

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа со словарями в языке Python»

ОТЧЕТ
по лабораторной работе №9
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

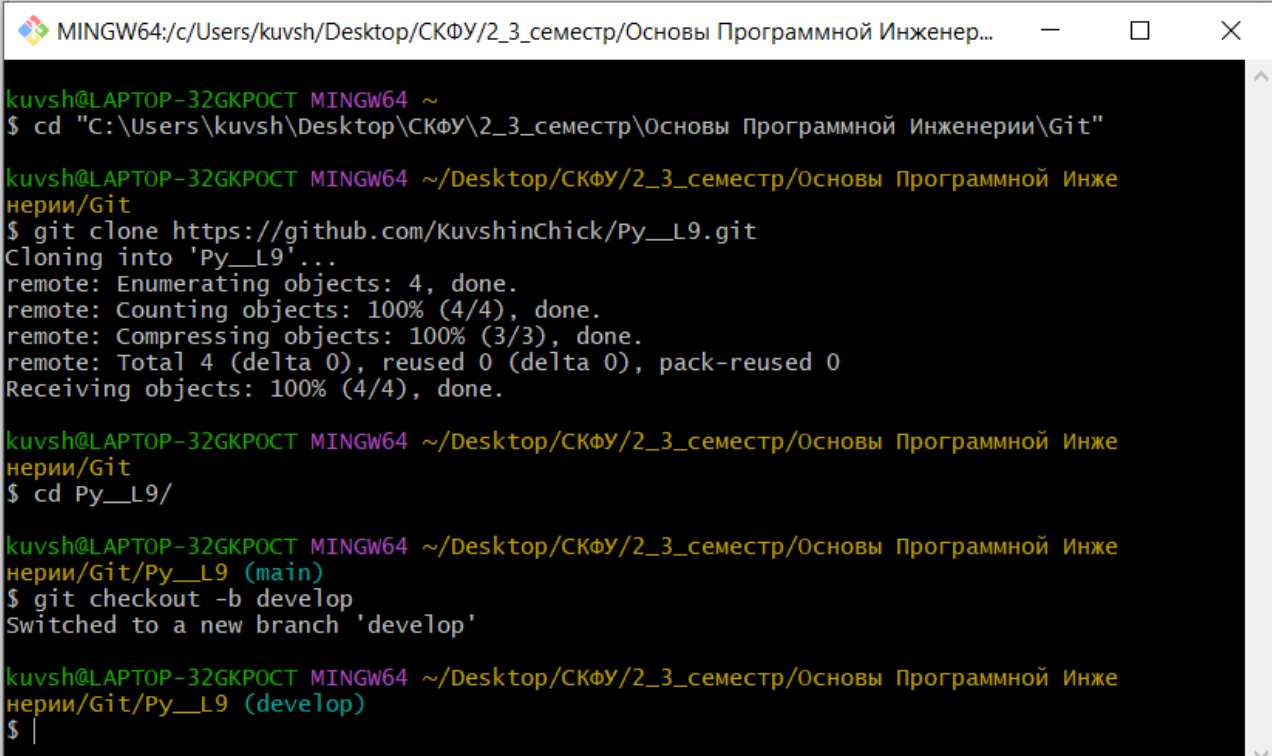
Ставрополь, 2022 г.

Цель работы: приобретение навыков по работе со словарями при написании программ с помощью языка программирования Python версии 3.x.

Ссылка на репозиторий: https://github.com/KuvshinChick/Py__L9.git

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
MINGW64:/c/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программной Инженер...
kuvsh@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git"

