

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 №9
дисциплины «Основы программной инженерии»

Выполнил:
Звездин Алексей Сергеевич
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:
Воронкин Р. А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Ход работы

1. Я изучил теоретический материал работы

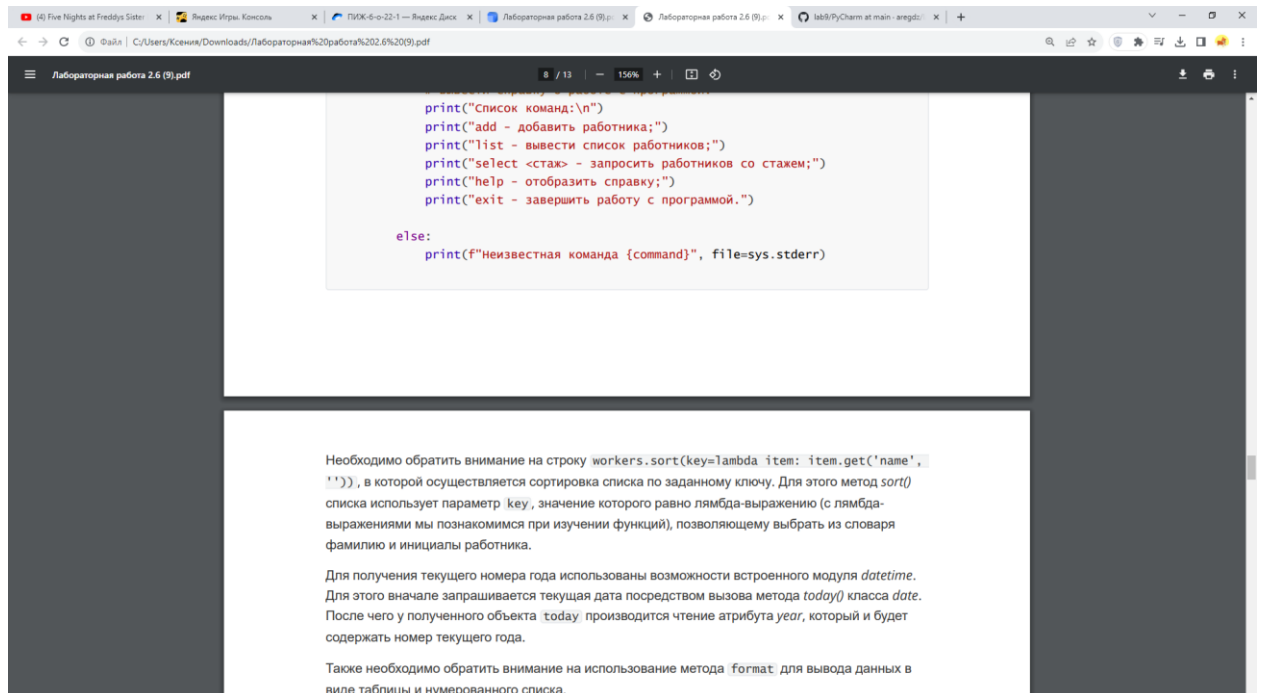


Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создал 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 (*).

Owner *



InternetHacker1123

Repository name *

/ Laba2(6)_9



Your new repository will be created as Laba2-6_9.

The repository name can only contain ASCII letters, digits, and the characters -, ., and _.

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

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.

