

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
Отчет по лабораторной работе № 2.6
«Работа со словарями в языке Python»
по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » декабря 2022 г.

Подпись студента _____

Работа защищена

« » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2022

Цель работы:

Приобретение навыков по работе со словарями при написании программ с помощью языка программирования Python версии 3.x.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: <https://github.com/afk552/lab2.6>

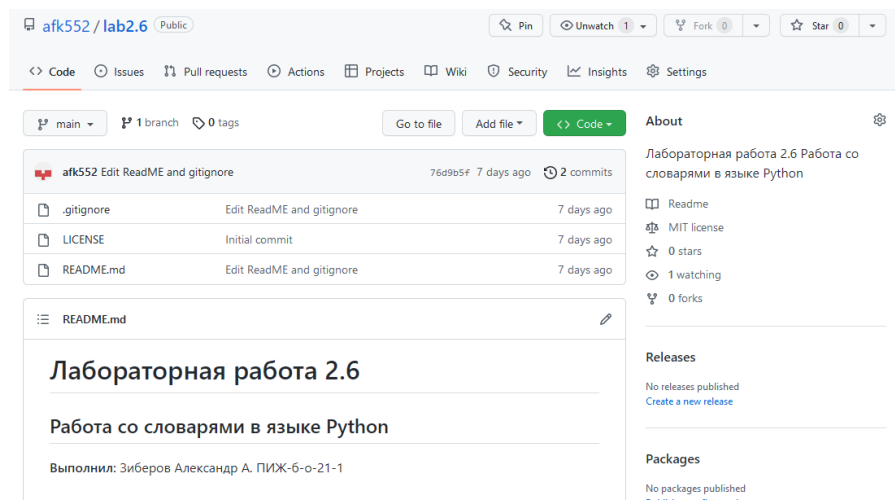


Рисунок 1 – Репозиторий GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

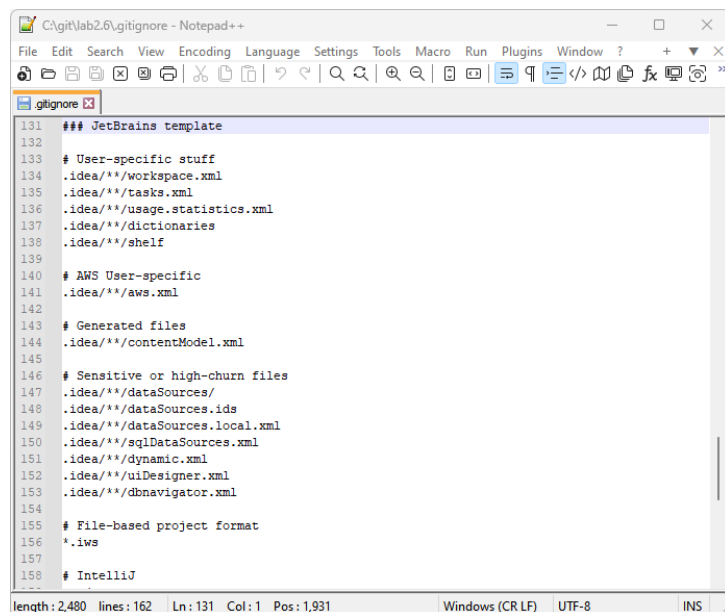
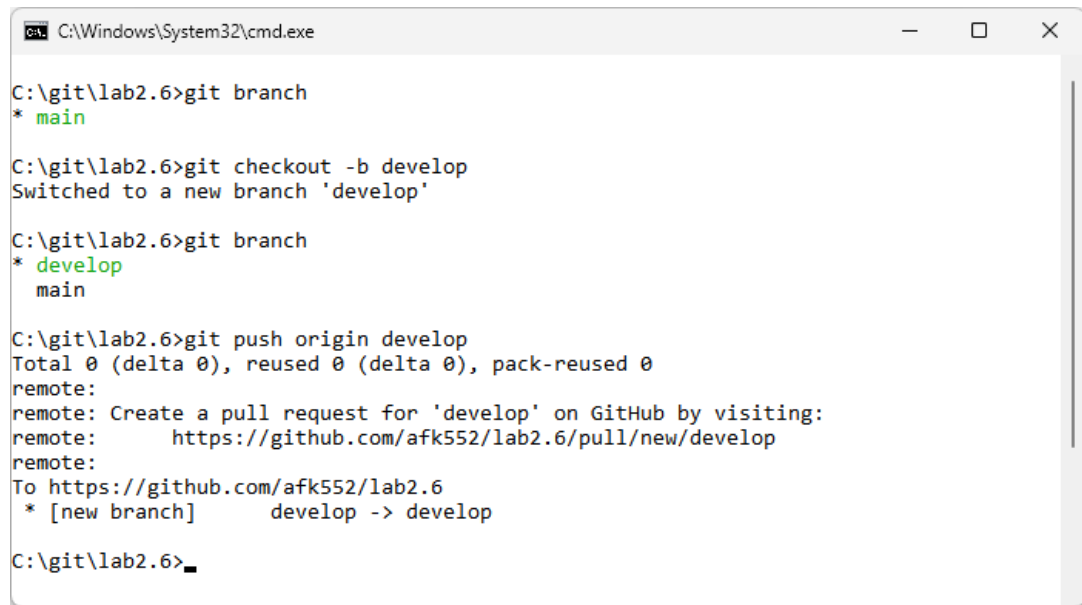


Рисунок 2 – Окно блокнота

Организируйте свой репозиторий в соответствии с моделью ветвления git-flow, рисунок 3.



```
C:\Windows\System32\cmd.exe

C:\git\lab2.6>git branch
* main

C:\git\lab2.6>git checkout -b develop
Switched to a new branch 'develop'

C:\git\lab2.6>git branch
* develop
  main

C:\git\lab2.6>git push 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/afk552/lab2.6/pull/new/develop
remote:
To https://github.com/afk552/lab2.6
 * [new branch]      develop -> develop

C:\git\lab2.6>
```

Рисунок 3 – Окно командной строки

Создайте проект PyCharm в папке репозитория, рисунок 4.

