

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

**Кафедра  
инфокоммуникаций  
Институт цифрового  
развития**

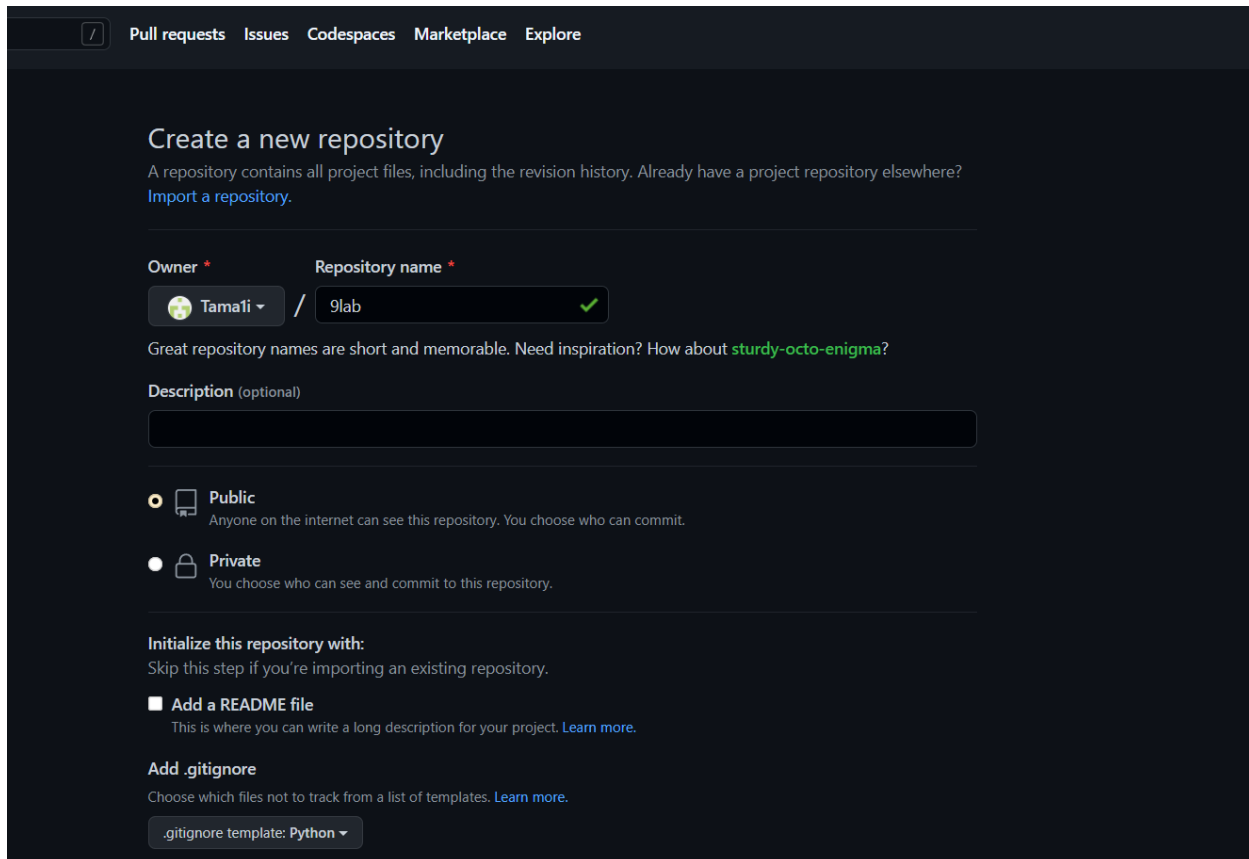
**ОТЧЁТ**  
**по лабораторной работе №9**  
Дисциплина: «Основы программной инженерии»  
Тема: «Работа со словарями в языке Python»

Выполнил: студент 2  
курса группы Пиж-б-о-  
21-1  
Рязанцев Матвей  
Денисович

Ставрополь 2022

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

## Выполнение работы



1 Pull requests Issues Codespaces Marketplace Explore

### 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 \*

Tama1i / 9lab ✓

Great repository names are short and memorable. Need inspiration? How about [sturdy-octo-enigma?](#)

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:  
Skip this step if you're importing an existing repository.