Create repository

Рисунок 2.1 – Настройка репозитория

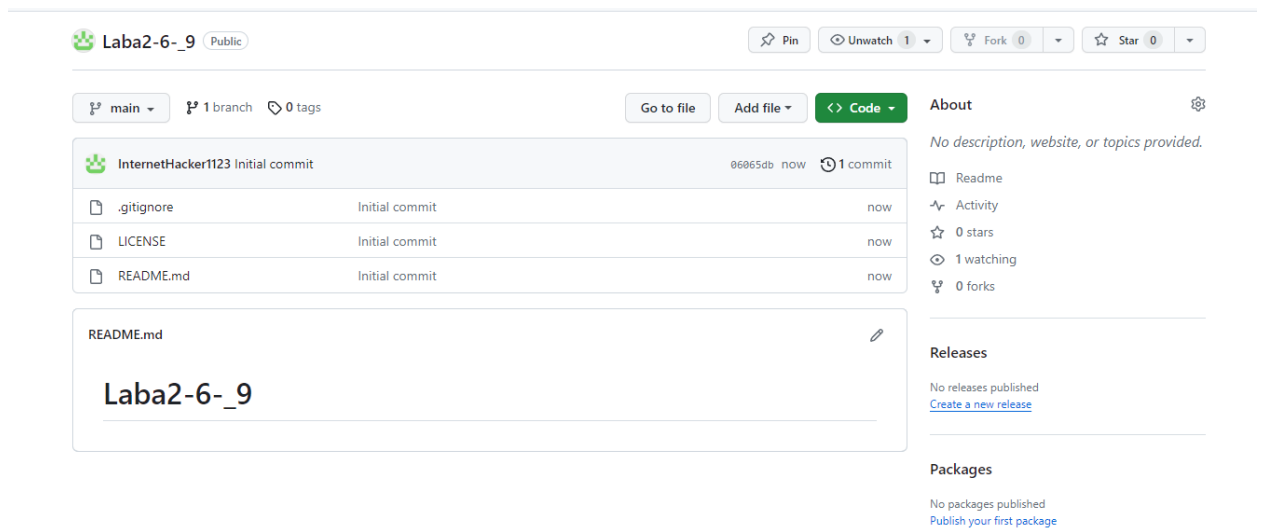


Рисунок 2.2 – Готовый репозиторий

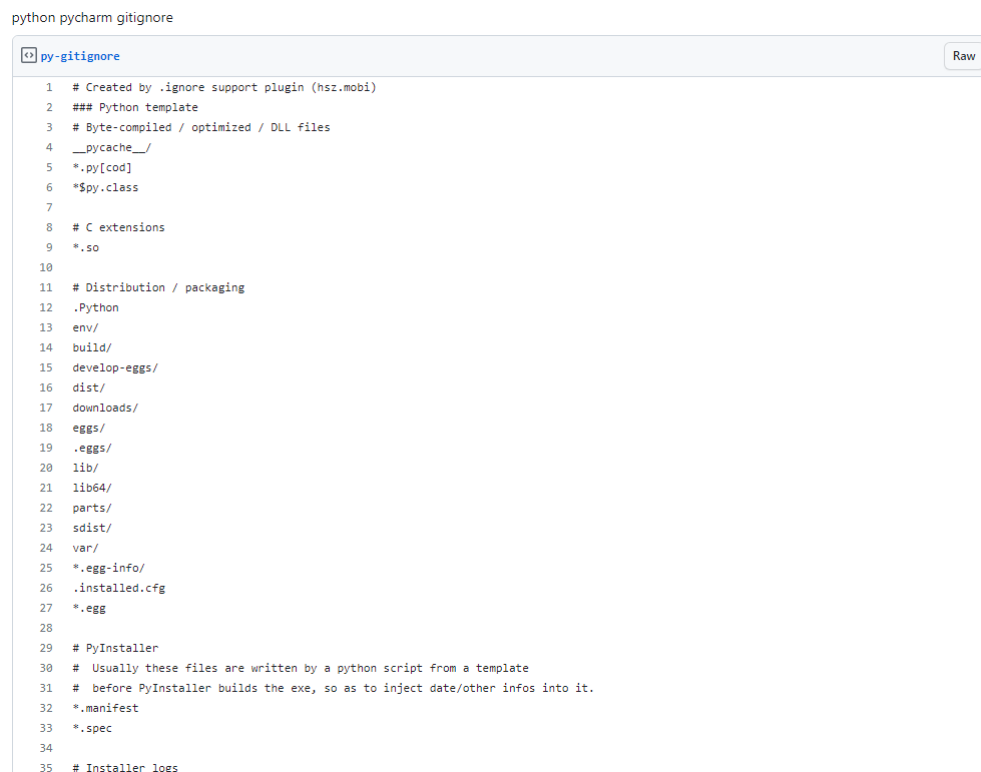
3. Выполняю клонирование созданного репозитория

```
C:\Users\tyt\Desktop\SE\laba9>git clone https://github.com/InternetHacker1123/Laba2-6-_9.git
Cloning into 'Laba2-6-_9'...
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\tyt\Desktop\SE\laba9>
```

