

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»**

Выполнила:  
Панюкова Ксения Юрьевна  
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 изучила теоретический материал работы

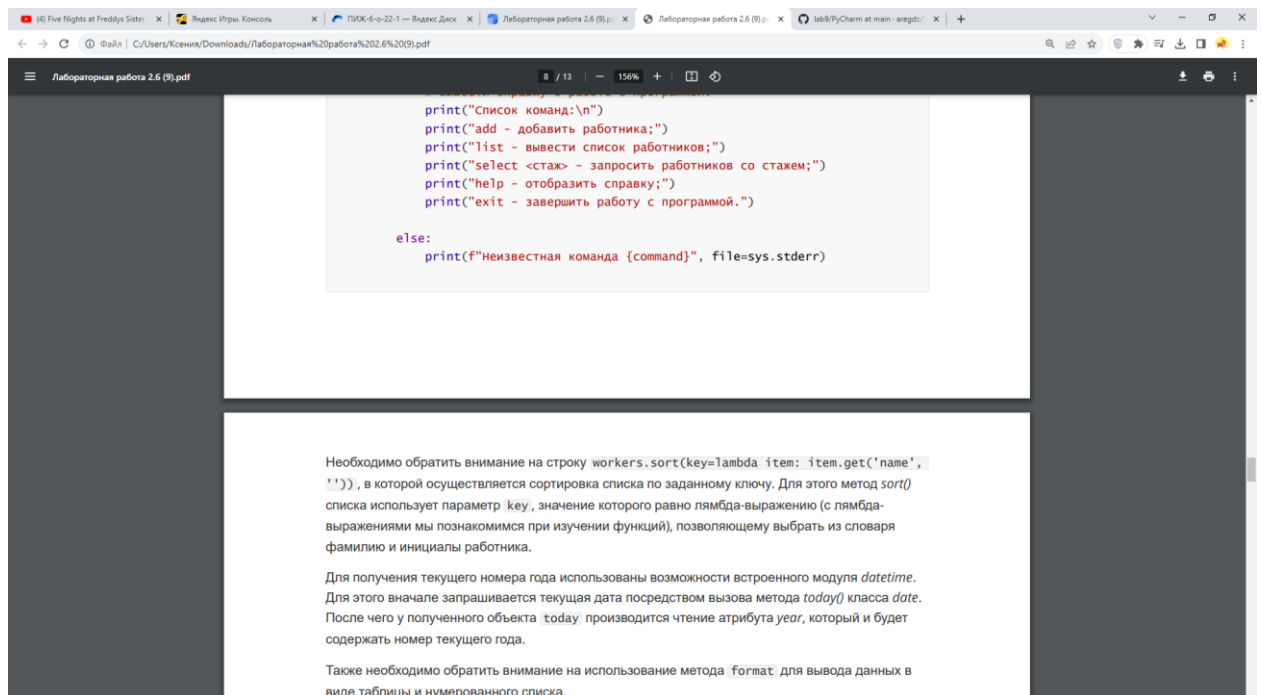