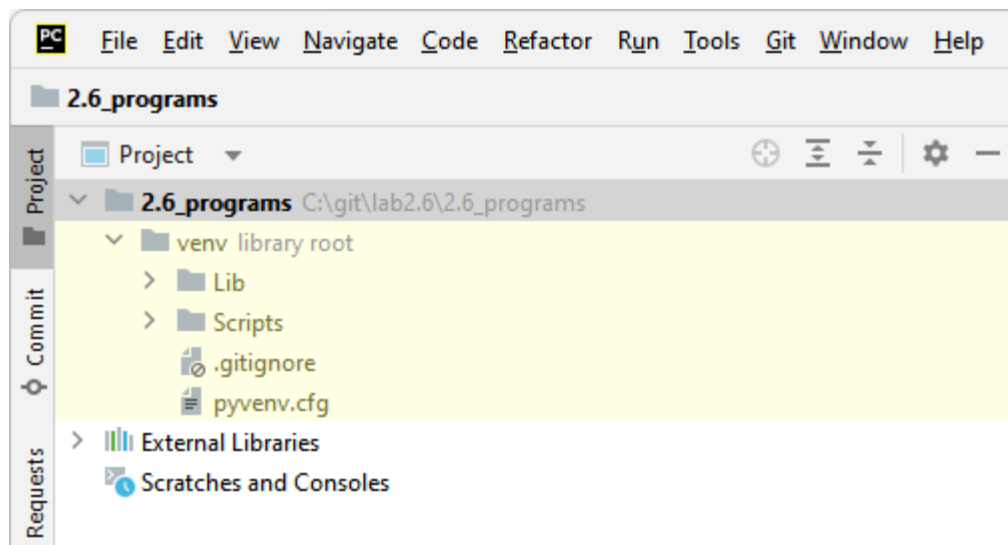


Рисунок 4 – Окно проекта в PyCharm

Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных, вводимых с клавиатуры. Рисунки 5-6.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  from datetime import date
6
7  if __name__ == '__main__':
8      # Список работников.
9      workers = []
10     # Организовать бесконечный цикл запроса команд.
11     while True:
12         # Запросить команду из терминала.
13         command = input(">>> ").lower()
14         # Выполнить действие в соответствие с командой.
15         if command == 'exit':
16             break
17
18         elif command == 'add':
19             # Запросить данные о работнике.
20             name = input("Фамилия и инициалы? ")
21             post = input("Должность? ")
22             year = int(input("Год поступления? "))
23             # Создать словарь.
24             worker = {
25                 'name': name,
26                 'post': post,
27                 'year': year,
28             }
29             # Добавить словарь в список.
30             workers.append(worker)
31             # Отсортировать список в случае необходимости.
32             if len(workers) > 1:
33                 workers.sort(key=lambda item: item.get('name', ''))
34
35         elif command == 'list':
36             # Заголовок таблицы.
37             line = '+-{}-{}-{}-{}-{}-{}-+'.format(
38                 '-' * 4,
39                 '-' * 30,
40                 '-' * 20,
41                 '-' * 8
42             )
43             print(line)
44             print(
45                 '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
46                     "No",
47                     "Ф.И.О.",
48                     "Должность",
49                     "Год"
50                 )
51             )
52             print(line)
53             # Вывести данные о всех сотрудниках.
54
55             for idx, worker in enumerate(workers, 1):
56                 print(
57                     '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
58                         idx,
59                         worker.get('name', ''),
60                         worker.get('post', ''),
61                         worker.get('year', 0)
62                     )
63                 )
64                 print(line)
65
66         elif command.startswith('select '):
67             # Получить текущую дату.
68             today = date.today()
69             # Разбить команду на части для выделения номера года.
70             parts = command.split(' ', maxsplit=1)
71             # Получить требуемый стаж.
72             period = int(parts[1])
73             # Инициализировать счетчик.
74             count = 0
75             # Проверить сведения работников из списка.
76             for worker in workers:
77                 if today.year - worker.get('year', today.year) >= period:
78                     count += 1
79                     print(
80                         '{:>4}: {}'.format(count, worker.get('name', ''))
81                     )
82             # Если счетчик равен 0, то работники не найдены.
83             if count == 0:
84                 print("Работники с заданным стажем не найдены.")
85
86         elif command == 'help':
87             # Вывести справку о работе с программой.
88             print("Список команд:\n")
89             print("add - добавить работника;")
90             print("list - вывести список работников;")
91             print("select <стаж> - запросить работников со стажем;")
92             print("help - отобразить справку;")
93             print("exit - завершить работу с программой.")
94         else:
95             print(f"Неизвестная команда {command}", file=sys.stderr)

```

Рисунок 5 – Код программы из примера 1

```
Run: example1 x
C:\git\lab2.6\2.6_programs\venv\Scripts\python.exe C:\git\lab2.6\2.6_programs\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.
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? Рабочий
Год поступления? 2077
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Иванов И.И.              |      Рабочий       |      2077     |
+-----+-----+-----+-----+
>>> add
Фамилия и инициалы? Детров Д.С.
Должность? Менеджер
Год поступления? 1990
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Детров Д.С.              |      Менеджер       |      1990     |
|  2 | Иванов И.И.              |      Рабочий       |      2077     |
+-----+-----+-----+-----+
>>> select 32
      1: Детров Д.С.
>>> exit

Process finished with exit code 0
```

Рисунок 6 – Результат выполнения программы из примера 1

```
C:\Windows\System32\cmd.exe

C:\git\lab2.6>git add .

C:\git\lab2.6>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   2.6_programs/example1.py

C:\git\lab2.6>git commit -m "Add example"
[develop 74f3a07] Add example
 1 file changed, 94 insertions(+)
 create mode 100644 2.6_programs/example1.py

C:\git\lab2.6>
```

Рисунок 7 – Окно командной строки

Задание 1. Решите 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.

