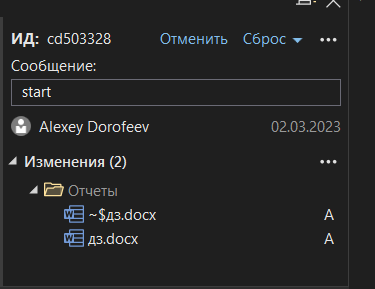
for i in l2:

l = l.replace(",", "")

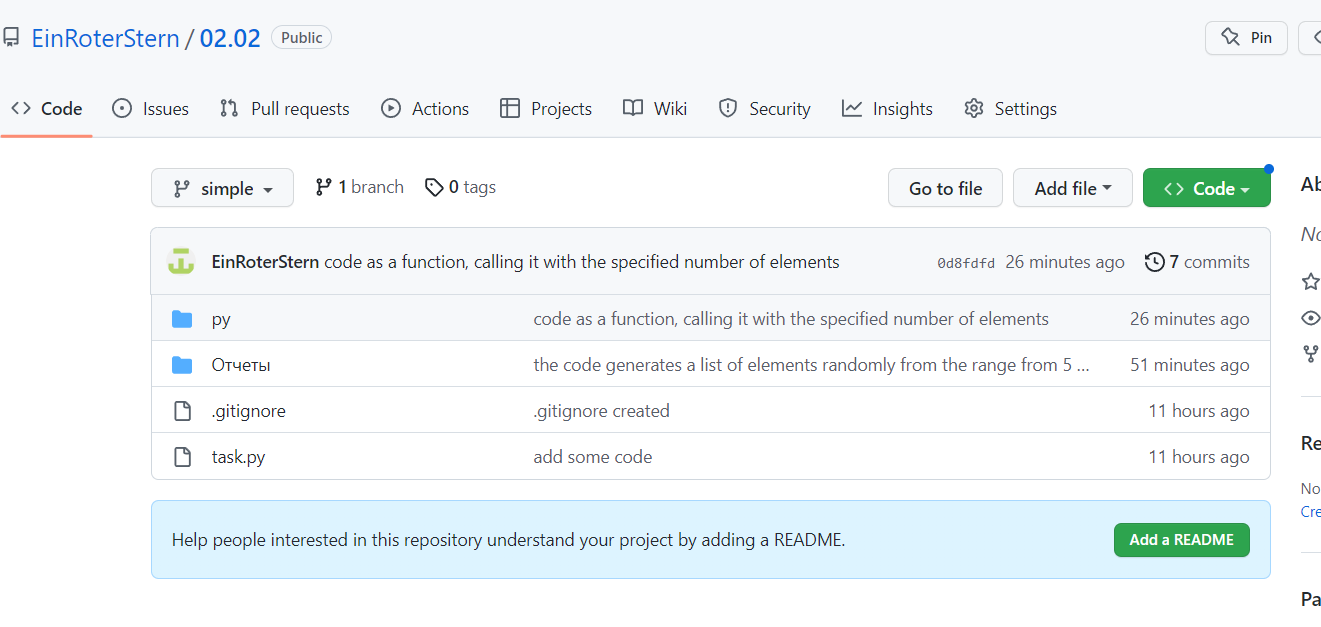
print(l)