☒ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python

Рисунок 1 -создание репозитория

```
D:\gite>git flow init -f

which branch should be used for bringing forth production releases?
- develop
- master
Branch name for production releases: [master]

which branch should be used for integration of the "next release"?
- develop
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/gite/.git/hooks]

D:\gite>
```

Рисунок 2 - Организация репозитория по модели ветвления git flow

Код общего задания:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
from datetime import date
if __name__ == '__main__':
    # Список работников.
    workers = []

    # 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 работнике.
            name = input("Фамилия и инициалы? ")
            post = input("Должность? ")
            year = int(input("Год поступления? "))

            # Создать словарь.
            worker = {
                'name': name,
                'post': post,
                'year': year,
            }

            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))

        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 8
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                    "№",
                    "Ф.И.О.",
                    "Должность",
                    "Год"
                )
            )
            print(line)

            # Вывести данные о всех сотрудниках.
            for idx, worker in enumerate(workers, 1):
                print(
                    '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                        idx,
                        worker.get('name', ''),
                        worker.get('post', ''),
                        worker.get('year', 0)
                    )
                )
```

```

    )
    )

print(line)

elif command == 'select':
    # Получить текущую дату.
    today = date.today()

    # Разбить команду на части для выделения номера года.
    parts = command.split(' ', maxsplit=1)
    # Получить требуемый стаж.
    period = int(parts[1])

    # Инициализировать счетчик.
    count = 0
    # Проверить сведения работников из списка.
    for worker in workers:
        if today.year - worker.get('year', today.year) >= period:
            count += 1
            print(
                '{:>4}: {}'.format(count, worker.get('name', ''))
            )

    # Если счетчик равен 0, то работники не найдены.
    if count == 0:
        print("Работники с заданным стажем не найдены.")
elif command == 'help':
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить работника;")
    print("list - вывести список работников;")
    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)

```

Код программы идз 1:

```

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

if __name__ == '__main__':
    school = {
        "1А": 32,
        "1Б": 25,
        "2А": 22,
        "2Б": 31,
        "3А": 32,
        "3Б": 18,
    }

    for key, value in school.items():
        print(f" В {key} классе количество детей = {value}.")

    # Часть а)

```

```

print("\nВ одном из классов поменялось количество детей, теперь:\n")
school['2Б'] = 16
for key, value in school.items():
    print(f" В {key} классе количество детей = {value}.")

# Часть 6)
print("\nПоявился новый класс, теперь:\n")
school.setdefault("3В", 31)
for key, value in school.items():
    print(f" В {key} классе количество детей = {value}.")

# Часть с)
print("\nРасформировали один класс, теперь:\n")
school.pop("1А")
for key, value in school.items():
    print(f" В {key} классе количество детей = {value}.")

count = 0
for value in school.values():
    count += value
print(f"\nВсего учеников в школе - {count}")

```

```

C:\Users\ryaza\AppData\Local\Programs\Python\Python310\python.exe
В одном из классов поменялось количество детей, теперь:

В 1А классе количество детей = 32.
В 1Б классе количество детей = 25.
В 2А классе количество детей = 22.
В 2Б классе количество детей = 16.
В 3А классе количество детей = 32.
В 3Б классе количество детей = 18.

Появился новый класс, теперь:

В 1А классе количество детей = 32.
В 1Б классе количество детей = 25.
В 2А классе количество детей = 22.
В 2Б классе количество детей = 16.
В 3А классе количество детей = 32.
В 3Б классе количество детей = 18.
В 3В классе количество детей = 31.

Расформировали один класс, теперь:

В 1Б классе количество детей = 25.
В 2А классе количество детей = 22.
В 2Б классе количество детей = 16.
В 3А классе количество детей = 32.
В 3Б классе количество детей = 18.
В 3В классе количество детей = 31.