```
task1.py x
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == '__main__':
5      school = {
6          "1А": 21,
7          "1Б": 33,
8          "2А": 10,
9          "2Б": 23,
10         "3А": 30,
11         "3Б": 26,
12     }
13
14     for key, value in school.items():
15         print(f"В классе: {key}: {value} учеников")
16
17     print("\nВ одном из классов изменилось количество учащихся!")
18     school['1А'] = 99
19     for key, value in school.items():
20         print(f"В классе: {key}: {value} учеников")
21
22     print("\nВ школе появился новый класс!")
23     school.setdefault("11А", 10)
24     for key, value in school.items():
25         print(f"В классе: {key}: {value} учеников")
26
27     print("\nВ школе был расформирован (удален) один из классов!")
28     school.pop("1Б")
29     for key, value in school.items():
30         print(f"В классе: {key}: {value} учеников")
31
32     amount = 0
33     for value in school.values():
34         amount += value
35     print(f"\nВсего учеников: {amount}")
36
```

Рисунок 8 – Код программы задания 1

```
Run: task1 x
C:\git\lab2.6\2.6_programs\venv\Scripts\python.exe C:\git\lab2.6\2.6_programs\task1.py
В классе: 1А: 21 учеников
В классе: 1Б: 33 учеников
В классе: 2А: 10 учеников
В классе: 2Б: 23 учеников
В классе: 3А: 30 учеников
В классе: 3Б: 26 учеников

В одном из классов изменилось количество учащихся!
В классе: 1А: 99 учеников
В классе: 1Б: 33 учеников
В классе: 2А: 10 учеников
В классе: 2Б: 23 учеников
В классе: 3А: 30 учеников
В классе: 3Б: 26 учеников

В школе появился новый класс!
В классе: 1А: 99 учеников
В классе: 1Б: 33 учеников
В классе: 2А: 10 учеников
В классе: 2Б: 23 учеников
В классе: 3А: 30 учеников
В классе: 3Б: 26 учеников
В классе: 11А: 10 учеников

В школе был расформирован (удален) один из классов!
В классе: 1А: 99 учеников
В классе: 2А: 10 учеников
В классе: 2Б: 23 учеников
В классе: 3А: 30 учеников
В классе: 3Б: 26 учеников
В классе: 11А: 10 учеников