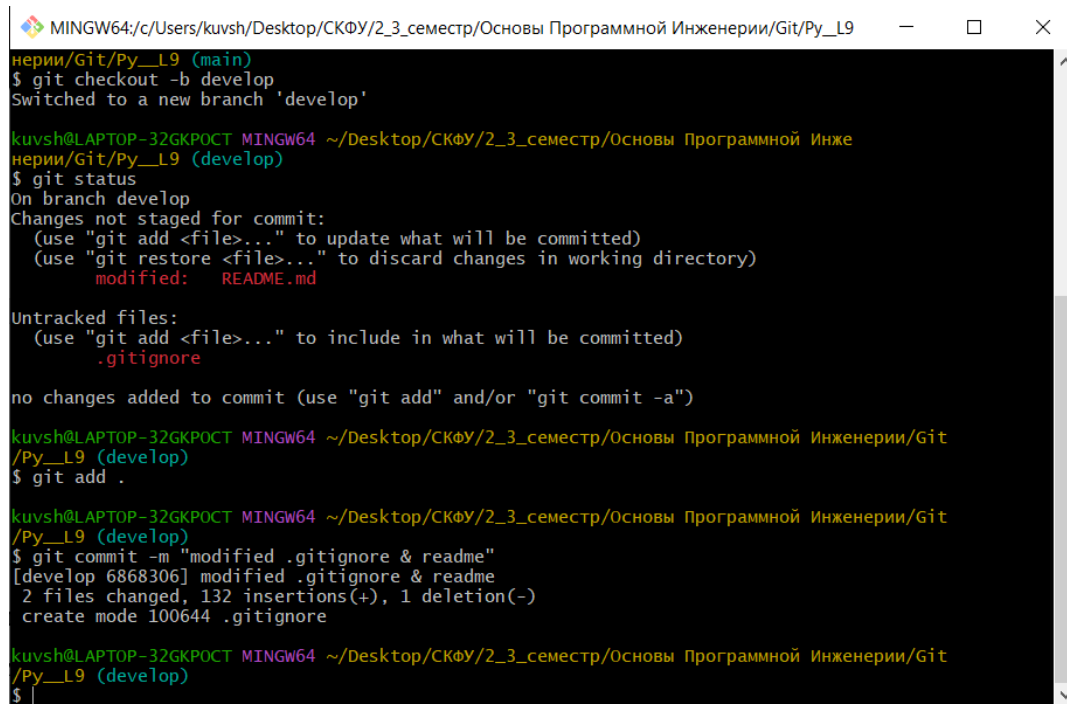
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git clone https://github.com/KuvshinChick/Py__L9.git
Cloning into 'Py__L9'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ cd Py__L9/

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L9 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L9 (develop)
$ |
```

Рисунок 9.1 – Клонирование репозитория и создание ветки develop



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L9
квери/Git/Py_L9 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

кувш@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
квери/Git/Py_L9 (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

кувш@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
/Py_L9 (develop)
$ git add .

кувш@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
/Py_L9 (develop)
$ git commit -m "modified .gitignore & readme"
[develop 6868306] modified .gitignore & readme
 2 files changed, 132 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore

кувш@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
/Py_L9 (develop)
$
```

Рисунок 9.2 – Обновление .gitignore и readme

6. Создайте проект PyCharm в папке репозитория.
7. Проработайте пример лабораторной работы. Создайте для него отдельный модуль языка Python. Зафиксируйте изменения в репозитории.
9. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных вводимых с клавиатуры.

Пример 1.

Использовать словарь, содержащий следующие ключи: фамилия и инициалы работника; название занимаемой должности; год поступления на работу. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в список, состоящий из заданных словарей;
- записи должны быть размещены по алфавиту;
- вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры;
- если таких работников нет, вывести на дисплей соответствующее сообщение.

```
ex_1.py x
Q- Cc W .* 0 results
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
11     # Организовать бесконечный цикл запроса команд.
12     while True:
13         # Запросить команду из терминала.
14         command = input(">>> ").lower()
15
16         # Выполнить действие в соответствие с командой.
17         if command == 'exit':
18             break
19
20         elif command == 'add':
21             # Запросить данные о работнике.
22             name = input("Фамилия и инициалы? ")
23             post = input("Должность? ")
24             year = int(input("Год поступления? "))
25
```

```
ex_1.py x
Q- Cc W .* 0 results
25
26     # Создать словарь.
27     worker = {
28         'name': name,
29         'post': post,
30         'year': year,
31     }
32
33     # Добавить словарь в список.
34     workers.append(worker)
35     # Отсортировать список в случае необходимости.
36     if len(workers) > 1:
37         workers.sort(key=lambda item: item.get('name', ''))
38
39     elif command == 'list':
40         # Заголовок таблицы.
41         line = '+--{}+--{}+--{}+--{}+--'.format(
42             '-' * 4,
43             '-' * 30,
44             '-' * 20,
45             '-' * 8
46         )
```

```
ex_1.py x
Q- Cc W * 0 results
47 print(line)
48 print(
49     '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
50         "№",
51         "Ф.И.О.",
52         "Должность",
53         "Год"
54     )
55 )
56 print(line)
57
58 # Вывести данные о всех сотрудниках.
59 for idx, worker in enumerate(workers, 1):
60     print(
61         '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
62             idx,
63             worker.get('name', ''),
64             worker.get('post', ''),
65             worker.get('year', 0)
66         )
67     )
68     print(line)
69
70 elif command.startswith('select'):
71     # Получить текущую дату.
72     today = date.today()
73
```

```
ex_1.py x
Q- Cc W * 0 results
71 # Получить текущую дату.
72 today = date.today()
73
74 # Разбить команду на части для выделения номера года.
75 parts = command.split(' ', maxsplit=1)
76 # Получить требуемый стаж.
77 period = int(parts[1])
78
79 # Инициализировать счетчик.
80 count = 0
81 # Проверить сведения работников из списка.
82 for worker in workers:
83     if today.year - worker.get('year', today.year) >= period:
84         count += 1
85     print(
86         '|{:>4}: {}'.format(count, worker.get('name', ''))
87     )
88
89 # Если счетчик равен 0, то работники не найдены.
90 if count == 0:
91     print("Работники с заданным стажем не найдены.")
92
93 elif command == 'help':
94     # Вывести справку о работе с программой.
95     print("Список команд:\n")
96     print("add - добавить работника;")
97     print("list - вывести список работников;")
98     print("select <стаж> - запросить работников со стажем;")
```

```
print("help - отобразить справку;")
print("exit - завершить работу с программой.")

else:
    print(f"Неизвестная команда {command}", file=sys.stderr)
```

