

ГУАП

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

преподаватель

У.С. Опалева

должность, уч. степень, звание

подпись, дата

инициалы, фамилия

ОТЧЕТЫ О ЛАБОРАТОРНЫХ РАБОТАХ

По дисциплине: МДК.02.02 Инструментальные средства разработки
программного обеспечения

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

021к

В.Д. Панков

подпись, дата

инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

| | |
|---------------------------------|---|
| 1 Лабораторная работа № 1 | 2 |
|---------------------------------|---|

1 Лабораторная работа № 1

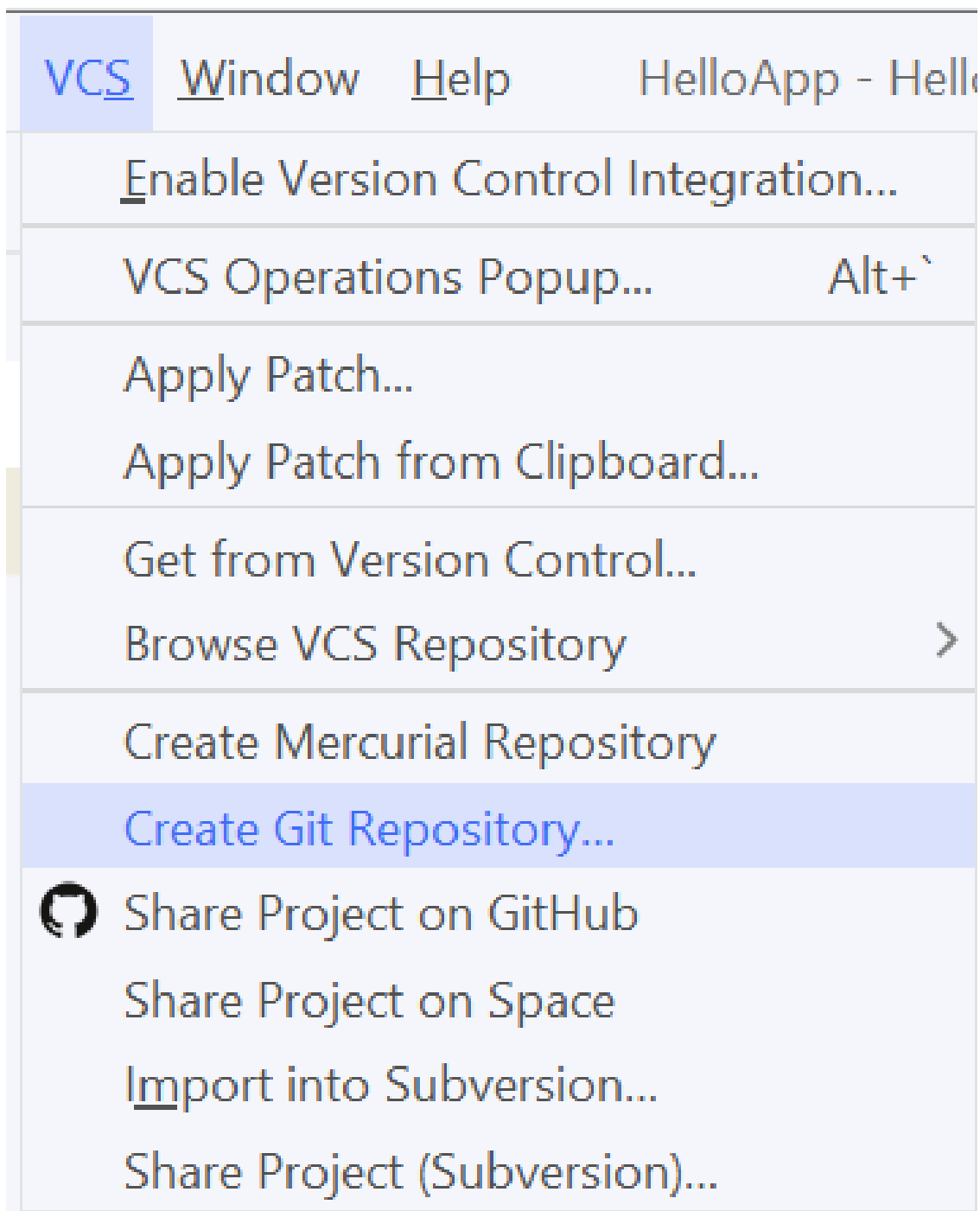
Создание консольного приложения в IDE Visual Studio. Работа с системой контроля версий