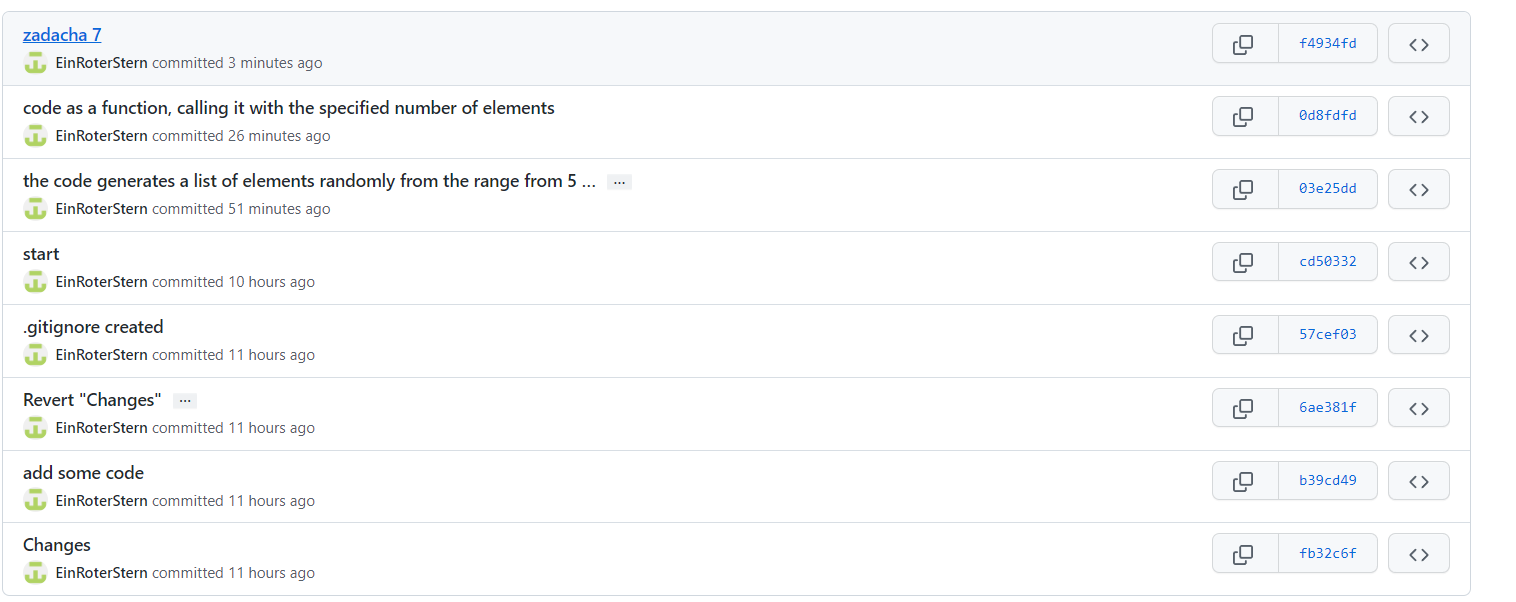


Выбрав один из коммитов, просмотреть его детали с помощью команды контекстного меню:



Перейти на Github и продемонстрировать результат. Должен быть создать новый репозиторий:





Просмотреть содержимое файла **.gitignore**. Внести в отчёт вывод о том, какого рода файла не отслеживаются системой Git при работе с IDE Visual Studio

Файлы конфигурации IDE

Файлы проекта Visual Studio

Файлы резервных копий

Файлы временных данных

11 задание

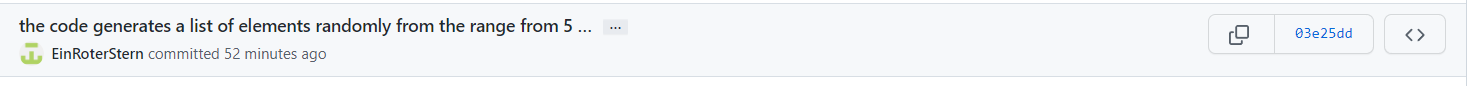
Написать код генерирующий список элементов случайным образом из диапазона от 5 до № по журналу \* 100 (число элементов № по журналу + 10). Выполнить коммит (содержание должно соответствовать задаче).

#4 номер по журналу

import random

for i in range(0,14):

print(random.randint(5,4\*100))



Оформить код в виде функции, вызвав её с указанным числом элементов.