Рисунок 9.3 – Код программы-примера

```
ex_1 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git\Py__L9\Р
>>> choose
>>> Неизвестная команда choose
add
Фамилия и инициалы? Kuvshin IA
Должность? Programmer
Год поступления? 2023
>>> add
Фамилия и инициалы? Mizin GE
Должность? Programmer
Год поступления? 2018
>>> select 4
1: Mizin GE
>>>
```

```
ex_1 x
>>> add
Фамилия и инициалы? Ivanov II
Должность? Administrator
Год поступления? 2015
>>> add
>>> Неизвестная команда add
add
Фамилия и инициалы? Novikov SI
Должность? Engineer
Год поступления? 2010
>>> select 4
1: Ivanov II
2: Novikov SI
>>> list
+-----+-----+-----+
| № |      Ф.И.О.      |      Должность      |      Год      |
+-----+-----+-----+
| 1 | Ivanov II         | Administrator        | 2015 |
| 2 | Kuvshin IA        | Programmer           | 2023 |
| 3 | Mizin GE          | Programmer           | 2018 |
| 4 | Novikov SI        | Engineer             | 2010 |
+-----+-----+-----+
>>> help
>>> Неизвестная команда help
help
Список команд:
add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку.
```

Рисунок 9.4 – Результат работы программы - примера

```
ex_1 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Пр
>>> choose
>>> Неизвестная команда choose
|
```

Рисунок 9.5 – Результат работы программы - примера
(ошибка)

9. 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.