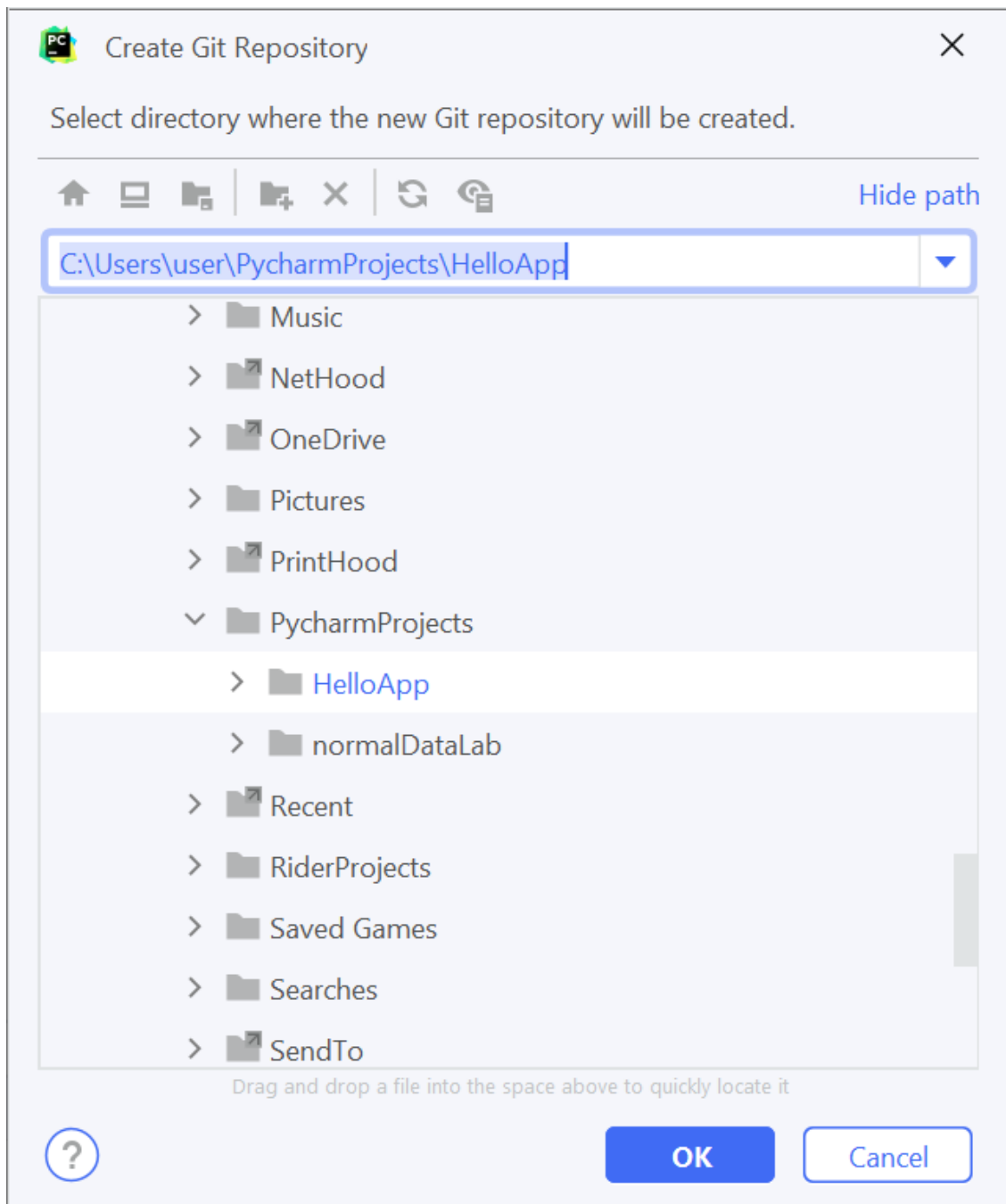
Цель: изучение IDE Visual Studio, создание простейшего приложения на Python, подключение Git.