Рисунок 3.1 – Клонирование репозитория на локальный диск

4. Дополнил файл .gitignore необходимыми правилами для работы с VS Code

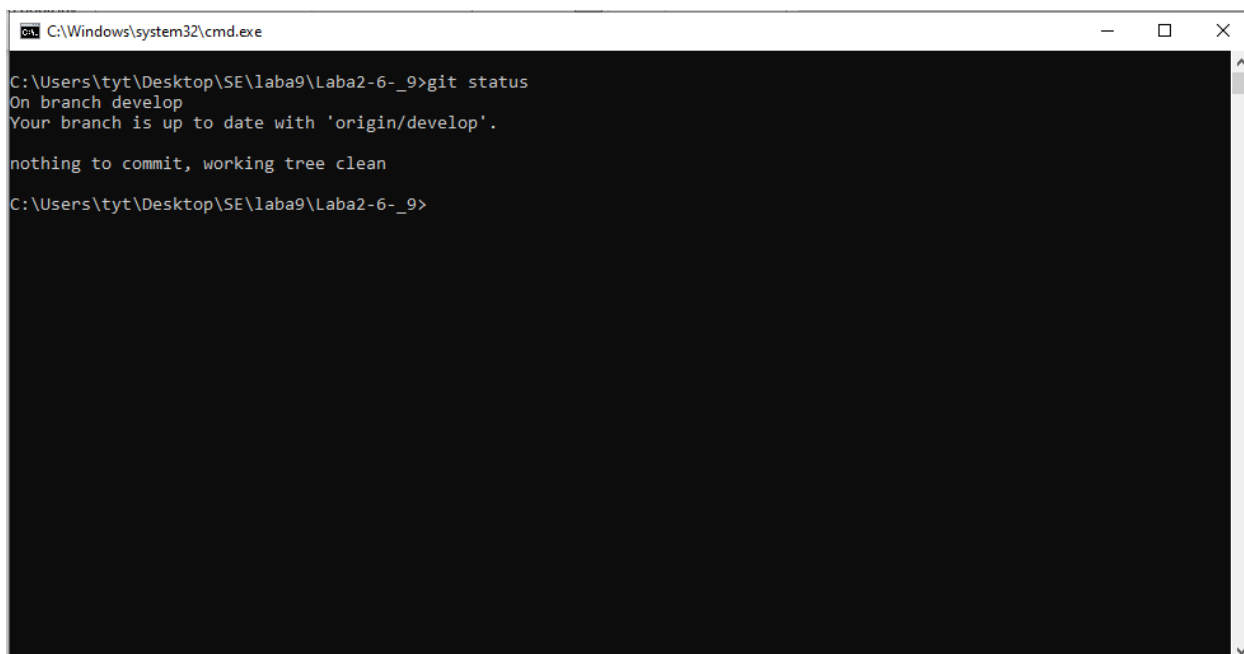
python pycharm gitignore



```
1 # Created by .ignore support plugin (hsz.mobi)
2 ### Python template
3 # Byte-compiled / optimized / DLL files
4 __pycache__/
5 *.py[cod]
6 *$py.class
7
8 # C extensions
9 *.so
10
11 # Distribution / packaging
12 .Python
13 env/
14 build/
15 develop-eggs/
16 dist/
17 downloads/
18 eggs/
19 .eggs/
20 lib/
21 lib64/
22 parts/
23 sdist/
24 var/
25 *.egg-info/
26 .installed.cfg
27 *.egg
28
29 # PyInstaller
30 # Usually these files are written by a python script from a template
31 # before PyInstaller builds the exe, so as to inject date/other infos into it.
32 *.manifest
33 *.spec
34
35 # Installer logs
```