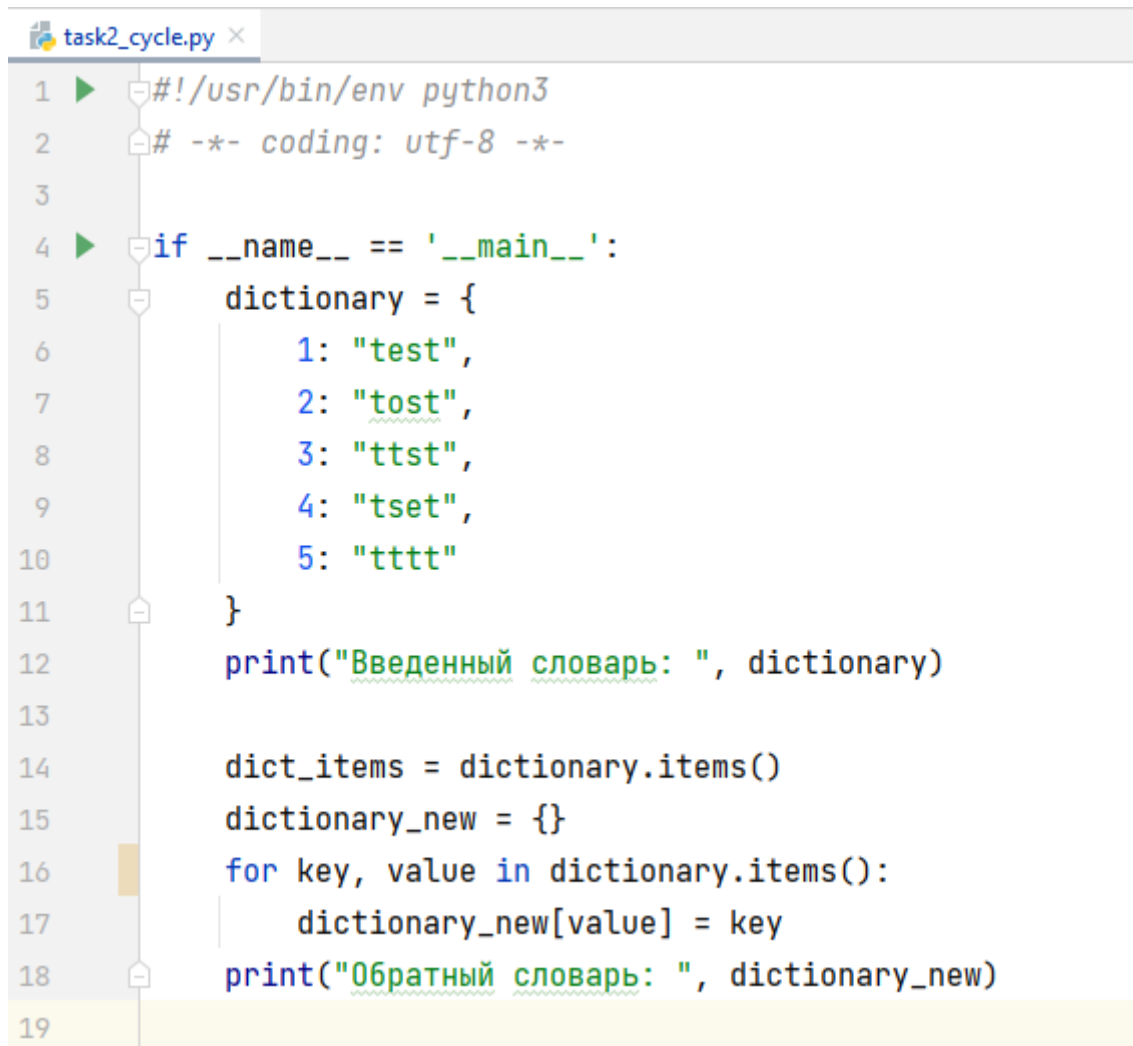
Всего учеников: 198

Process finished with exit code 0
```

Рисунок 9 – Результат выполнения программы задания 1

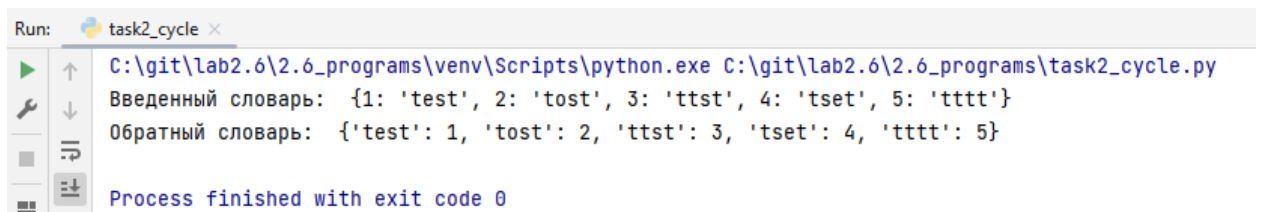
Задание 2. Решите задачу: создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод `items()`, и с помощью полученного объекта `dict_items` создайте новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.

Рисунки 10-13.



```
task2_cycle.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      dictionary = {
6          1: "test",
7          2: "tost",
8          3: "ttst",
9          4: "tset",
10         5: "tttt"
11     }
12     print("Введенный словарь: ", dictionary)
13
14     dict_items = dictionary.items()
15     dictionary_new = {}
16     for key, value in dictionary.items():
17         dictionary_new[value] = key
18     print("Обратный словарь: ", dictionary_new)
19
```

Рисунок 10 – Код программы задания 2 (через циклы)



```
Run: task2_cycle x
C:\git\lab2.6\2.6_programs\venv\Scripts\python.exe C:\git\lab2.6\2.6_programs\task2_cycle.py
Введенный словарь: {1: 'test', 2: 'tost', 3: 'ttst', 4: 'tset', 5: 'tttt'}
Обратный словарь: {'test': 1, 'tost': 2, 'ttst': 3, 'tset': 4, 'tttt': 5}
Process finished with exit code 0
```

Рисунок 11 – Результат выполнения программы задания 2 (через циклы)


```
task2_dc.py x
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == '__main__':
5      dictionary = {
6          1: "test",
7          2: "tost",
8          3: "ttst",
9          4: "tset",
10         5: "tttt"
11     }
12     print("Введенный словарь: ", dictionary)
13
14     dict_items = dictionary.items()
15     dictionary_new = {key: value for value, key in dict_items}
16     print("Обратный словарь: ", dictionary_new)
17
```

Рисунок 12 – Код программы задания 2 (через словари включений)

```
Run: task2_dc x
C:\git\lab2.6\2.6_programs\venv\Scripts\python.exe C:\git\lab2.6\2.6_programs\task2_dc.py
Введенный словарь: {1: 'test', 2: 'tost', 3: 'ttst', 4: 'tset', 5: 'tttt'}
Обратный словарь: {'test': 1, 'tost': 2, 'ttst': 3, 'tset': 4, 'tttt': 5}