1. Добавил в HelloApp.py

```
print("Hello Python from Visual Studio!")
```

2. Создал Git-репозиторий с помощью PyCharm





3. Добавил файл в Stage

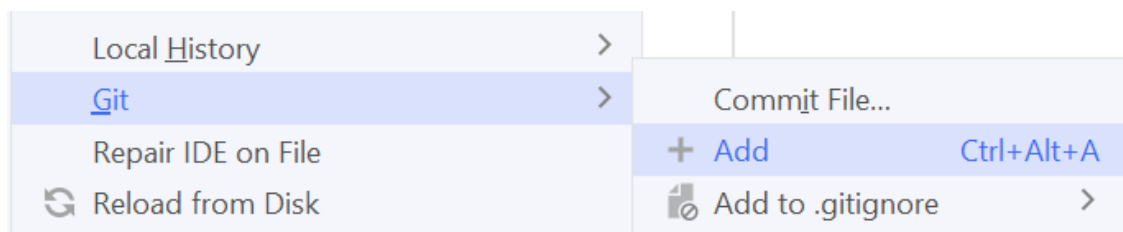


Рисунок 1 – Добавление файла в Stage в Pycharm

4. Создал commit изменений

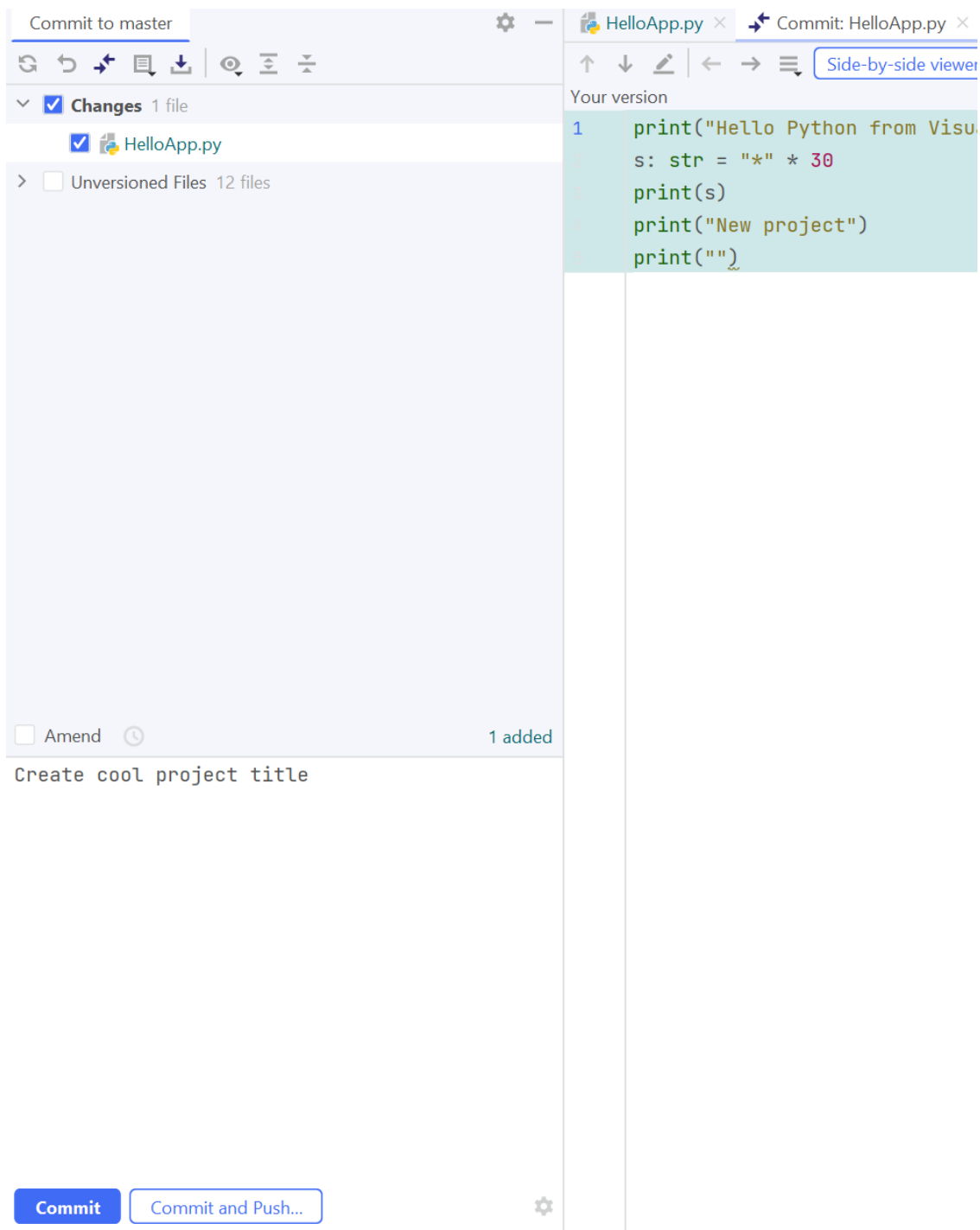


Рисунок 2 – Демонстрация commit

4. Добавление кода из прошлых заданий по Python

```
print("Hello Python from Visual Studio!")
s: str = "*" * 30
print(s)
print("New project")
print("")

import cProfile
import re

r = re.compile("\\d\\S")
cProfile.run("""[r.findall("sdfdsfD, 1d, 7f") for i in range(1000000)]""")
```