Рисунок 4.1 – .gitignore для VS Code

5. Организовал свой репозиторий в соответствии с моделью ветвления git-flow



```
C:\Windows\system32\cmd.exe
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9>git status
On branch develop
Your branch is up to date with 'origin/develop'.

nothing to commit, working tree clean
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9>
```

Рисунок 5.1 – Создание ветки develop от ветки main

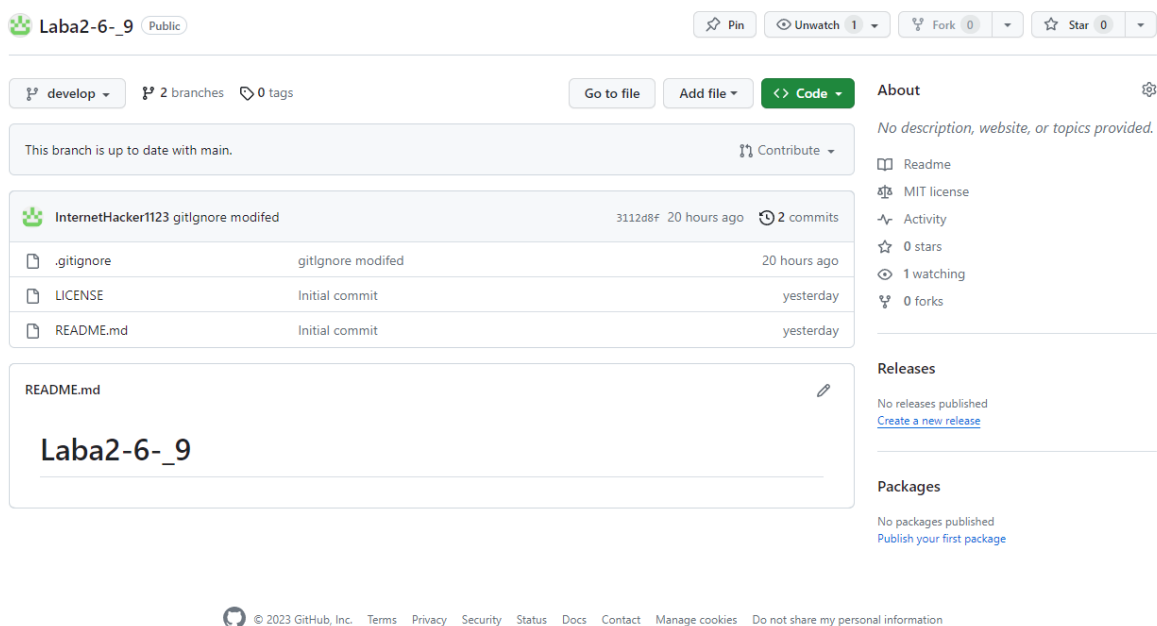


Рисунок 5.2 – Ветка develop на GitHub

6. Создал проект PyCharm в папке репозитория

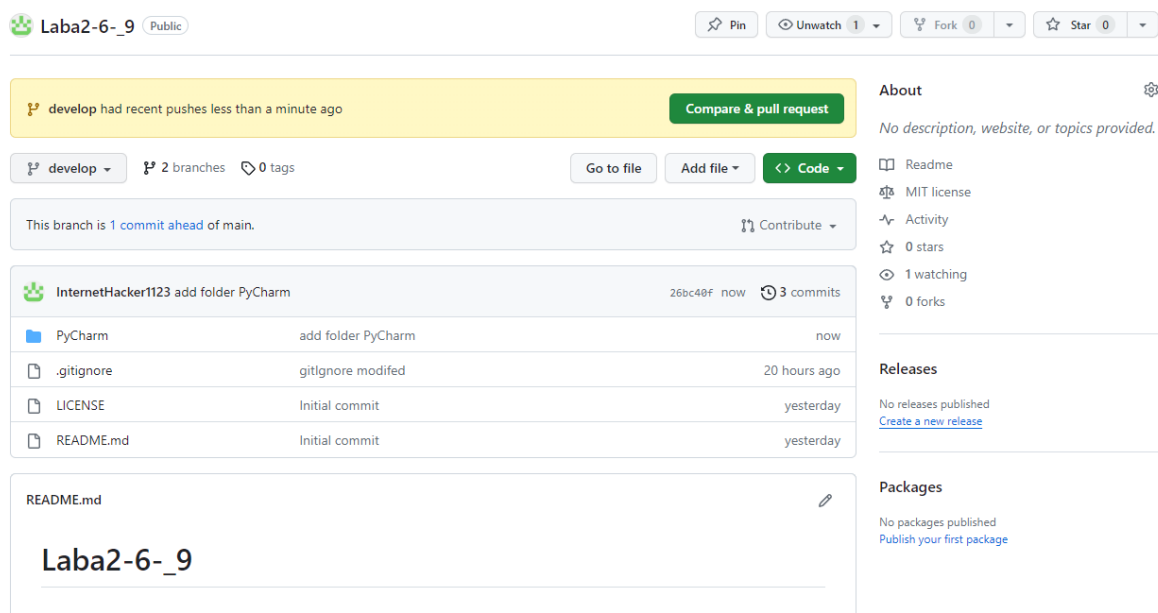


Рисунок 6.1 – Репозиторий с проектом PyCharm

7. Проработал примеры лабораторной работы. Создала для каждого примера отдельный модуль языка Python. Зафиксировал изменения в репозитории.

```

PyCharm > example1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import sys
4  from datetime import date
5  if __name__ == '__main__':
6      # Список работников.
7      workers = []
8      # Организовать бесконечный цикл запроса команд.
9      while True:
10         # Запросить команду из терминала.
11         command = input(">>> ").lower()
12         # Выполнить действие в соответствие с командой.
13         if command == 'exit':
14             break
15         elif command == 'add':
16             # Запросить данные о работнике.
17             name = input("Фамилия и инициалы? ")
18             post = input("Должность? ")
19             year = int(input("Год поступления? "))
20             # Создать словарь.
21             worker = {
22                 'name': name,
23                 'post': post,
24                 'year': year,
25             }
26             # Добавить словарь в список.
27             workers.append(worker)
28             # Отсортировать список в случае необходимости.
29             if len(workers) > 1:
30                 workers.sort(key=lambda item: item.get('name', ''))
31         elif command == 'list':
32             # Заголовок таблицы.
33             line = '+-{}-+-{}-+-{}-+-{}-+'.format(
34                 '-' * 4,
35                 '-' * 30,
36                 '-' * 20,
37                 '-' * 8
38             )
39             print(line)
40             print(
41                 '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
42                     "№",
43                     "Ф.И.О.",
44                     "Должность",
45                     "Год"
46                 )
47             )