Process finished with exit code 0
```

Рисунок 13 – Результат выполнения программы задания 2 (словари включений)

<pre>C:\Windows\System32\cmd.exe C:\git\lab2.6>git add . C:\git\lab2.6>git status On branch develop Changes to be committed: (use "git restore --staged <file>..." to unstage) new file: 2.6_programs/task1.py new file: 2.6_programs/task2.py C:\git\lab2.6>git commit -m "Add tasks" [develop 6b9398c] Add tasks 2 files changed, 54 insertions(+) create mode 100644 2.6_programs/task1.py create mode 100644 2.6_programs/task2.py C:\git\lab2.6></pre>	<pre>C:\Windows\System32\cmd.exe C:\git\lab2.6>git add . C:\git\lab2.6>git status On branch develop Changes to be committed: (use "git restore --staged <file>..." to unstage) renamed: 2.6_programs/task2.py -> 2.6_programs/task2_cycle.py new file: 2.6_programs/task2_dc.py C:\git\lab2.6>git commit -m "Add dict. compr. for task 2" [develop 86cc293] Add dict. compr. for task 2 2 files changed, 17 insertions(+), 1 deletion(-) rename 2.6_programs/{task2.py => task2_cycle.py} (90%) create mode 100644 2.6_programs/task2_dc.py C:\git\lab2.6></pre>
--	---

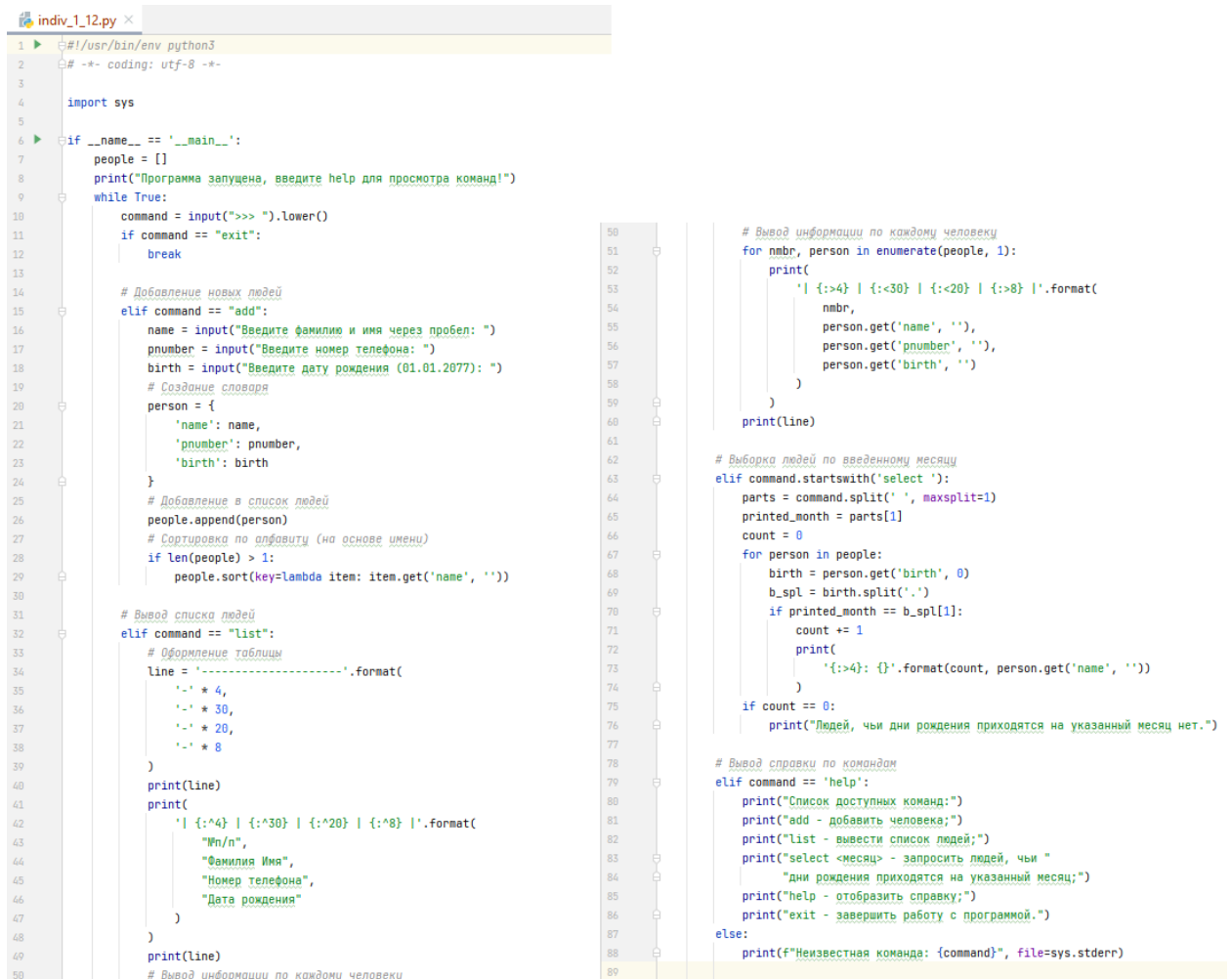
Рисунок 14 – Окно командной строки

Выполните индивидуальные задания, согласно своего варианта.

Вариант 12

Задание 1. Использовать словарь, содержащий следующие ключи: фамилия, имя; номер телефона; дата рождения (список из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть размещены по алфавиту; вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение.

Рисунки 15-16.



```
1  #!/usr/bin/env python3
2  #-*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      people = []
8      print("Программа запущена, введите help для просмотра команд!")
9      while True:
10         command = input(">>> ").lower()
11         if command == "exit":
12             break
13
14         # Добавление новых людей
15         elif command == "add":
16             name = input("Введите фамилию и имя через пробел: ")
17             pnumber = input("Введите номер телефона: ")
18             birth = input("Введите дату рождения (01.01.2077): ")
19             # Создание словаря
20             person = {
21                 'name': name,
22                 'pnumber': pnumber,
23                 'birth': birth
24             }
25             # Добавление в список людей
26             people.append(person)
27             # Сортировка по алфавиту (на основе имени)
28             if len(people) > 1:
29                 people.sort(key=lambda item: item.get('name', ''))
30
31         # Вывод списка людей
32         elif command == "list":
33             # Оформление таблицы
34             line = "-----".format(
35                 '-' * 4,
36                 '-' * 30,
37                 '-' * 20,
38                 '-' * 8
39             )
40             print(line)
41             print(
42                 '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
43                     "\n/n",
44                     "Фамилия Имя",
45                     "Номер телефона",
46                     "Дата рождения"
47                 )
48             )
49             print(line)
50             # Вывод информации по каждому человеку
51             for nmb, person in enumerate(people, 1):
52                 print(
53                     '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
54                         nmb,
55                         person.get('name', ''),
56                         person.get('pnumber', ''),
57                         person.get('birth', '')
58                     )
59                 )
60                 print(line)
61
62         # Выборка людей по введенному месяцу
63         elif command.startswith('select '):
64             parts = command.split(' ', maxsplit=1)
65             printed_month = parts[1]
66             count = 0
67             for person in people:
68                 birth = person.get('birth', 0)
69                 b_spl = birth.split('.')
70                 if printed_month == b_spl[1]:
71                     count += 1
72                     print(
73                         '| {:>4} |'.format(count, person.get('name', ''))
74                     )
75             if count == 0:
76                 print("Людей, чьи дни рождения приходятся на указанный месяц нет.")
77
78         # Вывод справки по командам
79         elif command == "help":
80             print("Список доступных команд:")
81             print("add - добавить человека;")
82             print("list - вывести список людей;")
83             print("select <месяц> - запросить людей, чьи ")
84                 print("дни рождения приходятся на указанный месяц;")
85             print("help - отобразить справку;")
86             print("exit - завершить работу с программой.")
87         else:
88             print(f"Неизвестная команда: {command}", file=sys.stderr)
89
```

Рисунок 15 – Код программы индивидуального задания 1