10. Зафиксируйте сделанные изменения в репозитории.

```
ex_1.py x task_9.py x task_11.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      # Создать словарь.
6      school = {
7          '1A': 13,
8          '1B': 18,
9          # '1B': 5,
10         '2A': 18,
11         '2B': 19,
12         # '3A': 22,
13         # '3B': 11,
14         # '4A': 22,
15     }
16
17     # Организовать бесконечный цикл запроса команд.
18     while True:
19         # Запросить команду из терминала.
20         # На Python с помощью ANSI-код можно делать цвет, фон и т.д.
21         # \033[0m - сброс форматирования
22         command = input("\033[0mEnter command: ").lower()
23
24         # Выполнить действие в соответствие с командой.
25         if command == 'exit':
26             break
27
28         # Добавить новый класс
29         elif command == 'add':
30             while True:
31                 class_name = input("Enter class: ")
32                 if class_name == '':
33                     continue
34                 school[class_name] = 0
35                 break
36
37         elif command == 'del':
38             while True:
39                 class_name = input("Enter class: ")
40                 if class_name == '':
41                     continue
42                 if class_name in school:
43                     del school[class_name]
44                     break
45
46         elif command == 'update':
47             while True:
48                 class_name = input("Enter class: ")
49                 if class_name == '':
50                     continue
51                 if class_name in school:
52                     new_value = input("Enter new value: ")
53                     school[class_name] = int(new_value)
54                     break
55
56         else:
57             print("Unknown command")
58
59     # Вывести словарь
60     print(school)
61
62     # Вычислить общее количество учащихся
63     total_students = sum(school.values())
64     print("Total students: ", total_students)
65
66     # Завершить программу
67     print("Program finished")
68
69 if __name__ == '__main__':
70     main()
71
72 # EOF
```

```
ex_1.py x task_9.py x
31
32 # Добавить новый класс
33 elif command == 'add':
34     while True:
35         class_name = input("Enter class: ")
36         if class_name in school:
37             print(f"\033[31m{f'Класс {class_name} уже существует'}")
38         else:
39             class_num = int(input("Enter number of pupils: "))
40             school[class_name] = class_num
41             break
42
43 # Сортировка по ключу
44 school = dict(sorted(school.items()))
45
46 # Показать все классы
47 elif command == 'show':
48     # Заголовок таблицы.
49     line = f'{"+" + "-"*7 + "+" + "-"*32 + "+"}'
50     print(line)
51     print(f'|{'Class' :^7}|{'Number of pupils' :^32}|')
52     print(line)
53
54 # Вывести данные о всех классах.
55 for k, v in school.items():
56     print(f'|{k :^7}|{v :^32}|')
57     print(line)
58
59 # Изменить число учеников в классе
60 elif command == 'change':
61     while True:
```

```
ex_1.py x task_9.py x
58
59 # Изменить число учеников в классе
60 elif command == 'change':
61     while True:
62         class_name = input("\033[0mEnter class: ")
63         if class_name in school:
64             class_num = int(input("Enter number of pupils: "))
65             school[class_name] = class_num
66             break
67         else:
68             print(f"\033[31m{f'Класс {class_name} не найден'}")
69
70 # Удалить класс
71 elif command == 'remove':
72     while True:
73         class_name = input("\033[0mEnter class: ")
74         if class_name in school:
75             del school[class_name]
76             break
77         else:
78             # \033[31m - 31m - управляющий цвет(красный) код
79             print(f"\033[31m{f'Класс {class_name} не найден'}")
80
81 # Кол-во учеников в школе
82 elif command == 'sum':
83     for k, v in school.items():
84         # Просуммировать значения словаря
85         s = sum(school.values())
86     print(f"Общее кол-во учеников в школе: {s}")
87
88 # Список команд с описанием
89 if __name__ == '__main__':
90     while True:
```



```
ex_1.py x task_9.py x
58
59 # Изменить число учеников в классе
60 elif command == 'change':
61     while True:
62         class_name = input("\033[0mEnter class: ")
63         if class_name in school:
64             class_num = int(input("Enter number of pupils: "))
65             school[class_name] = class_num
66             break
67         else:
68             print(f"\033[31m{'Класс {class_name} не найден'}")
69
70 # Удалить класс
71 elif command == 'remove':
72     while True:
73         class_name = input("\033[0mEnter class: ")
74         if class_name in school:
75             del school[class_name]
76             break
77         else:
78             # \033[31m - 31m - управляющий цветом(красный) код
79             print(f"\033[31m{'Класс {class_name} не найден'}")
80
81 # Кол-во учеников в школе
82 elif command == 'sum':
83     for k, v in school.items():
84         # Просуммировать значения словаря
85         s = sum(school.values())
86         print(f"Общее кол-во учеников в школе: {s}")
87
88 # Список команд с описанием
if __name__ == '__main__': while True
```

```
87
88 # Список команд с описанием
89 elif command == 'help':
90     # Вывести справку о работе с программой.
91     print("Список команд:\n")
92     print("change - изменить число обучающихся;")
93     print("add - добавить новый класс;")
94     print("remove - расформировать класс;")
95     print("sum - общее кол-во учащихся;")
96     print("show - показать список классов;")
97     print("help - отобразить справку;")
98     print("exit - завершить работу с программой.")
99
100 # Неопознанная команда
101 else:
102     print("\033[31mНеизвестная команда")
103
```

Рисунок 9.6 – Код программы

```
ex_1.py x task_9.py x
Run: task_9 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git\Py_L9\PyCha
Enter command: show
+-----+-----+
| Class |      Number of pupils      |
+-----+-----+
| 1A  |      13      |
| 1B  |      18      |
| 2A  |      18      |
| 2B  |      19      |
+-----+-----+
Enter command: sum
Общее кол-во учеников в школе: 68
Enter command: remov
Неизвестная команда
Enter command: remove
Enter class: 2B
Enter command: change
Enter class: 2B
Класс 2B не найден
Enter class: 2A
Enter number of pupils: 24
Enter command: add
Enter class: 1B
Enter number of pupils: 9
Enter command: show
+-----+-----+
| Class |      Number of pupils      |
+-----+-----+
| 1A  |      13      |
| 1B  |      18      |
```