Рисунок 1.1 – Изучение материала для лабораторной работы

## 2. Создала общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python

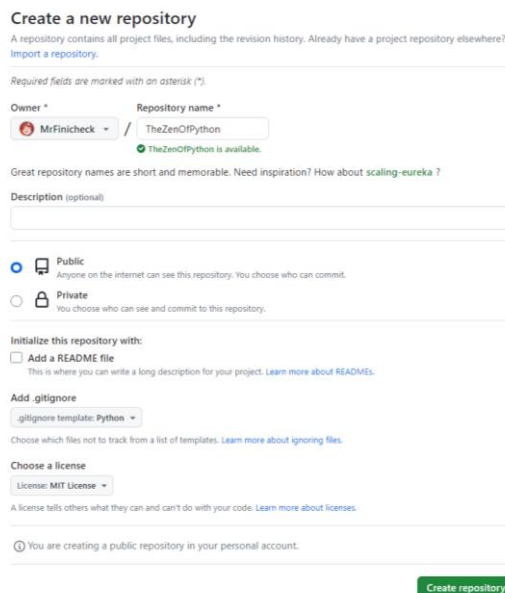


Рисунок 2.1 – Настройка репозитория

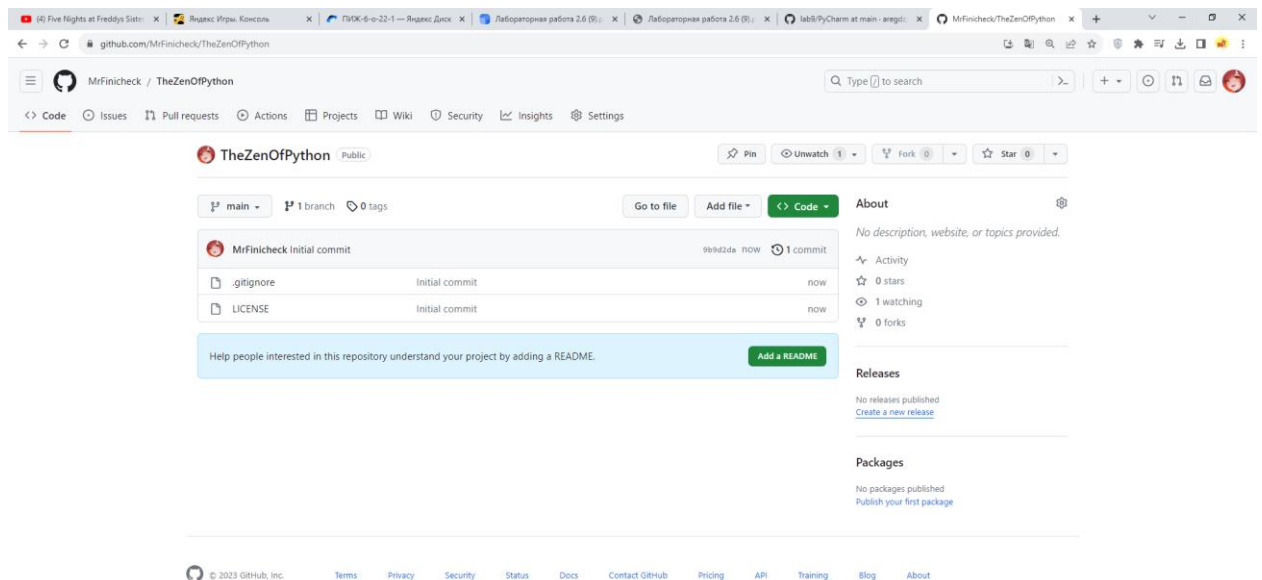


Рисунок 2.2 – Готовый репозиторий

### 3. Выполняю клонирование созданного репозитория

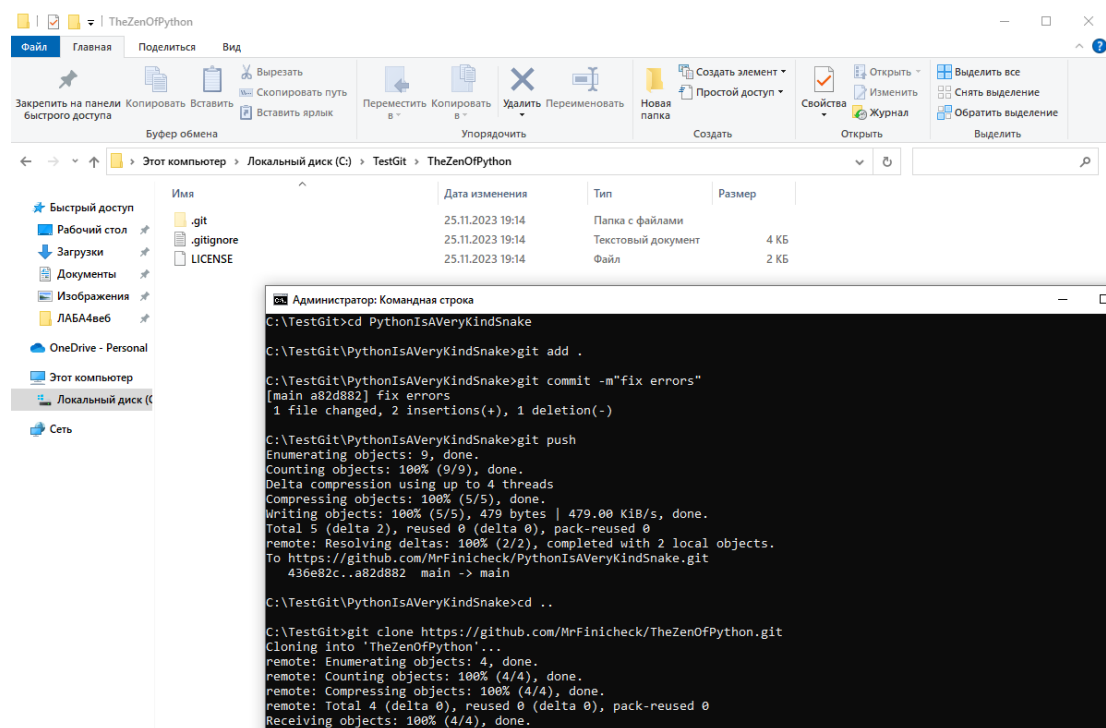
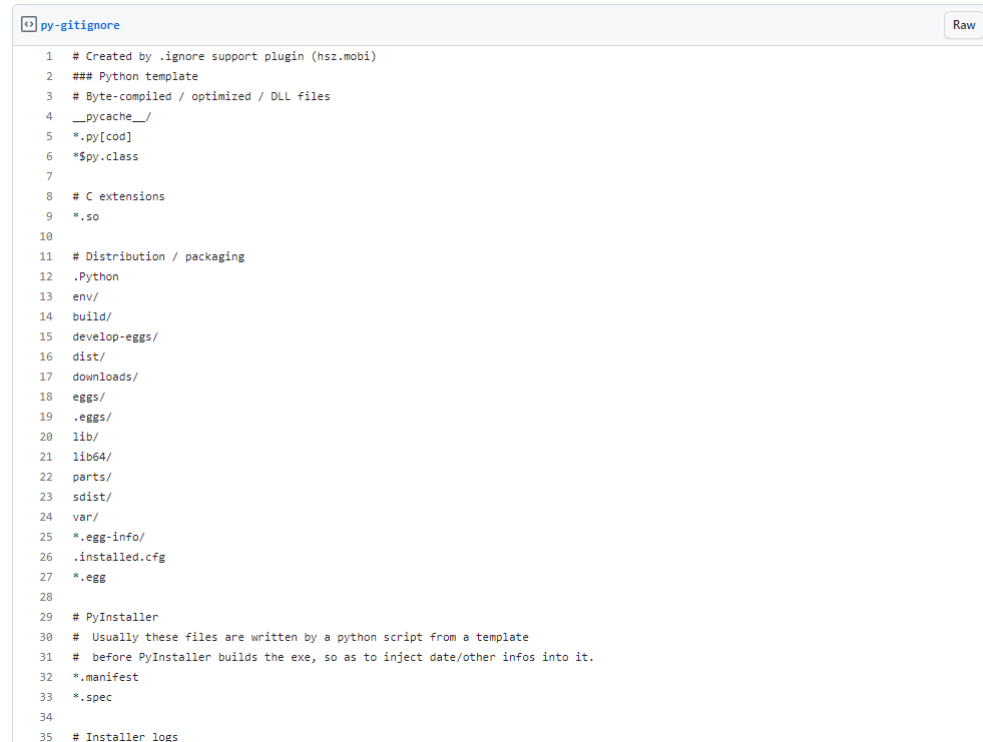