```

Рисунок 7.1 – Проработка примера 1

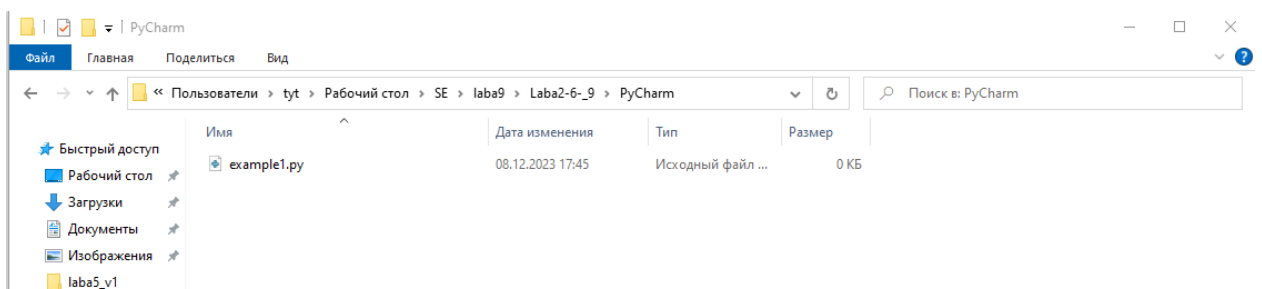


Рисунок 7.4 – Создание отдельного модуля для примера

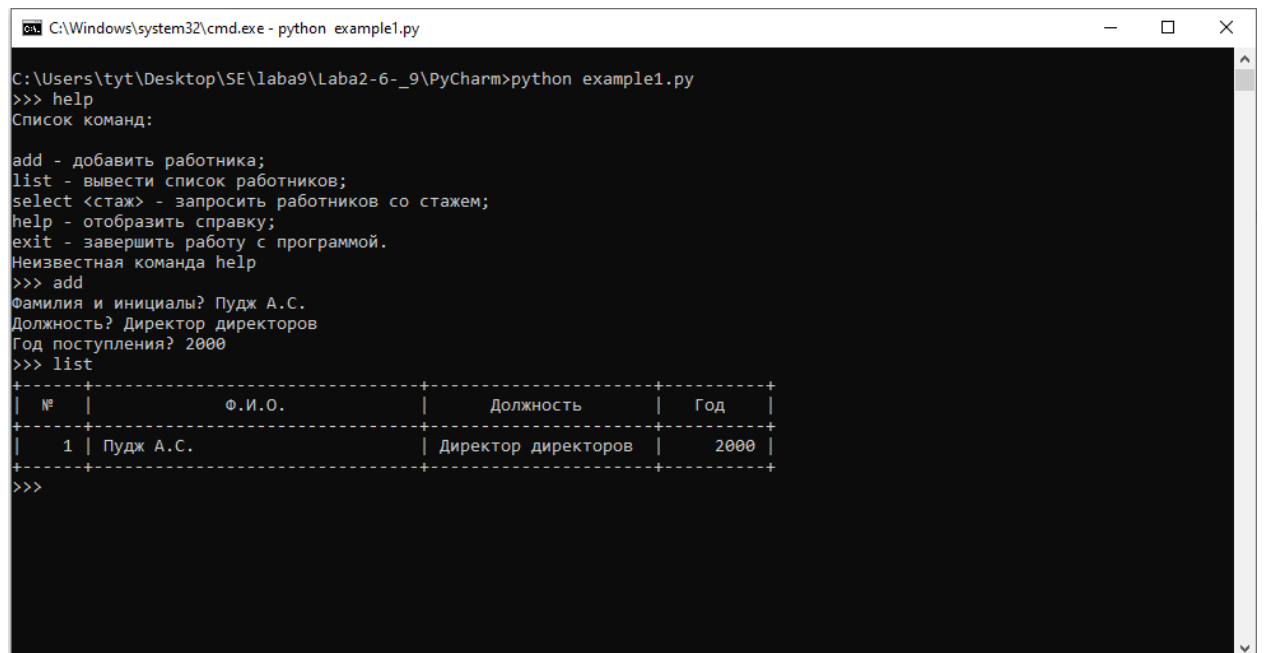
```

C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9\PyCharm>git add .
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9\PyCharm>git commit -m"add example again"
[develop 9f6eaca] add example again
1 file changed, 87 insertions(+)
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9\PyCharm>

```

Рисунок 7.5 – Фиксирование изменений в репозитории

8. Привел в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных, вводимых с клавиатуры.



```

C:\Windows\system32\cmd.exe - python example1.py
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9\PyCharm>python example1.py
>>> help
Список команд:
add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
Неизвестная команда help
>>> add
Фамилия и инициалы? Пудж А.С.
Должность? Директор директоров
Год поступления? 2000
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Пудж А.С. | Директор директоров | 2000 |
+-----+-----+-----+-----+
>>>

```

Рисунок 8.1 – Результат примера 1

9. Решил 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.


```
PyCharm > zadacha.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      school = {
6          '1а': 11,
7          '1б': 12,
8          '2б': 13,
9          '6а': 14,
10         '7в': 15,
11         '8в': 16,
12         '9в': 17,
13         '10в': 18,
14         '11в': 19,
15     }
16
17     # Меняем кол-во учащихся
18     school['11в'] = 1
19
20     # Добавляем новый класс
21     school['5а'] = 23
22
23     # Удаляем один класс
24     del school['10в']
25
26     # Сумма учеников
27     vse_studenty = sum(school.values())
28
29     print(f"Общее количество учащихся в школе: {vse_studenty}")
30
```