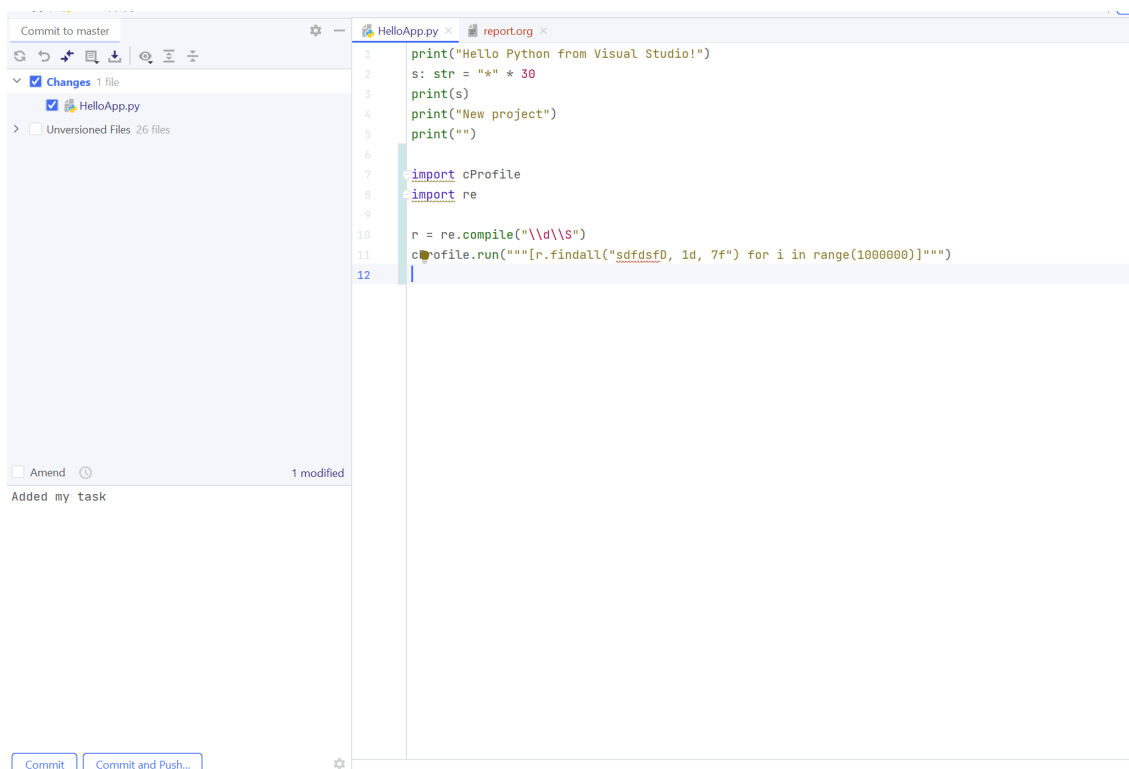


Рисунок 3 – Создание нового commit

5. Демонстрация git-log

| | | | |
|---------------------------|---------------|---------|--------------------|
| Added my task | origin/master | pank-su | 2/22/2023 10:30 AM |
| Create cool project title | | pank-su | 2/22/2023 9:47 AM |


6. Push репозитория на GitHub

Create a new repository


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

Owner *

Repository name *


 suai-materials ▾

/


HelloApp 

Great repository names are short and memorable. Need inspiration? How about [miniature-fishstick?](#)

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.