#4 номер по журналу

# -\*- coding: cp1251 -\*- #кодировка

import random

n=int(input("Введите число элементов - "))

def function(x):

for i in range(x):

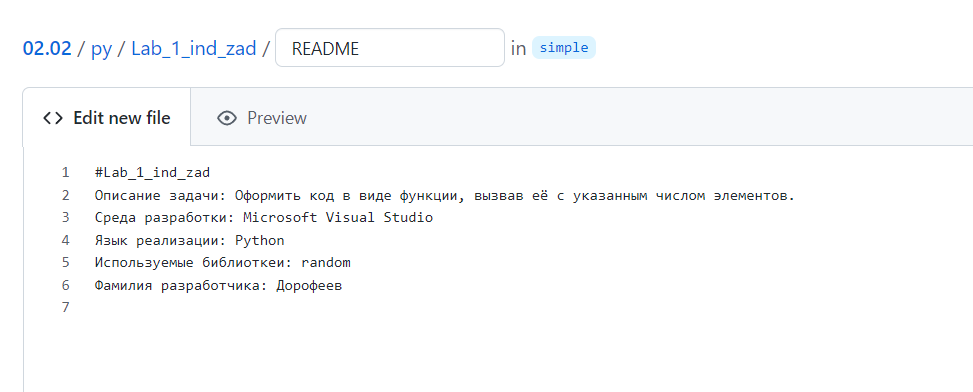
a = print(random.randint(5,4\*100))

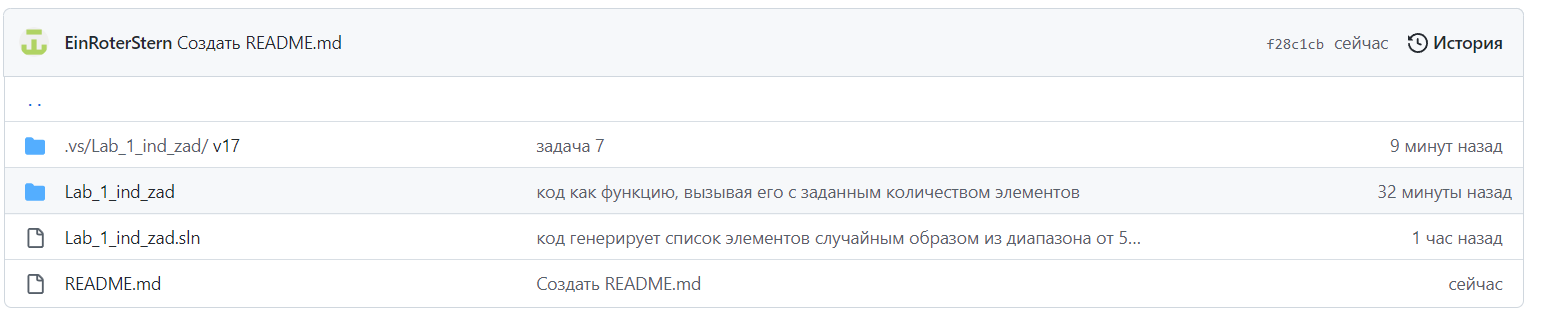
return a

function(n)



Создали файл Readme





**Контрольные вопросы:**

1. Что такое система контроля версий Git? Требуется ли её установка при работе с VS?

Программное обеспечение для облегчения работы с изменяющейся информацией. Установка не требуется, но может быть интегрирована для удобства.

1. Какие основные возможности предоставляет Git в среде VS?

-Создание репозитория

-Создание веток

-Слияние веток

-Фиксация изменений программы

-Импорт на GitHub

1. Что из настроек Git является обязательным при работе с удалённым репозиторием?

Потребуется настроить имя и email пользователя для идентификации автора коммита. В настройке удалённого репозитория указание URL, адреса и имени ветки

1. Какую систему защиты и сертификации данных использует Git по умолчанию?

Git использует систему хэширование SHA-1

1. Можно ли вернуться к прежней версии файла с помощью Git? Каким образом?

Можно. Для этого нужно использовать команду git checkout с указанием идентификатора коммита или ветки.

1. Что нужно сделать, если требуется изменить сообщение последнего коммита?

Нужно поставить галочку amend, но необходимо указать новое сообщение коммита

1. Как называется главная ветвь разработки? Можно ли её переименовать?

Главная ветвь – master . С помощью git-branch можно переименовать, но это может привести к проблемам при работе с другими разработчиками, так как они используют стандартные имена ветвей.

1. Зачем нужен файл .gitignore и каким образом он создаётся?

Файл нужен для указания в нем новых файлов, которые должны быть проигнорированы. Создаётся автоматически