Всего учеников в школе - 144
Press any key to continue . . .

```

Рисунок 11 – результат работы программы

Код программы идз 2:

```

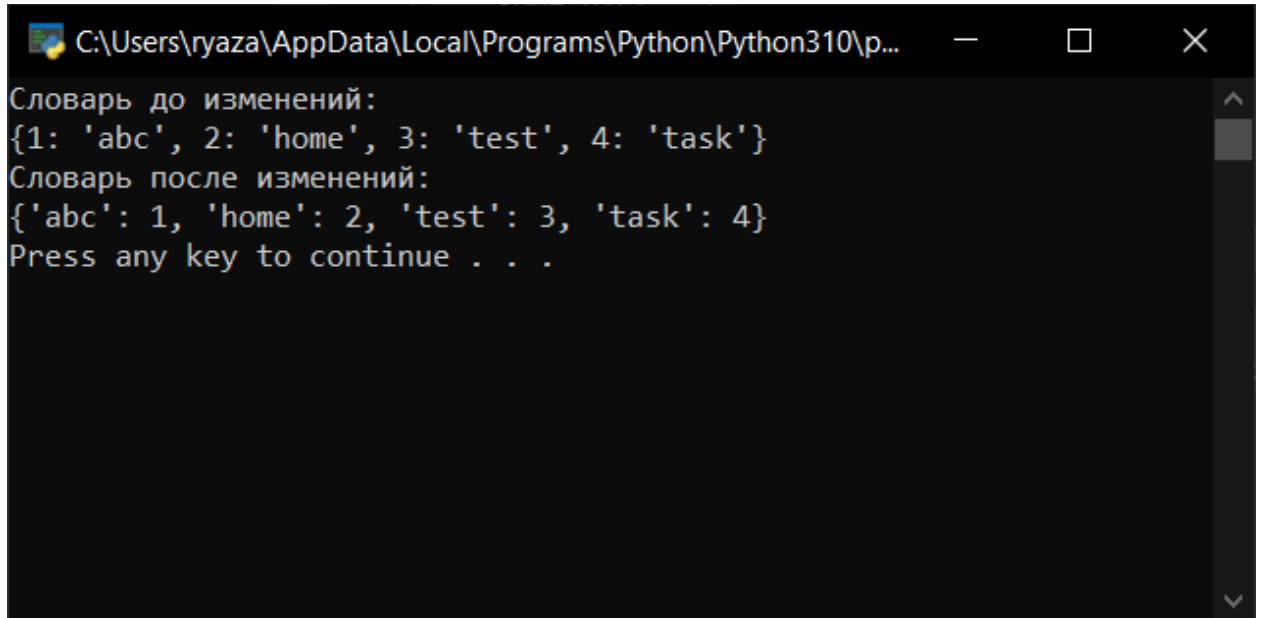
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    some_dict = {
        1: "abc",
        2: "home",
        3: "test",
    }

```

```
4: "task"  
}  
print(f"Словарь до изменений:\n{some_dict}")  
dict_items = some_dict.items()  
changed_dict = {i: j for j, i in dict_items}  
print(f"Словарь после изменений:\n{changed_dict}")
```

Рисунок 12 – код программы идз 2



```
C:\Users\ryaza\AppData\Local\Programs\Python\Python310\p...  
Словарь до изменений:  
{1: 'abc', 2: 'home', 3: 'test', 4: 'task'}  
Словарь после изменений:  
{'abc': 1, 'home': 2, 'test': 3, 'task': 4}  
Press any key to continue . . .
```

Рисунок 13 – результат работы программы

Индивидуальное задание

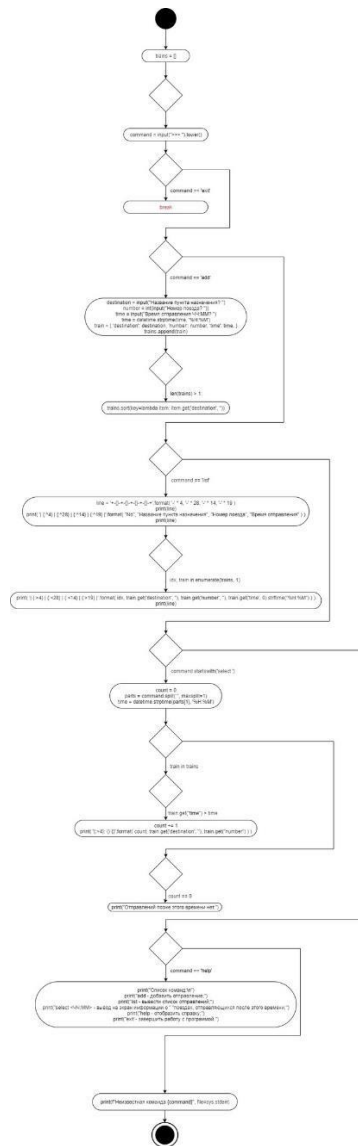


Рисунок 14 – uml диаграмма

Код программы:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
from datetime import date
if __name__ == '__main__':
    # Список работников.
    rep = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствии с командой.
        if command == 'exit':
            break

        elif command == 'add':
            # Запросить данные о работнике.