```
Run: C:\git\lab2.6\2.6_programs\venv\Scripts\python.exe C:\git\lab2.6\2.6_programs\indiv_1_12.py
Программа запущена, введите help для просмотра команд!
>>> help
Список доступных команд:
add - добавить человека;
list - вывести список людей;
select <месяц> - запросить людей, чьи дни рождения приходятся на указанный месяц;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Введите фамилию и имя через пробел: Андреев Андрей
Введите номер телефона: 894326427123
Введите дату рождения (01.01.2077): 01.03.2001
>>> add
Введите фамилию и имя через пробел: Ганичев Валентин
Введите номер телефона: 8943274234
Введите дату рождения (01.01.2077): 23.04.2000
>>> add
Введите фамилию и имя через пробел: Сухачев Денис
Введите номер телефона: 89547374574
Введите дату рождения (01.01.2077): 24.04.1999
>>> add
Введите фамилию и имя через пробел: Детров Алексей
Введите номер телефона: 89324724773
Введите дату рождения (01.01.2077): 03.03.1994
>>> list
-----
| №п/п |          Фамилия Имя          |   Номер телефона   |   Дата рождения   |
-----
|   1   | Андреев Андрей                | 894326427123       | 01.03.2001        |
|   2   | Ганичев Валентин              | 8943274234         | 23.04.2000        |
|   3   | Детров Алексей                | 89324724773        | 03.03.1994        |
|   4   | Сухачев Денис                 | 89547374574        | 24.04.1999        |
-----
>>> select 03
1: Андреев Андрей
2: Детров Алексей
>>> select 04
1: Ганичев Валентин
2: Сухачев Денис
>>> select 10
1: Петров Петр
>>> select 12
Людей, чьи дни рождения приходятся на указанный месяц нет.
>>> не хочу вводить команду :(
>>> Известная команда: не хочу вводить команду :(
```

Рисунок 16 – Результаты выполнения программы индивидуального задания 1

```
C:\Windows\System32\cmd.exe

C:\git\lab2.6>git add .

C:\git\lab2.6>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   2.6_programs/indiv_1_12.py

C:\git\lab2.6>git commit -m "Add individual task"
[develop 90541b9] Add individual task
 1 file changed, 88 insertions(+)
 create mode 100644 2.6_programs/indiv_1_12.py