Рисунок 3.1 – Клонирование репозитория на локальный диск

### 4. Дополнила файл .gitignore необходимыми правилами для работы с IDE PyCharm

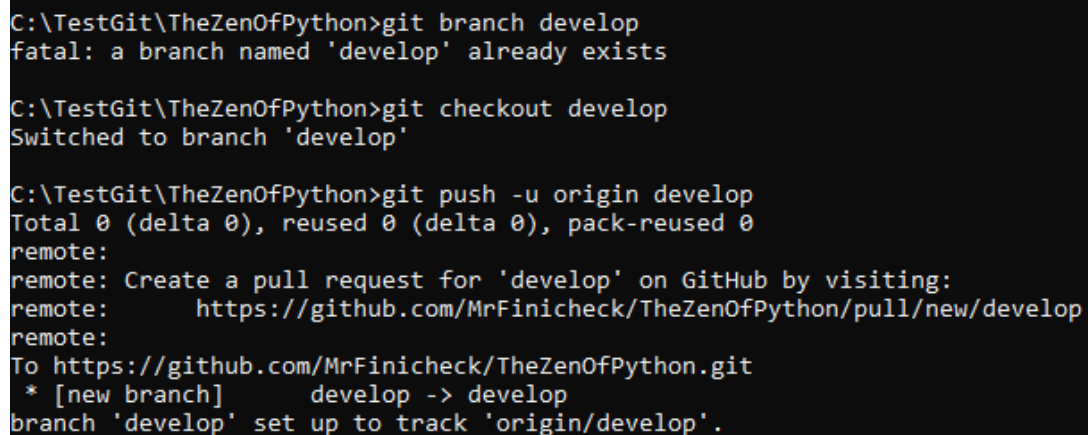
python pycharm gitignore



```
1 # Created by .ignore support plugin (hsz.mobi)
2 ### Python template
3 # Byte-compiled / optimized / DLL files
4 __pycache__/
5 *.py[co]
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 IDE PyCharm

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



```
C:\TestGit\TheZenOfPython>git branch develop
fatal: a branch named 'develop' already exists

C:\TestGit\TheZenOfPython>git checkout develop
Switched to branch 'develop'