Рисунок 9.1 – Код программы zadacha.py

10. Зафиксировал сделанные изменения в репозитории

```
C:\TestGit\TheZenOfPython>git commit -m"adding Task9.py"
[develop eb8b8b0] adding Task9.py
 2 files changed, 32 insertions(+)
 create mode 100644 PyCharm/Task/Task11.py
 create mode 100644 PyCharm/Task/Task9.py

C:\TestGit\TheZenOfPython>git status
On branch develop
Your branch is ahead of 'origin/develop' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Рисунок 10.1 – Коммит файлов в репозитории git

11. Решил 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.

```
PyCharm > zadacha2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      dictionary = {
6          1: 'Я',
7          2: 'Люблю',
8          3: 'Играть',
9          4: 'В',
10         5: 'Доту'
11     }
12
13     # Создаем словарь с ключами наоборот
14     result = {value: key for key, value in dictionary.items()}
15
16     # вывод результата
17     print(f"Обратный словарь: {result}")
```

Рисунок 11.1 – Код программы Task11.py

12. Зафиксировал сделанные изменения в репозитории

```
C:\Windows\system32\cmd.exe
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9\PyCharm>git add .
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9\PyCharm>git commit -m"task complete"
[develop 71347c1] task complete
2 files changed, 32 insertions(+)
create mode 100644 PyCharm/zadacha.py
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9\PyCharm>git add .
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9\PyCharm>git commit -m"add task2 complete"
[develop bbb3f52] add task2 complete
1 file changed, 17 insertions(+)
create mode 100644 PyCharm/zadacha2.py
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-9\PyCharm>
```

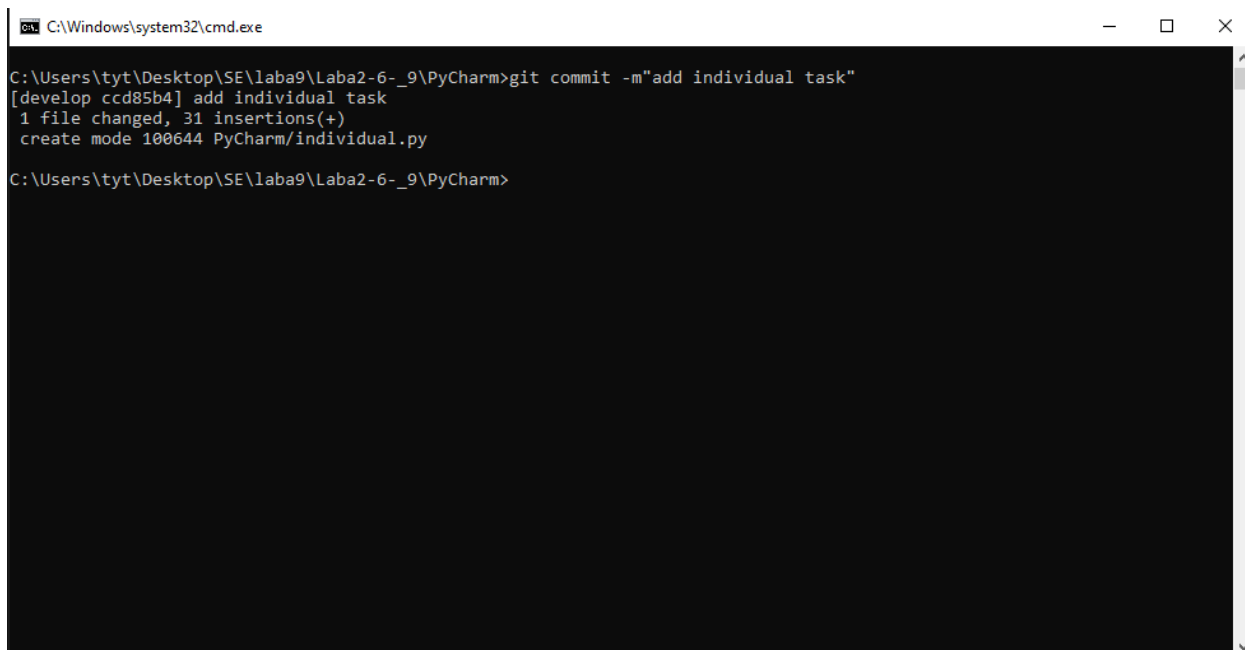
Рисунок 12.1 – Коммит файлов в репозитории git

13. Привел в отчете скриншоты работы программ решения индивидуального задания.