C:\git\lab2.6>
```

Рисунок 17 – Окно командной строки

Приведите в отчете скриншоты работы программ и UML-диаграммы деятельности решения индивидуального задания.

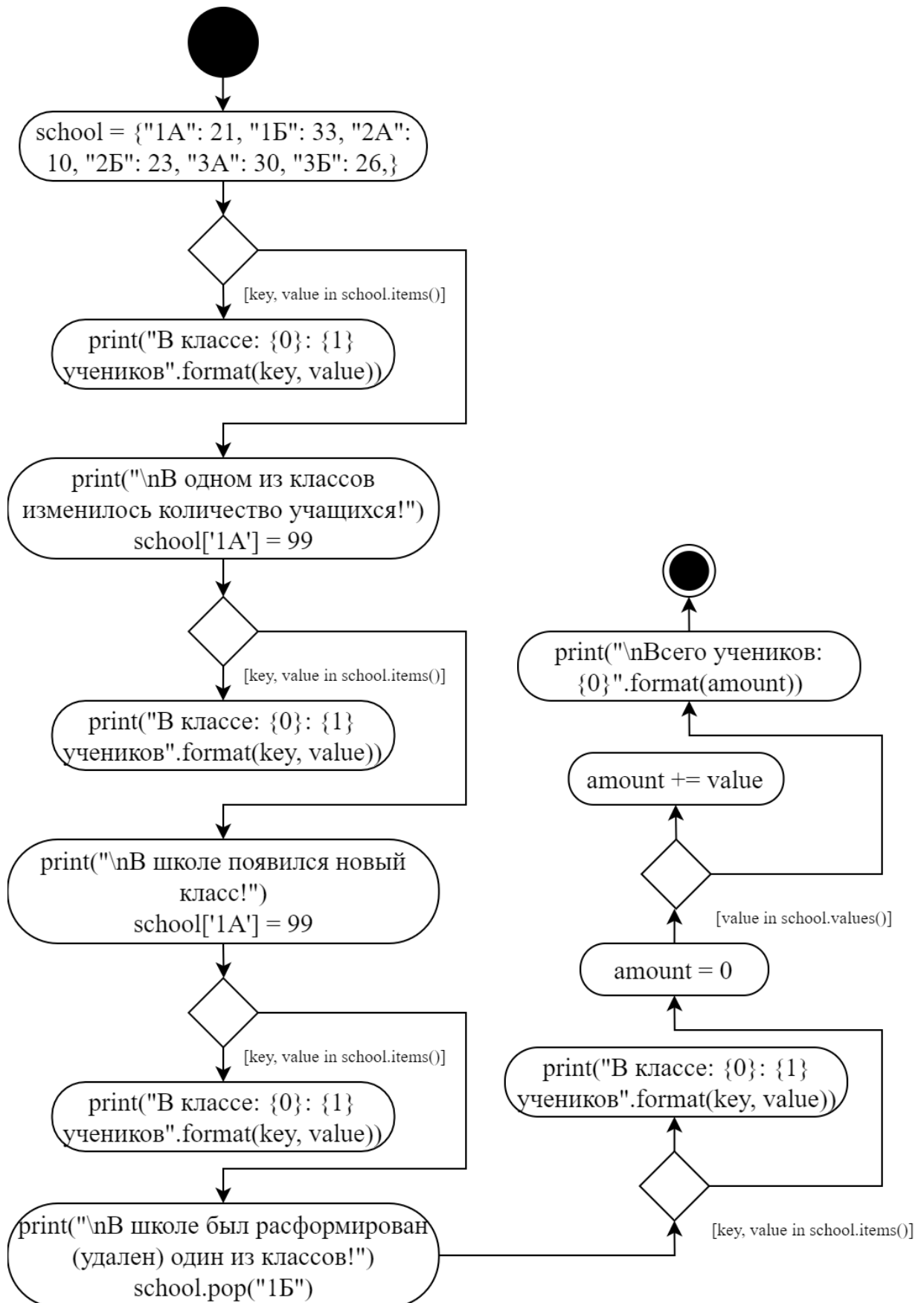


Рисунок 18 – UML-диаграмма деятельности программы задания 1

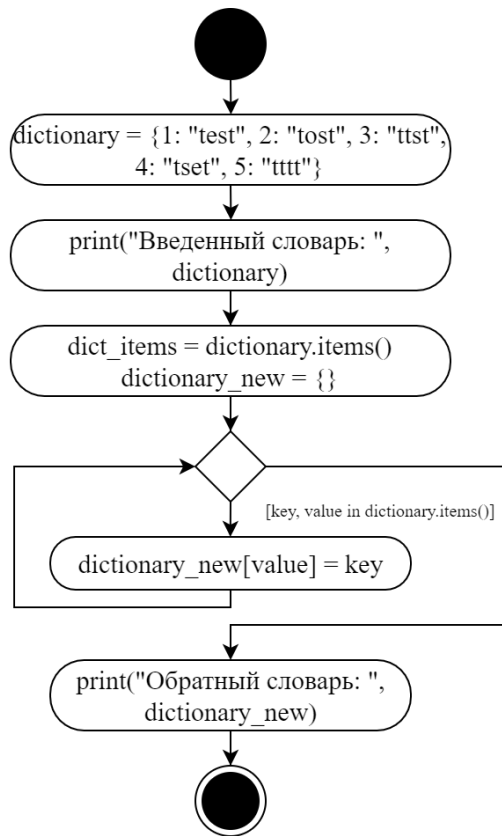


Рисунок 19 – UML-диаграмма деятельности программы задания 2 (через циклы)

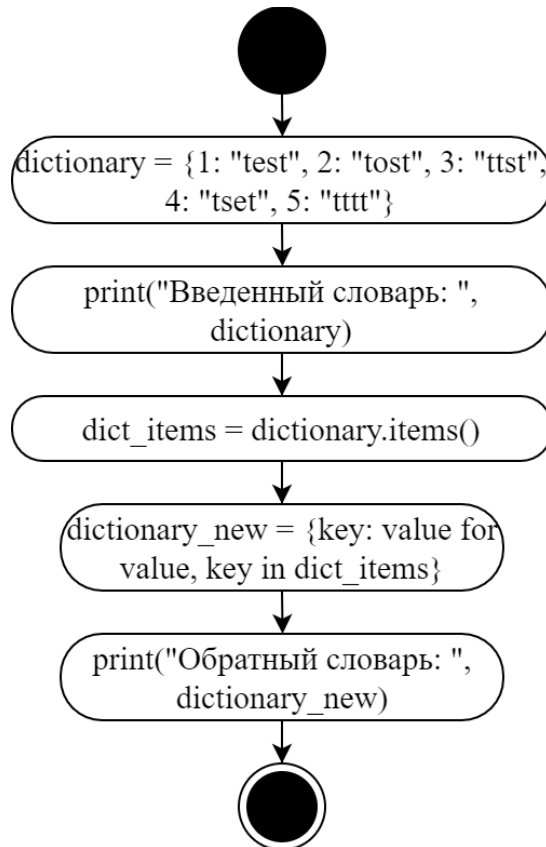


Рисунок 20 – UML-диаграмма деятельности программы задания 2 (через словари включений)

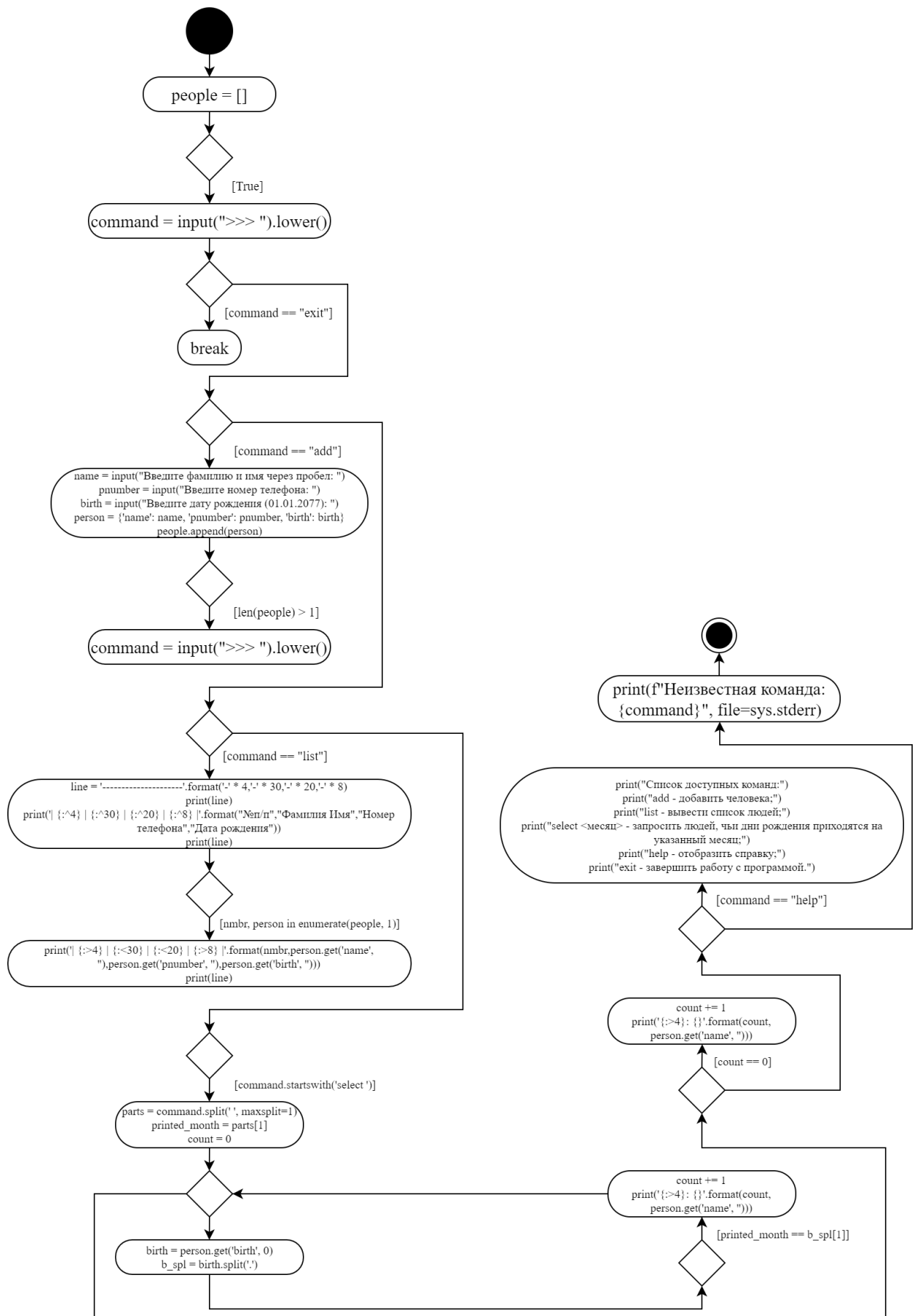


Рисунок 21 – UML-диаграмма деятельности программы индивидуального задания

Выполните слияние ветки для разработки с веткой main / master.
Отправьте сделанные изменения на сервер GitHub. Рисунки 13, 14.