C:\TestGit\TheZenOfPython>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/MrFinicheck/TheZenOfPython/pull/new/develop
remote:
To https://github.com/MrFinicheck/TheZenOfPython.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.
```

Рисунок 5.1 – Создание ветки develop от ветки main

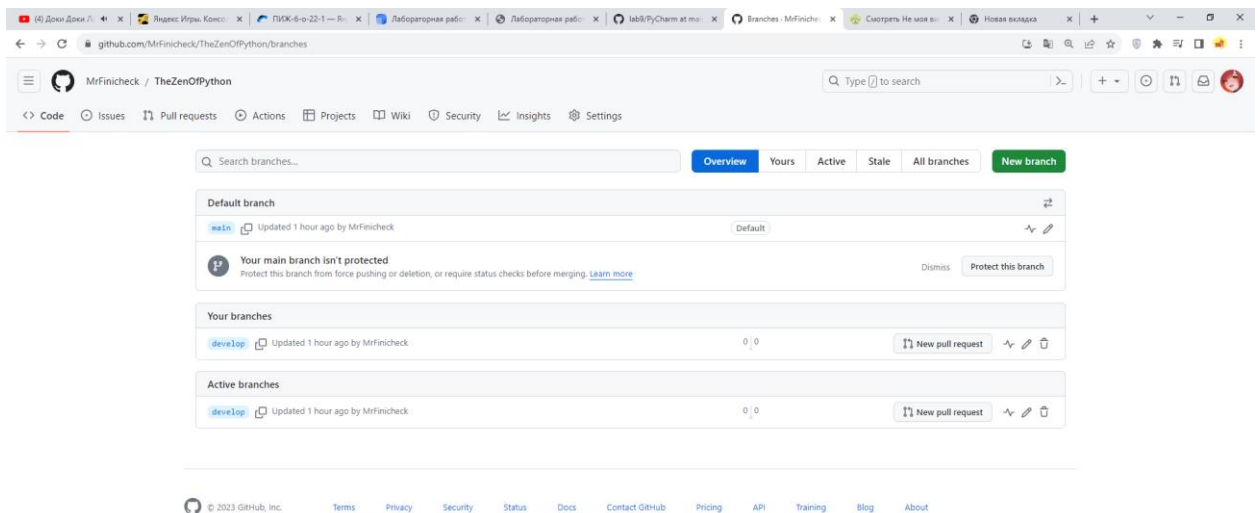


Рисунок 5.2 – Ветка develop на GitHub

## 6. Создала проект PyCharm в папке репозитория

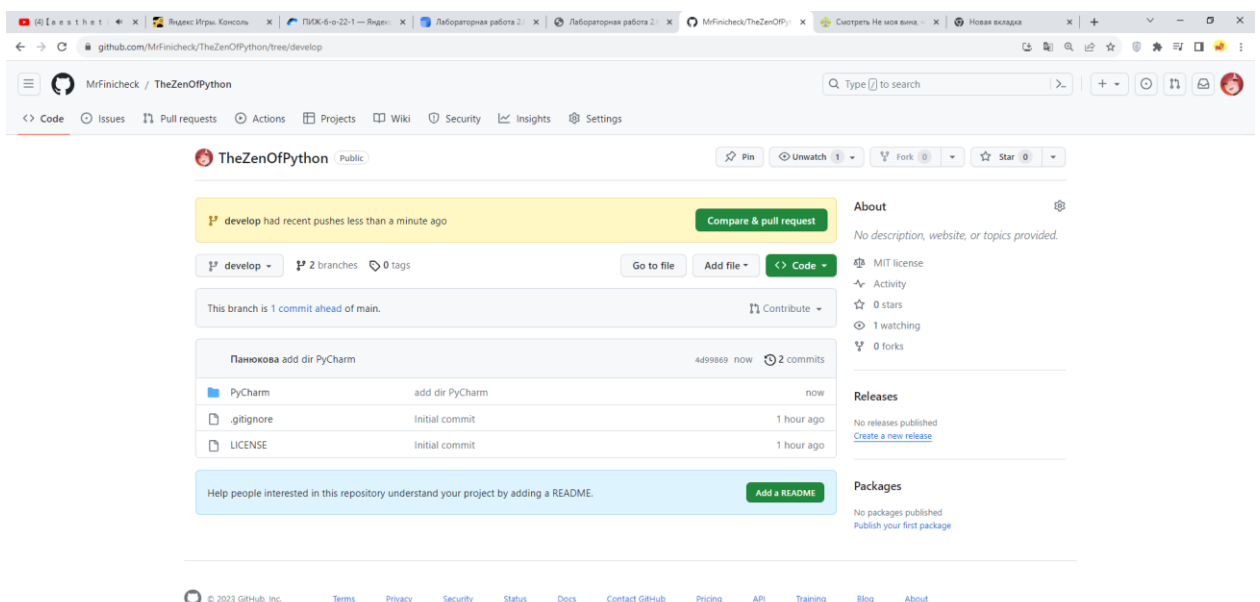


Рисунок 6.1 – Репозиторий с проектом PyCharm

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

```
Example333.py MYTask1.py MYTask2.py MYTask3.py Individual1.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  from datetime import date
6
7
8  if __name__ == '__main__':
9      # Список работников.
10     workers = []
11
12     # Организовать бесконечный цикл запроса команд.
13     while True:
14         # Запросить команду из терминала.
15         command = input(">>> ").lower()
16         # Выполнить действие в соответствие с командой.
17
18         if command == 'exit':
19             break
20
21         elif command == 'add':
22             # Запросить данные о работнике.
23             name = input("Фамилия и инициалы? ")
24             post = input("Должность? ")
25             year = int(input("Год поступления? "))
26
27             # Создать словарь.
28             worker = {'name': name, 'post': post, 'year': year}
29
30             # Добавить словарь в список.
31             workers.append(worker)
32
33             # Отсортировать список в случае необходимости.
34             if len(workers) > 1:
35                 workers.sort(key=lambda item: item.get('name', ''))
36
37         elif command == 'list':
38             # Заголовок таблицы.
39             line = '+-{}-+-{}-+-{}-+-{}-+'.format(
40                 *args: '-' * 4,
41                 '-' * 30,
42                 '-' * 20,
43                 '-' * 8
```

Рисунок 7.1 – Проработка примера 1

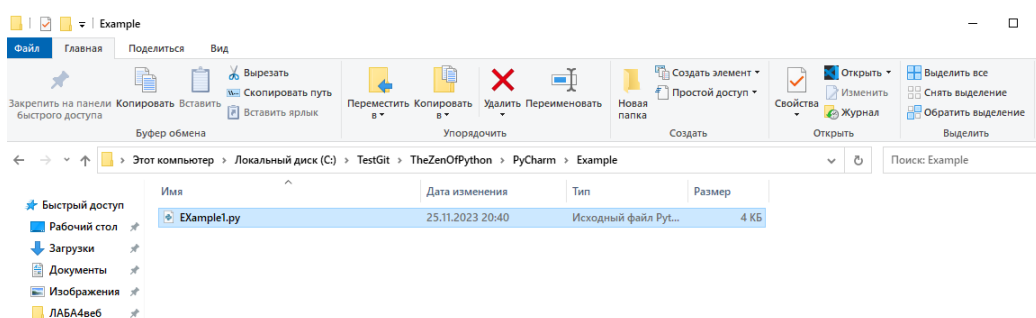


Рисунок 7.4 – Создание отдельного модуля для примера