Рисунок 4 – Создание репозитория на GitHub

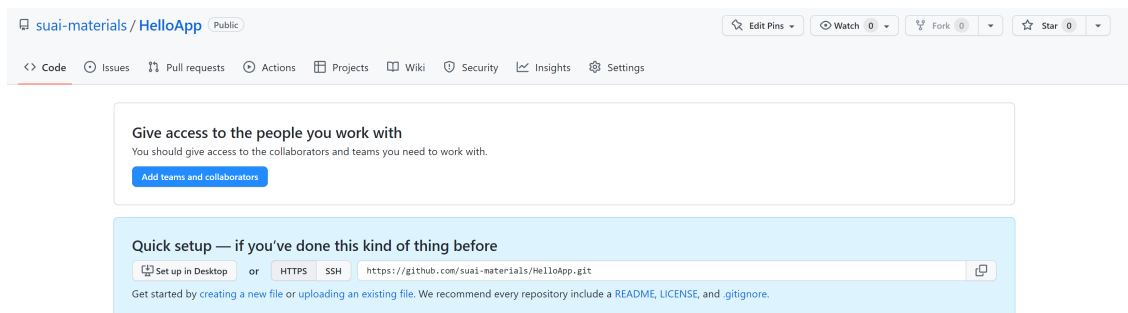


Рисунок 5 – Получил ссылку

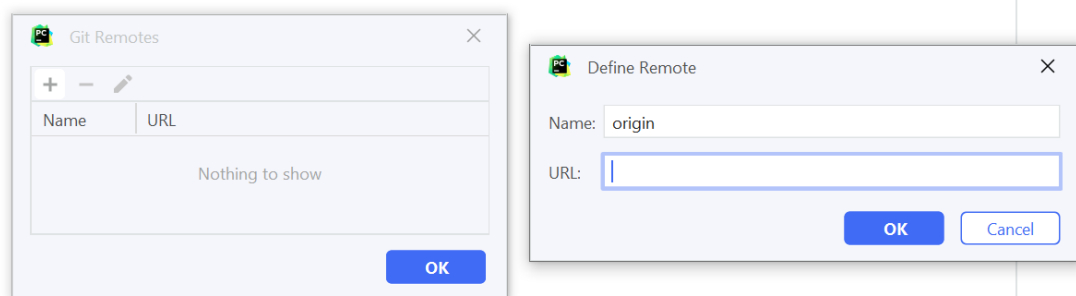


Рисунок 6 – Добавление remote репозитория

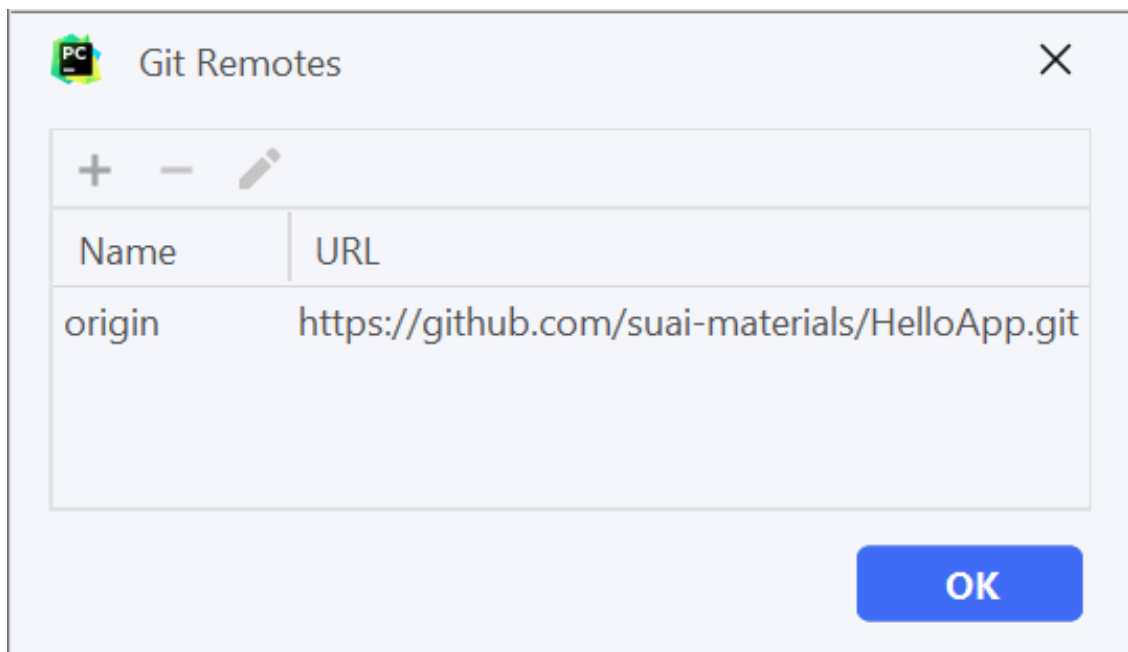


Рисунок 7 – После добавления remote

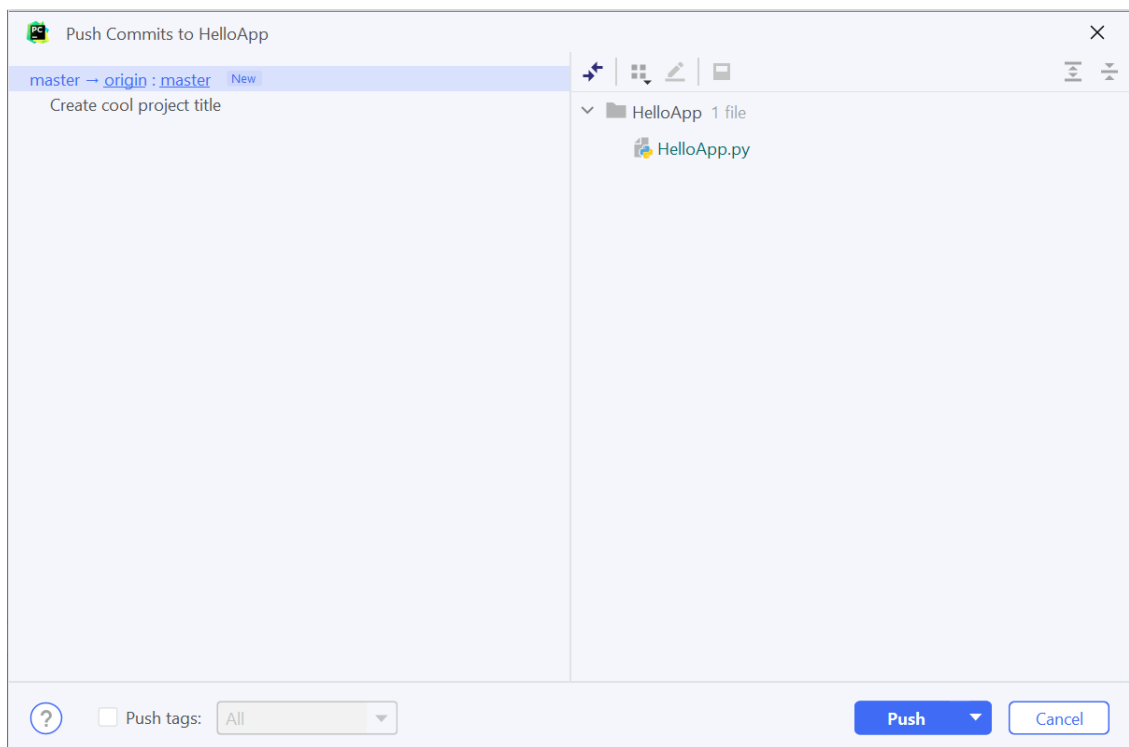


Рисунок 8 – Процесс push

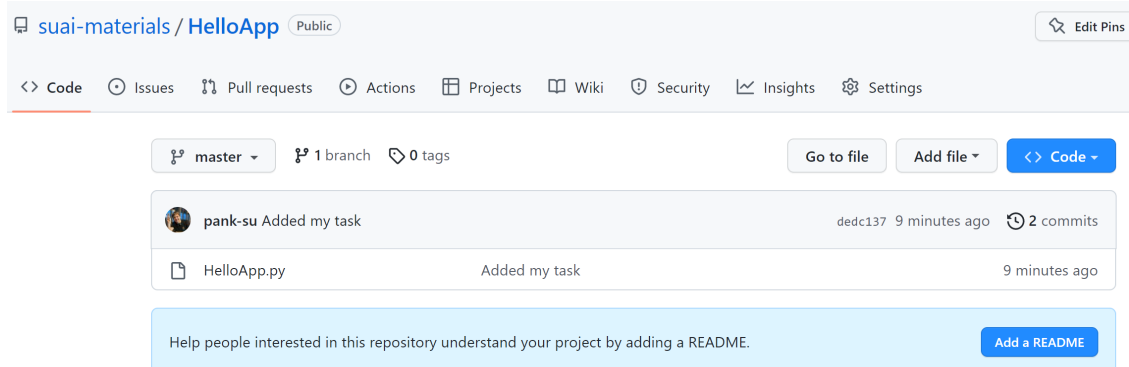


Рисунок 9 – Изменения репозитория

7. Так-как я использую Pycharm, то он не генерирует .gitignore, поэтому самостоятельно был создан .gitignore, который у меня исключает папки:

- .idea - файлы JetBrains, которые описывают проект
- venv - локальный интерпретатор

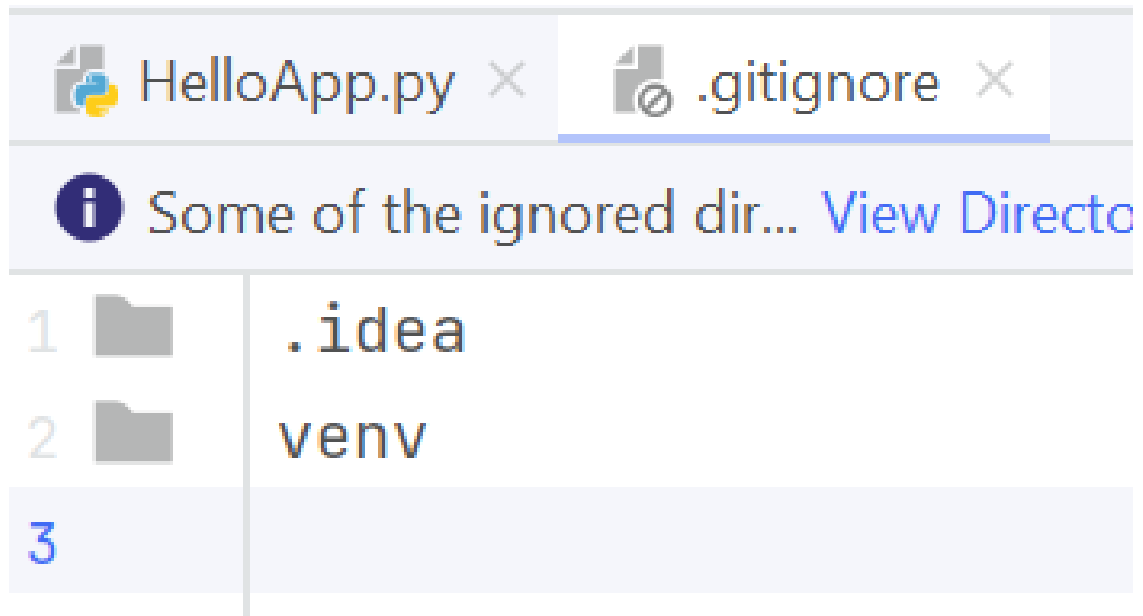


Рисунок 10 – Демонстрация .gitignore

8. Создать новый проект в VS. 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). Оформить код в виде функции, вызвав её с указанным числом элементов. Добавить коммит. Запустить на GitHub. На веб-сервисе создать файл README с описанием задачи, перечнем, включающим среду и язык реализации, используемые библиотеки, фамилию разработчика.