```
C:\Windows\System32\cmd.exe

C:\git\lab2.6>git branch
* develop
  main

C:\git\lab2.6>git push origin develop
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (18/18), 4.52 KiB | 2.26 MiB/s, done.
Total 18 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/afk552/lab2.6
  76d9b5f..90541b9  develop -> develop

C:\git\lab2.6>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git\lab2.6>git merge develop
Updating 76d9b5f..90541b9
Fast-forward
 2.6_programs/example1.py | 94 +++++
 2.6_programs/indiv_1_12.py | 88 +++++
 2.6_programs/task1.py      | 36 +++++
 2.6_programs/task2_cycle.py | 18 +++++
 2.6_programs/task2_dc.py   | 16 +++++
 5 files changed, 252 insertions(+)
 create mode 100644 2.6_programs/example1.py
 create mode 100644 2.6_programs/indiv_1_12.py
 create mode 100644 2.6_programs/task1.py
 create mode 100644 2.6_programs/task2_cycle.py
 create mode 100644 2.6_programs/task2_dc.py

C:\git\lab2.6>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/afk552/lab2.6
  76d9b5f..90541b9  main -> main

C:\git\lab2.6>
```

Рисунок 22 – Окно командной строки

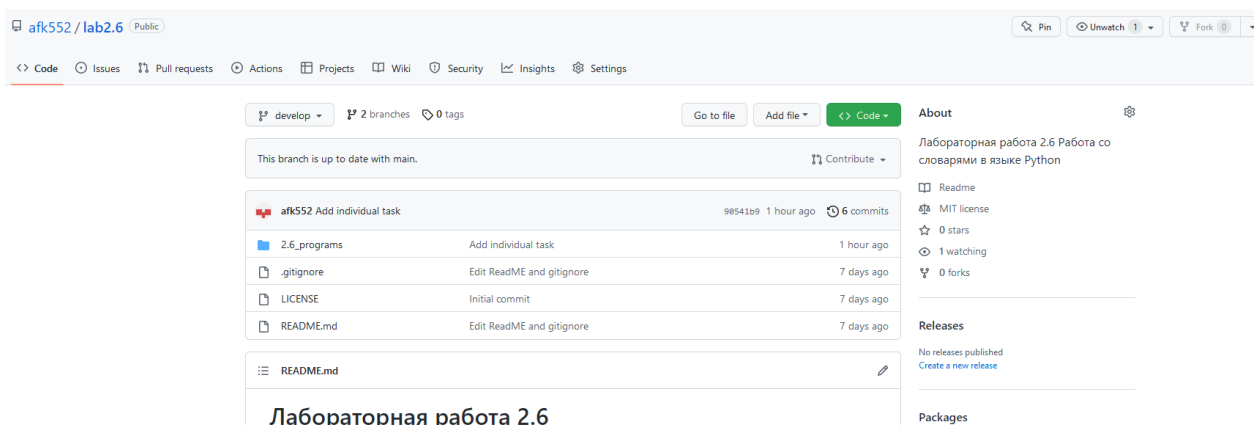


Рисунок 23 – Удаленный репозиторий на GitHub

Вывод: В результате выполнения работы были изучены словари в языке программирования Python 3, работа и методы взаимодействия с ними и написание программ с их использованием.

Контрольные вопросы:

1. Что такое словари в языке Python?

Словарь (dict) представляет собой структуру данных (которая ещё называется ассоциативный массив), предназначенную для хранения произвольных объектов с доступом по ключу. Данные в словаре хранятся в формате ключ – значение.

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

Да, может.

3. Какие методы обхода словарей Вам известны?

```
for i in nums:
```

```
    print(nums[i])
```

```
for key, value in nums.items():
```

```
    print(key, 'is', value)
```

Методы словаря `keys()` и `values()` позволяют получить отдельно перечни ключей и значений. Так что если, например, надо перебрать только значения или только ключи, лучше воспользоваться одним из этих методов.

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

```
dict["key"]
```

```
dict.get("key")
```

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

```
dict.setdefault("key", "value")
```

```
dict["key"] = "value"
```


6. Что такое словарь включений?

Словарь включений 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.

У функции zip() множество сценариев применения. Например, она пригодится, если нужно создать набор словарей из двух массивов, каждый из которых содержит имя и номер сотрудника.

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?

Модуль datetime предоставляет классы для обработки времени и даты разными способами. Он позволяет управлять датами и временем, представляя их в таком виде, в котором пользователи смогут их понимать. Так, он состоит из объектов следующих типов:

`date` — хранит дату

`time` — хранит время

`datetime` — хранит дату и время

Некоторые возможности модуля:

Получить текущую дату: `dt_now = datetime.datetime.now()`

Получить текущее время: `current_time = current_date_time.time()`

Для создания объекта даты нужно передать дату с использованием следующего синтаксиса:

`datetime.datetime(year,month,day)`

`date_obj = datetime.datetime(2022,12,01)`

Также, можно вычислять разницу между датами при помощи вычитания двух объектов дат.