```
C:\TestGit\TheZenOfPython>git add PyCharm
C:\TestGit\TheZenOfPython>git commit -m"adding example"
[develop 8c91d7b] adding example
2 files changed, 96 insertions(+)
create mode 100644 PyCharm/Example/Example1.py
delete mode 100644 "PyCharm/\321\202\320\265\320\272\321\201\321\202.txt"
```

Рисунок 7.5 – Фиксирование изменений в репозитории

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

```
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Фамилия и инициалы? Панюкова К. Ю.
Должность? Старший модератор Яндекса
Год поступления? 2023
>>> list
+-----+-----+-----+-----+
| 1 | Панюкова К. Ю. | Старший модератор Яндекса | 2023 |
+-----+-----+-----+-----+
>>> exit

Process finished with exit code 0
```

Рисунок 8.1 – Результат примера 1

9. Решила 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.

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

import sys

if __name__ == '__main__':
    # Заполняем список классов.
    school = {
        '1а': 25,
        '1б': 20,
        '2б': 18,
        '6а': 30,
        '7в': 22
    }

    # В одном из классов изменилось количество учащихся.
    school['6а'] = 29

    # В школе появился новый класс.
    school['9г'] = 27

    # В школе был расформирован (удален) другой класс.
    del school['7в']

    # Находим сумму.
    all_students = sum(school.values())

    # Вывод полученной суммы.
    print(f"Общее количество учащихся в школе: {total_students}")
```

Рисунок 9.1 – Код программы Task9.py в IDE PyCharm

## 10. Зафиксировала 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

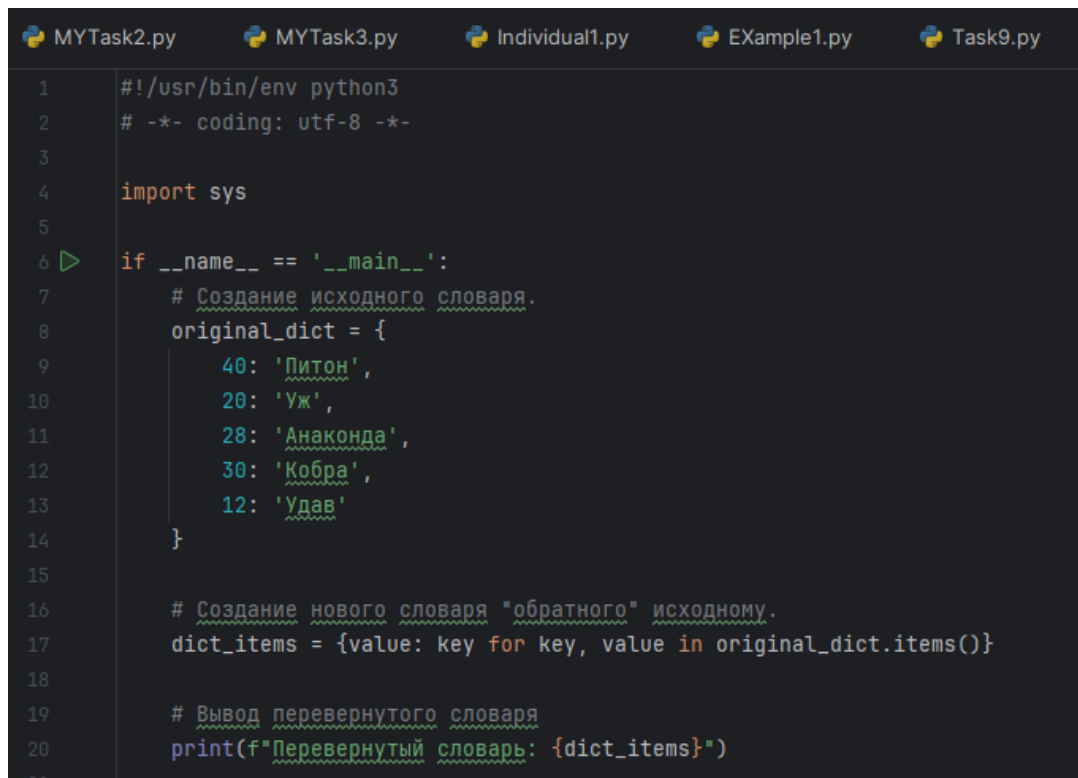
C:\TestGit\TheZenOfPython>git status
On branch develop
Your branch is ahead of 'origin/develop' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Рисунок 10.1 – Коммит файлов в репозитории git