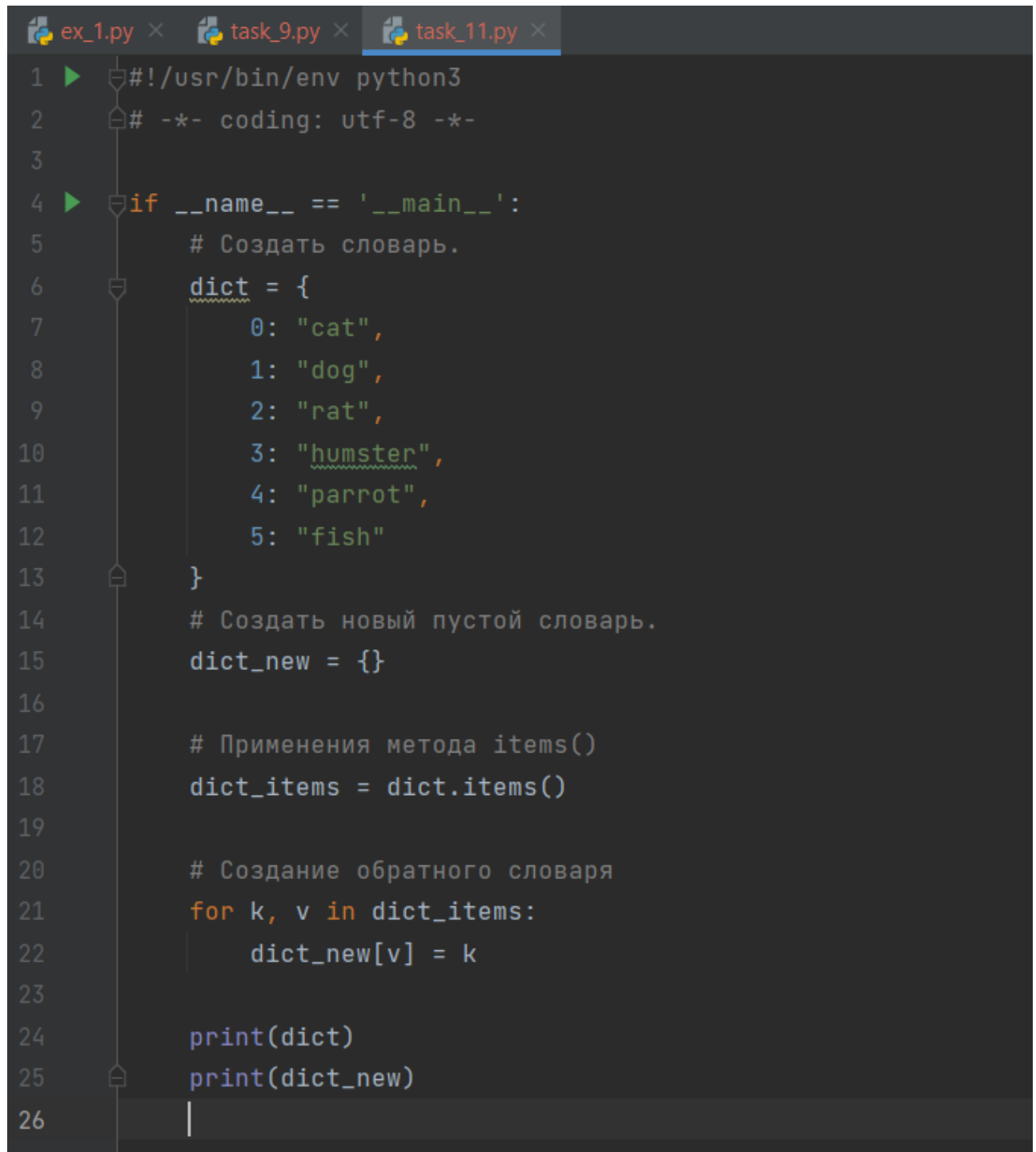
```
Enter command: add
Enter class: 1B
Enter number of pupils: 9
Enter command: show
+-----+-----+
| Class |      Number of pupils      |
+-----+-----+
| 1A  |      13      |
| 1B  |      18      |
| 1B  |      9       |
| 2A  |      24      |
+-----+-----+
Enter command: sum
Общее кол-во учеников в школе: 64
Enter command: exit

Process finished with exit code 0
```

Рисунок 9.7 – Результат работы программы

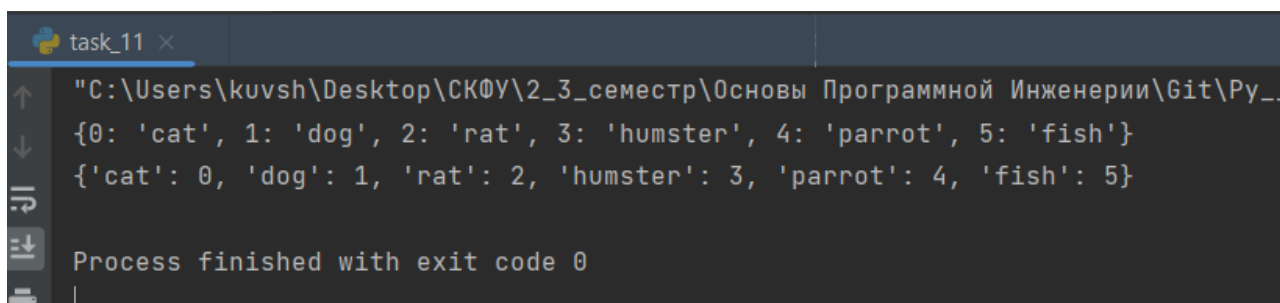
11. Решите задачу: создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод `items()`, с помощью полученного объекта `dict_items` создайте новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.