Код генератора списка:

```
from random import randrange

print([randrange(5, 12 * 100) for i in range(22)])
```

Код генератора списка в функции:

```
from random import randrange
from typing import List

def my_random_list(n: int) -> List[int]:
    """Генерация списка со случайными числами"""
    return [randrange(5, 12 * 100) for i in range(n)]

if __name__ == '__main__':
    print(my_random_list(22))
```

Текст README.org файла:

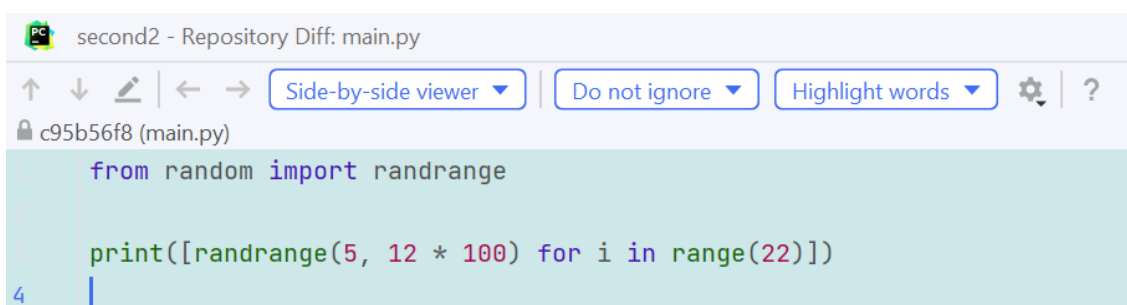


Рисунок 11 – Первый коммит

=

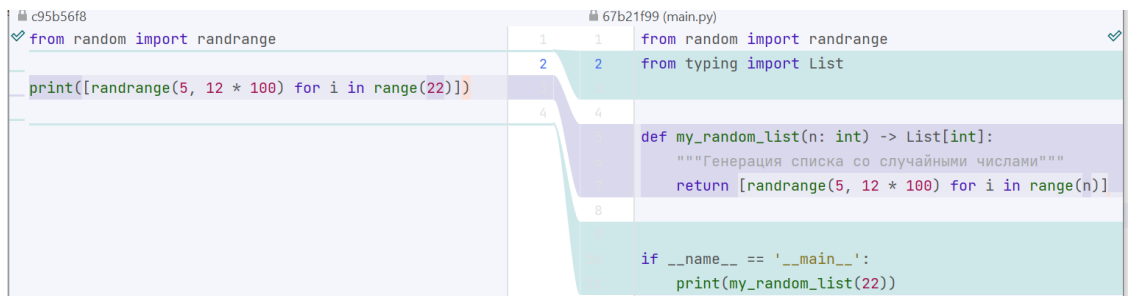


Рисунок 12 – Второй коммит

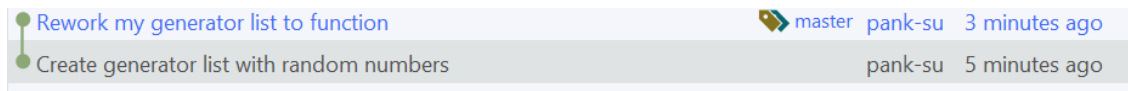


Рисунок 13 – Git-log

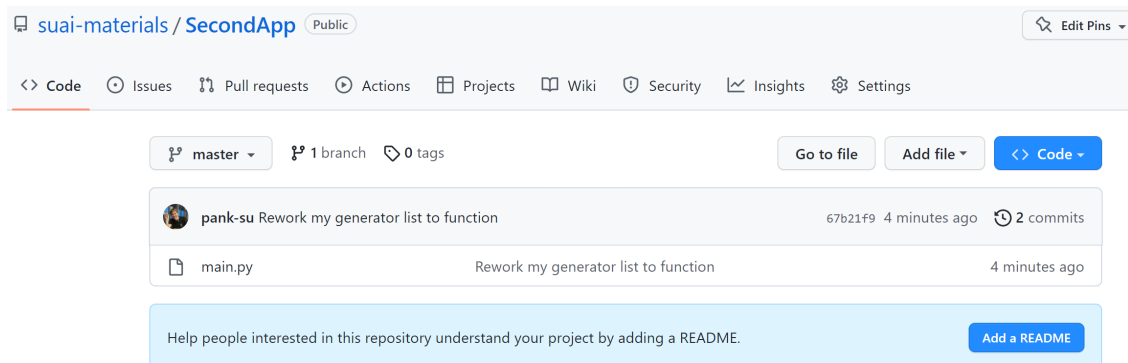


Рисунок 14 – Репозиторий на GitHub после двух commit'ов

<> Edit new file

Preview

```

1  #+TITLE: Генератор списка случайных чисел
2  #+AUTHOR: Панков Василий
3  #+OPTIONS: toc:nil
4
5  Реализовал функцию, которая генерирует список размером n,
6  элементы которого случайные числа от 5 до 1200.
7
8  - Среда: JetBrains PyCharm
9  - Язык: Python
10 - Используемые библиотеки
11   - random - для генерации случайных чисел
12   - typing - для статической типизации
13 - Автор: Панков В. Д.
14