11. Решила 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.

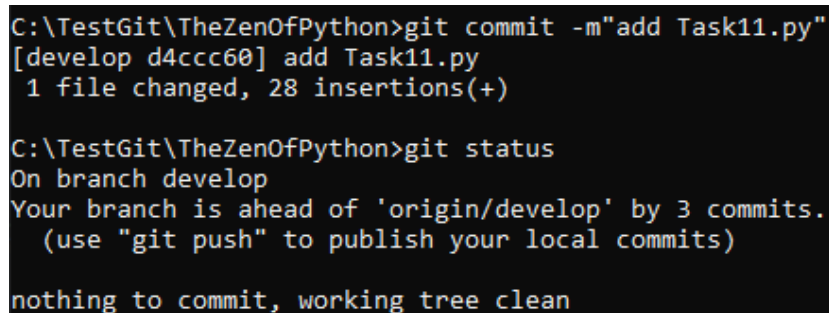




```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      # Создание исходного словаря.
8      original_dict = {
9          40: 'Питон',
10         20: 'Уж',
11         28: 'Анаконда',
12         30: 'Кобра',
13         12: 'Удав'
14     }
15
16     # Создание нового словаря "обратного" исходному.
17     dict_items = {value: key for key, value in original_dict.items()}
18
19     # Вывод перевернутого словаря
20     print(f"Перевернутый словарь: {dict_items}")
```

Рисунок 11.1 – Код программы Task11.py в IDE PyCharm

12. Зафиксировала сделанные изменения в репозитории



```
C:\TestGit\TheZenOfPython>git commit -m"add Task11.py"
[develop d4ccc60] add Task11.py
1 file changed, 28 insertions(+)

C:\TestGit\TheZenOfPython>git status
On branch develop
Your branch is ahead of 'origin/develop' by 3 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Рисунок 12.1 – Коммит файлов в репозитории git

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

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

import sys
from datetime import date

if __name__ == '__main__':
    # Список личностей.
    persons = []

    # 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 личности.
            name = input("Фамилия и имя? ")
            zodiac_sign = input("Знак Зодиака? ")
            birth_date = input("Дата рождения? ")

            # Создать словарь.
            person = {
                'name': name,
                'zodiac_sign': zodiac_sign,
                'birth_date': birth_date
            }

```

Рисунок 13.1 – Код программы INdividual.py в IDE PyCharm

14. Зафиксировала сделанные изменения в репозитории.

```
C:\TestGit\TheZenOfPython>git add .
C:\TestGit\TheZenOfPython>git commit -m"add individual task"
[develop e32a036] add individual task
3 files changed, 105 insertions(+), 4 deletions(-)
create mode 100644 PyCharm/Individual/INdividual.py
create mode 100644 "PyCharm/Individual/\320\222\320\260\321\200\320\270\320\260\320\275\321\20216.txt"
```

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.