12. Зафиксируйте сделанные изменения в репозитории.



```
1  ▶ #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  ▶ if __name__ == '__main__':
5      # Создать словарь.
6      dict = {
7          0: "cat",
8          1: "dog",
9          2: "rat",
10         3: "hamster",
11         4: "parrot",
12         5: "fish"
13     }
14     # Создать новый пустой словарь.
15     dict_new = {}
16
17     # Применения метода items()
18     dict_items = dict.items()
19
20     # Создание обратного словаря
21     for k, v in dict_items:
22         dict_new[v] = k
23
24     print(dict)
25     print(dict_new)
26
```

Рисунок 9.8 – Код программы



```
task_11 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git\Py_...
{0: 'cat', 1: 'dog', 2: 'rat', 3: 'hamster', 4: 'parrot', 5: 'fish'}
{'cat': 0, 'dog': 1, 'rat': 2, 'hamster': 3, 'parrot': 4, 'fish': 5}
Process finished with exit code 0
```

Рисунок 9.9 – Результат работы программы

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

14. Зафиксируйте сделанные изменения в репозитории.

15. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

16. Выполните слияние ветки для разработки с веткой main/master.

17. Отправьте сделанные изменения на сервер GitHub.

18. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Индивидуальные задания: (Вариант 15)

Составить программу с использованием списков и словарей для решения задачи. Номер варианта определяется по согласованию с преподавателем.

15. Использовать словарь, содержащий следующие ключи:

фамилия, имя;

знак Зодиака;

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 упорядочены по датам рождения;

вывод на экран информации о людях, родившихся под знаком, название которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение.