```

Рисунок 15 – Создание README файла

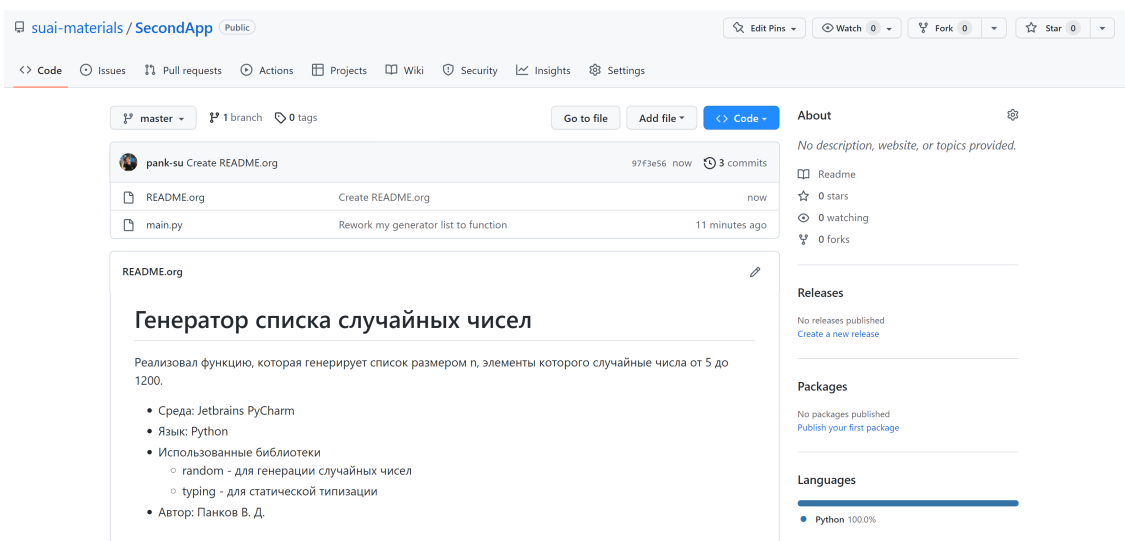


Рисунок 16 – Итоговый репозиторий на GitHub

Контрольные вопросы:

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

Git - это система контроля версий, которая позволяет отслеживать изменения в исходном коде проекта и управлять ими. Git позволяет хранить историю изменений, возвращаться к предыдущим версиям кода, вносить изменения параллельно, объединять изменения и многое другое.

Для Visual Studio требуется предустановленный Git.

2. Какие основные возможности предоставляет Git в среде VS?

В среде Visual Studio Git предоставляет возможности по созданию, клонированию, управлению и синхронизации репозитория Git. Основные возможности Git в среде Visual Studio:

- Создание новых репозитория Git
- Клонирование существующих репозитория Git
- Управление изменениями исходного кода в репозитории Git
- Отслеживание изменений, внесенных другими участниками проекта
- Работа с ветками и слияние изменений
- Отправка изменений в удаленный репозиторий и получение изменений из удаленного репозитория.

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

- Указание удаленного репозитория, куда будут отправляться изменения.
- Настройка локальной ветки для отслеживания удаленной ветки.
- Аутентификация на удаленном репозитории.

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

Git может использовать три различных протокола для передачи данных: Local, HTTP, Secure Shell (SSH).

При хранении файлов на компьютере используется Local.

Для передачи используют: HTTP или Smart HTTP(более умная версия HTTP).

SSH используется, если он используется сервером.

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

Да, можно вернуться к прежней версии файла с помощью Git. Для этого необходимо использовать команду "git checkout" с указанием хэша коммита или имени ветки, на которую нужно переключиться.

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

Если требуется изменить сообщение последнего коммита, можно использовать команду "git commit --amend". Она позволяет изменить сообщение последнего коммита или добавить изменения в него.

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

Главная ветвь разработки называется "master". В Git версии 2.28.0 и выше главная ветвь была переименована в "main". Да, её можно переименовать с помощью команды "git branch -m <old_{branchname}> <new_{branchname}>".

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

Файл .gitignore предназначен для указания Git файлов и папок, которые не должны быть добавлены в репозиторий. Он позволяет исключить файлы и папки, которые не нужны в репозитории, такие как временные файлы, конфигурационные файлы, файлы логов, файлы бинарных данных и многое другое.

Файл .gitignore создается в корневой папке проекта. Он может содержать шаблоны для исключения файлов и папок. Шаблоны могут включать имя файла или папки, а также использовать символы подстановки, такие как *, ?, [], { }, и многое другое. Файл .gitignore можно создать вручную или с помощью специальных инструментов, таких как Visual Studio или Git Extensions.