```
PyCharm > individual.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      sp = []
6
7      while True:
8          inp = input(">>> ").lower()
9
10         if inp == 'add':
11             punkt_naznachenia = input("Пункт назначения поезда: ")
12             train_number = input("Номер поезда: ")
13             time_otpravlenia = input("Время отправления: ")
14
15             dictionary = {
16                 'Пункт назначения ': punkt_naznachenia,
17                 'Номер поезда: ': train_number,
18                 'Время отправления:': time_otpravlenia
19             }
20
21             sp.append(dictionary)
22             sp = sorted(sp, key=lambda x: x['Номер поезда: '])
23
24
25         if inp.isdigit():
26             for d in sp:
27                 if inp in d.values():
28                     print(d)
29                 else:
30                     print('Поезда с таким номером нет')
31
32
```

Рисунок 13.1 – Код программы individual.py

14. Зафиксировала сделанные изменения в репозитории.



```
C:\Windows\system32\cmd.exe
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-_9\PyCharm>git commit -m"add individual task"
[develop ccd85b4] add individual task
1 file changed, 31 insertions(+)
create mode 100644 PyCharm/individual.py
C:\Users\tyt\Desktop\SE\laba9\Laba2-6-_9\PyCharm>
```

Рисунок 14.1 – Коммит файлов в репозитории git

Контрольные вопросы

1. Что такое словари в языке Python?

В языке программирования Python словари (тип `dict`) представляют собой еще одну разновидность структур данных наряду со списками и кортежами. Словарь — это изменяемый (как список) неупорядоченный (в отличие от строк, списков и кортежей) набор элементов "ключ: значение".

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

Да, функция `len()` может быть использована при работе со словарями в Python. Она возвращает количество элементов в словаре, то есть количество пар «ключ-значение».

3. Какие методы обхода словарей Вам известны?

Цикл `for` по ключам, использование метода `items()`, который возвращает пары ключ-значение

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

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

```
for value in my_dict.values():  
    print(value)
```

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

```
my_dict = {}  
my_dict['ключ'] = 'значение'
```

6. Что такое словарь включений?

Словарь включение аналогичен списковым включениям, за исключением того, что он создаёт объект словаря вместо списка.

7. Самостоятельно изучите возможности функции `zip()` приведите примеры ее использования.

Функция `zip()` в Python создает итератор, который объединяет элементы из нескольких источников данных. Эта функция работает со списками, кортежами, множествами и словарями для создания списков или кортежей, включающих все эти данные.

Предположим, что есть список имен и номером сотрудников, и их нужно объединить в массив кортежей. Для этого можно использовать функцию `zip()`.

```
employee_numbers = [2, 9, 18, 28]  
employee_names = ["Дима", "Марина", "Андрей", "Никита"]
```

```
zipped_values = zip(employee_names, employee_numbers)  
zipped_list = list(zipped_values)
```

```
print(zipped_list)
```

Функция `zip` возвращает следующее:

```
[('Дима', 2), ('Марина', 9), ('Андрей', 18), ('Никита', 28)]
```

8. Самостоятельно изучите возможности модуля `datetime`. Каким функционалом по работе с датой и временем обладает этот модуль?

Datetime — важный элемент любой программы, написанной на Python. Этот модуль позволяет управлять датами и временем, представляя их в таком виде, в котором пользователи смогут их понимать.

datetime включает различные компоненты. Так, он состоит из объектов следующих типов:

- `date` — хранит дату;
- `time` — хранит время;
- `datetime` — хранит дату и время.