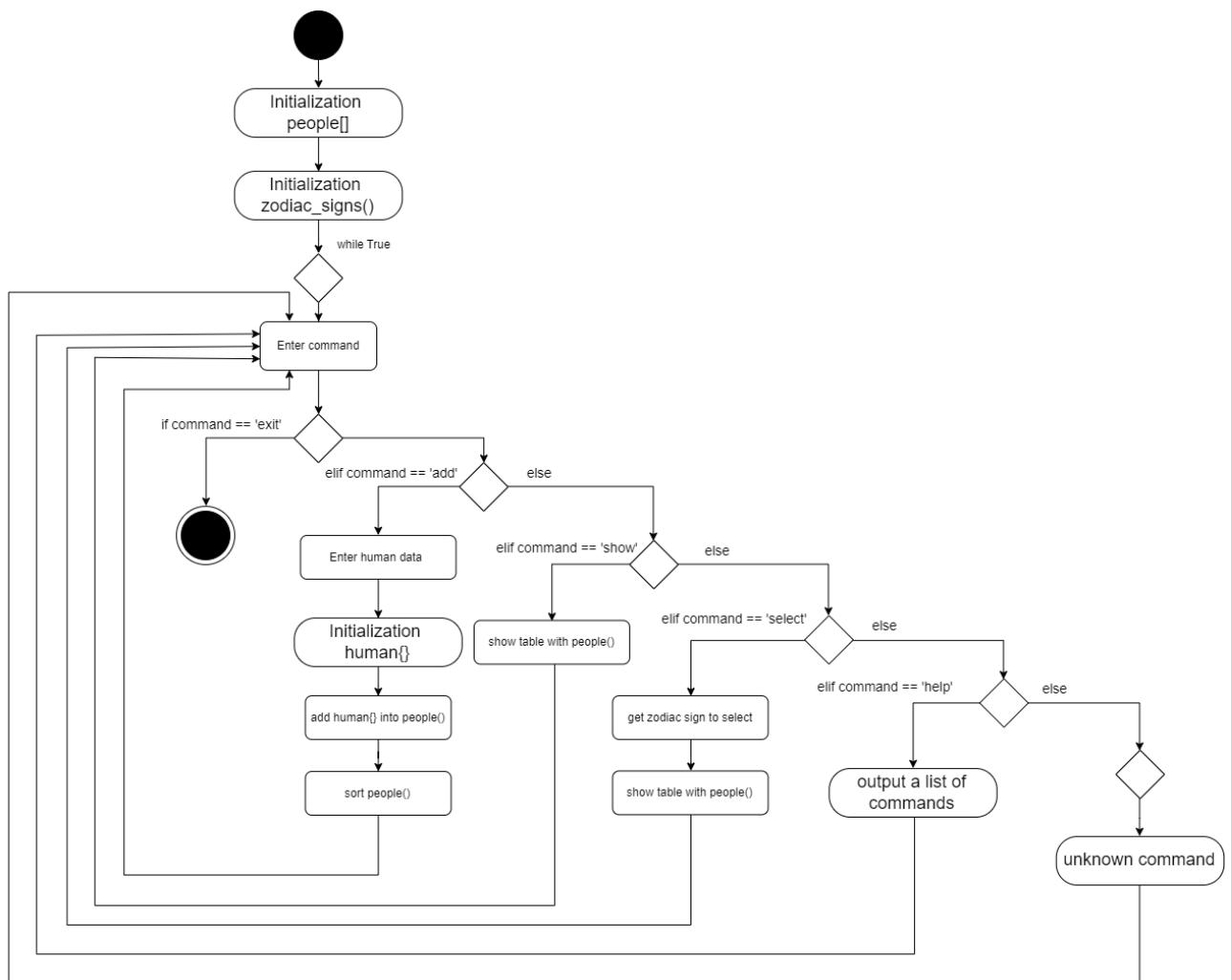


Рисунок 9.10 – UML-диаграмма программы

```

task_11.py x ex_1.py x task_9.py x ind_1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import datetime
5
6  if __name__ == '__main__':
7      # Список людей
8      people = []
9
10     # Кортеж 33
11     zodiac_signs = (
12         "овен",
13         "телец",
14         "близнецы",
15         "рак",
16         "лев",
17         "дева",
18         "весы",
19         "скорпион",
20         "стрелец",
21         "козерог",
22         "водолей",
23         "рыбы"
24     )
25
26     # Организовать бесконечный цикл запроса команд.
27     while True:
28         # Запросить команду из терминала.
29         # На Python с помощью ANSI-код можно делать цвет, фон и т.д.
30         # \033[0m - сброс форматирования
31         command = input("\033[0mEnter command: ") \
  
```

```
task_11.py x ex_1.py x task_9.py x ind_1.py x
25
26 # Организовать бесконечный цикл запроса команд.
27 while True:
28     # Запросить команду из терминала.
29     # На Python с помощью ANSI-код можно делать цвет, фон и т.д.
30     # \033[0m - сброс форматирования
31     command = input("\033[0mEnter command: ").lower()
32
33     # Выполнить действие в соответствие с командой.
34     if command == 'exit':
35         break
36
37     # Добавить нового человека
38     elif command == 'add':
39         # Запросить данные человека
40         name = input("Enter name: ")
41         # Проверка 33
42         while True:
43             zodiac_sign = input("\033[0mEnter zodiac sign: ").lower()
44             if zodiac_sign not in zodiac_signs:
45                 print("\033[31mIncorrect zodiac sign")
46             else:
47                 break
48         # Проверка Даты Рождения
49         while True:
50             birth = input("\033[0mEnter date of birth (yyyy.mm.dd): ")
51             # Обработка исключения (проверка правильности даты)
52             try:
53                 datetime.datetime.strptime(birth, '%Y.%m.%d')
54             except Exception:
55                 print("\033[31mIncorrect data format")
```

```
task_11.py x ex_1.py x task_9.py x ind_1.py x
58
59 # Создать словарь
60 human = {
61     'name': name,
62     'zodiac_sign': zodiac_sign,
63     'birth': birth,
64 }
65
66 # Добавить словарь в список.
67 people.append(human)
68
69 # Отсортировать список в случае необходимости.
70 if len(people) > 1:
71     people.sort(key=lambda item: item.get('birth', ''))
72
73 elif command == 'show':
74     # Заголовок таблицы.
75     line = (f'{"+" + "-" * 12 + "+" + "-" * 12 + "+"}'
76            f'{"-" * 15 + "+"}')
77     print(line)
78     print(f"|{'Name' :^12}|{'Birth' :^12}|{'Zodiac_sign' :^15}|")
79     print(line)
80
81     # Вывести данные о всех людях.
82     for idx, human in enumerate(people):...
83
84     # .startswith - Поиск строк с заданным началом строки
85     elif command.startswith('select'):
86         # Разбить команду на части для выделения номера года.
87         # Параметр maxsplit - int, сколько раз делить строку.
88         while True:
```

```
task_11.py x ex_1.py x task_9.py x ind_1.py x
94 while True:
95     parts = command.split(' ', maxsplit=1)
96     if len(parts) == 1:
97         command = input("\033[31mEnter command "
98                         "select and Zodiac_sign: ").lower()
99         # Получить требуемый ЗЗ.
100     else:
101         break
102
103     # Инициализировать счетчик.
104     count = 0
105     zz = parts[1]
106     # Заголовок таблицы.
107     line = (f'\033[0m{"+" + "-" * 12 + "+" + "-" * 12 + "+"}'
108            f'{"-" * 15 + "+"}')
109     print(line)
110     print(
111         f'|{'Name' : ^12}|{'Birth' : ^12}|'
112         f'|{'Zodiac_sign' : ^15}|'
113     )
114     print(line)
115     # Таблица с людьми
116     for p in people:
117         if zz == p.get('zodiac_sign'):
118             count += 1
119             print(
120                 f'|{p.get("name", "") : ^12}|'
121                 f'|{p.get("birth", "") : ^12}|'
122                 f'|{p.get("zodiac_sign", "") : ^15}|'
123             )
124             print(line)
125
126 if __name__ == '__main__':
127     Run Python Packages TODO Python Console Problems Terminal Services
```

```
task_11.py x ex_1.py x task_9.py x ind_1.py x
114 print(line)
115 # Таблица с людьми
116 for p in people:
117     if zz == p.get('zodiac_sign'):
118         count += 1
119         print(
120             f'|{p.get("name", "") : ^12}|'
121             f'|{p.get("birth", "") : ^12}|'
122             f'|{p.get("zodiac_sign", "") : ^15}|'
123         )
124         print(line)
125
126 # Если счетчик равен 0, то люди не найдены.
127 if count == 0:
128     print("Люди с заданным ЗЗ не найдены")
129
130 # Список команд с описанием
131 elif command == 'help':
132     # Вывести справку о работе с программой.
133     print("\033[32mСписок команд:\n")
134     print("add - добавить человека;")
135     print("select <ЗЗ> - запросить людей с определенным ЗЗ;")
136     print("show - показать список людей;")
137     print("help - отобразить справку;")
138     print("exit - завершить работу с программой.")
139
140 # Неопознанная команда
141 else:
142     print("\033[31mНеизвестная команда")
143
144 if __name__ == '__main__':
```