```

```

name = input("name faname? ")
num = int(input("number? "))
br = int(input("burftday? "))

# Создать словарь.
chel = {
    'name': name,
    'num': num,
    'br': br,
}

# Добавить словарь в список.
pep.append(chel)
# Отсортировать список в случае необходимости.
if len(pep) > 1:
    pep.sort(key=lambda item: item.get('br', ''))

elif command == 'list':
# Заголовок таблицы.
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20,
        '-' * 8
    )
    print(line)
    print(
        '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
            "№",
            "F.I.O.",
            "NUMBER",
            "BRDAY"
        )
    )
    print(line)

# Вывести данные о всех сотрудниках.
for idx, chel in enumerate(pep, 1):
    print(
        '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
            idx,
            chel.get('name', ''),
            chel.get('num', ''),
            chel.get('br', 0)
        )
    )

    print(line)

elif command == 'select':

# Получить требуемый стаж.
zapro = int(input("zapro po numeru "))

# Инициализировать счетчик.
count = 0
# Проверить сведения работников из списка.
for chel in pep:
    if chel.get('num') == zapros:
        count += 1
        print(
            '{:>4}: {}'.format(count, chel.get('name', ''))
        )

# Если счетчик равен 0, то работники не найдены.

```



```

if count == 0:
    print("cheela s takim nomerom net")
elif command == 'help':
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - add чел;")
    print("list - show list of pep;")
    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)

```

```

C:\Users\ryaza\AppData\Local\Programs\Python\Python310\python.exe
>>> add
name faname? mat
number? 89624409324
brday? 22122003
>>> add
name faname? mei
number? 89884417437
brday? 10112003
>>> add
name faname? yan
number? 89687
brday? 11
>>> list
+-----+-----+-----+-----+
| № | F.I.O. | NUMBER | BRDAY |
+-----+-----+-----+-----+
| 1 | yan | 89687 | 11 |
| 2 | mei | 89884417437 | 10112003 |
| 3 | mat | 89624409324 | 22122003 |
+-----+-----+-----+-----+
>>> select
zapros po numeru 89624409324
1: mat
>>>

```

Рисунок 15 – результат работы программы идз

## Контрольные вопросы

1. Что такое словари в языке Python?

Словари в Python – это изменяемые отображения ссылок на объекты, доступные по ключу.

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

Функция `len()` возвращает длину (количество элементов) в объекте. Аргумент может быть последовательностью, такой как строка, байты, кортеж, список или диапазон или коллекцией (такой как словарь, множество или неизменяемое множество).

### 3. Какие методы обхода словарей Вам известны?

Самый очевидный вариант обхода словаря — это попытаться напрямую запустить цикл `for` по объекту словаря, так же как мы делаем это со списками, кортежами, строками и любыми другими итерируемыми объектами. `for something in currencies: print(something)`

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

С помощью метода `.get()`

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

С помощью функции `dict.update()`

### 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. Предположим, что есть список имен и номером сотрудников, и их нужно объединить в массив кортежей. Для этого можно использовать функцию `zip()`. Вот пример программы, которая делает именно это:

```
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

Как получить текущие дату и время?

```
import datetime  
  
dt_now = datetime.datetime.now()  
  
print(dt_now)
```

Результат:

```
2022-09-11 15:43:32.249588
```

Получить текущую дату:

```
from datetime import date  
  
current_date = date.today()  
  
print(current_date)
```

Результат:

```
2022-09-11
```

Получить текущее время:

```
import datetime
```

```
current_date_time = datetime.datetime.now()  
current_time = current_date_time.time()  
print(current_time)
```

Результат:

15:51:05.627643