Рисунок 9.11 – Код программы

```
ind_1 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git\Py_L9\PyCharm\venv\Scripts\python.exe
Enter command: add
Enter name: Иванов И.
Enter zodiac sign: рак
Enter date of birth (yyyy.mm.dd): 2005.12.12
Enter command: Петров В.
Неизвестная команда
Enter command: add
Enter name: Петров И.
Enter zodiac sign: лев
Enter date of birth (yyyy.mm.dd): 2001.12.06
Enter command: add
Enter name: Радченко И.
Enter zodiac sign: лев
Enter date of birth (yyyy.mm.dd): 1995.06.15
Enter command: show
+-----+-----+-----+
| Name | Birth | Zodiac_sign |
+-----+-----+-----+
| Радченко И. | 1995.06.15 | лев |
+-----+-----+-----+
| Петров И. | 2001.12.06 | лев |
+-----+-----+-----+
| Иванов И. | 2005.12.12 | рак |
+-----+-----+-----+
Enter command: select
```

```
Enter command select and Zodiac_sign: select рак
+-----+-----+-----+
| Name | Birth | Zodiac_sign |
+-----+-----+-----+
| Иванов И. | 2005.12.12 | рак |
+-----+-----+-----+
Enter command: select лев
+-----+-----+-----+
| Name | Birth | Zodiac_sign |
+-----+-----+-----+
| Радченко И. | 1995.06.15 | лев |
+-----+-----+-----+
| Петров И. | 2001.12.06 | лев |
+-----+-----+-----+
Enter command: exit

Process finished with exit code 0
```

Рисунок 9.12 – Результат работы программы

Вопросы для защиты работы:

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?

Функция len() широко используется для определения размера объектов в Python. В нашем случае передача объекта словаря этой функции вернет размер словаря, то есть количество пар ключ-значение, присутствующих в словаре.

3. Какие методы обхода словарей Вам известны?

Элементы словаря перебираются в цикле for также, как элементы других сложных объектов. Однако "по-умолчанию" извлекаются только ключи.

С другой стороны у словаря как класса есть метод items(), который создает особую структуру, состоящую из кортежей. Каждый кортеж включает ключ и значение.

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?

```
>>> for i in nums:
```

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

```
...
```

```
one
```

```
two
```

```
three
```

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

С помощью метода `setdefault()`, при непосредственном обращении к ключу словарю.

6. Что такое словарь включений?

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

Основной пример:

```
>>> {x: x * x for x in (1, 2, 3, 4)}  
{1: 1, 2: 4, 3: 9, 4: 16}
```

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. Функция `zip()` принимает итерируемый объект, например, список, кортеж, множество или словарь в качестве аргумента. Затем она генерирует список кортежей, которые содержат элементы из каждого объекта, переданного в функцию. Предположим, что есть список имен и номером сотрудников, и их нужно объединить в массив кортежей. Для этого можно использовать функцию `zip()`.

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. Поддерживается и стандартный способ представления времени, однако больший упор сделан на простоту манипулирования датой, временем и их частями.