

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
«Работа со словарями в языке Python»**

**Отчет по лабораторной работе № 2.6
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2022г.

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.

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT 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.](#)

Owner * **Repository name ***

Yana-Kh / Lab-9-OPJ ✓

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

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 ▼

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▼

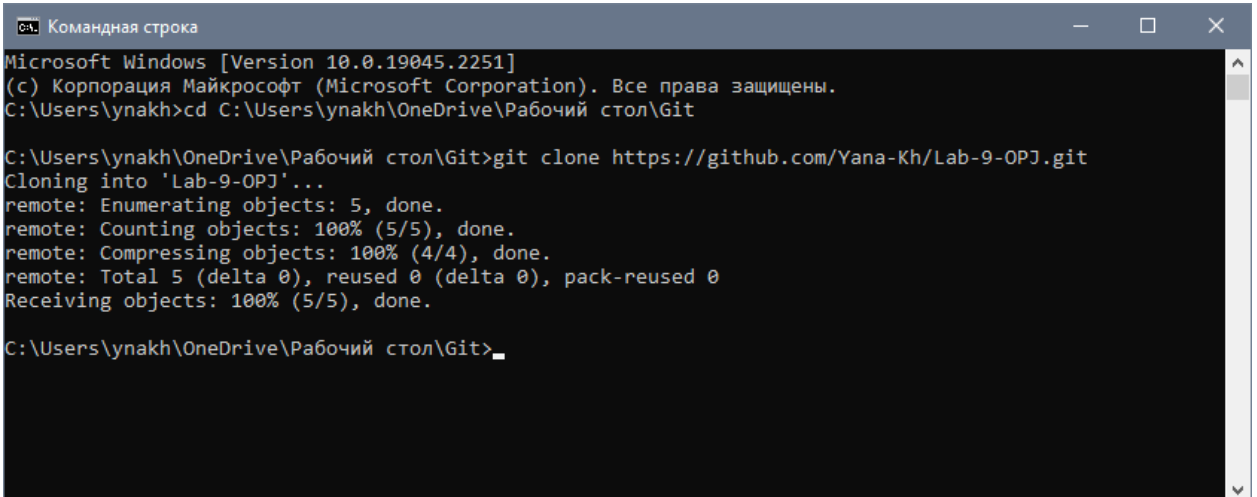
This will set main as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

Creating repository...

Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.



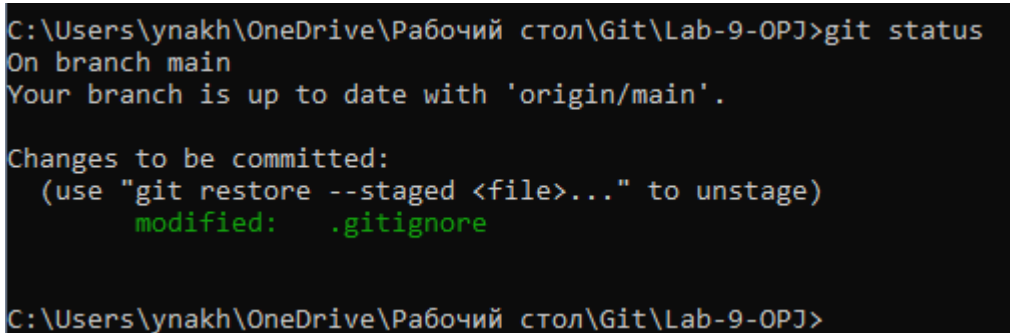
```
cmd
Microsoft Windows [Version 10.0.19045.2251]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
C:\Users\ynakh>cd C:\Users\ynakh\OneDrive\Рабочий стол\Git

C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/Yana-Kh/Lab-9-OPJ.git
Cloning into 'Lab-9-OPJ'...
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\ynakh\OneDrive\Рабочий стол\Git>
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.



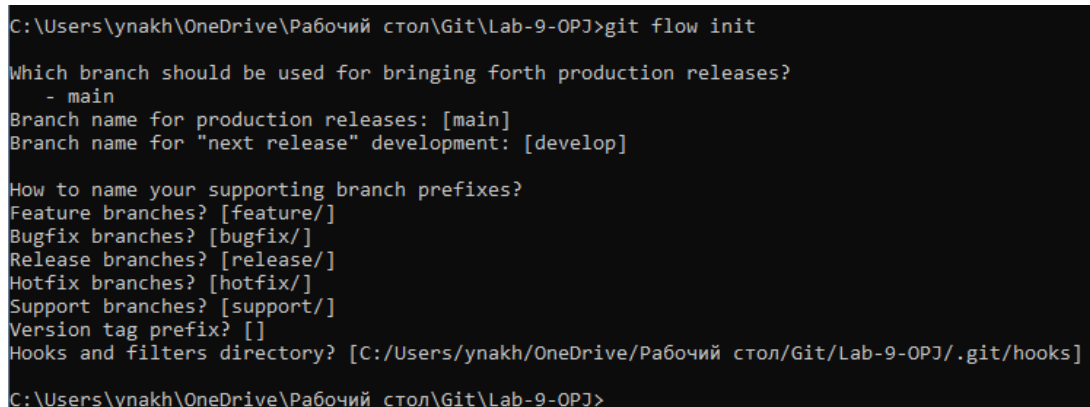
```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-9-OPJ>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-9-OPJ>
```

Рисунок 3 – Дополнение файла .gitignore

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



```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-9-OPJ>git flow init
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
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? [C:/Users/ynakh/OneDrive/Рабочий стол/Git/Lab-9-OPJ/.git/hooks]

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-9-OPJ>
```

Рисунок 4 – Организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

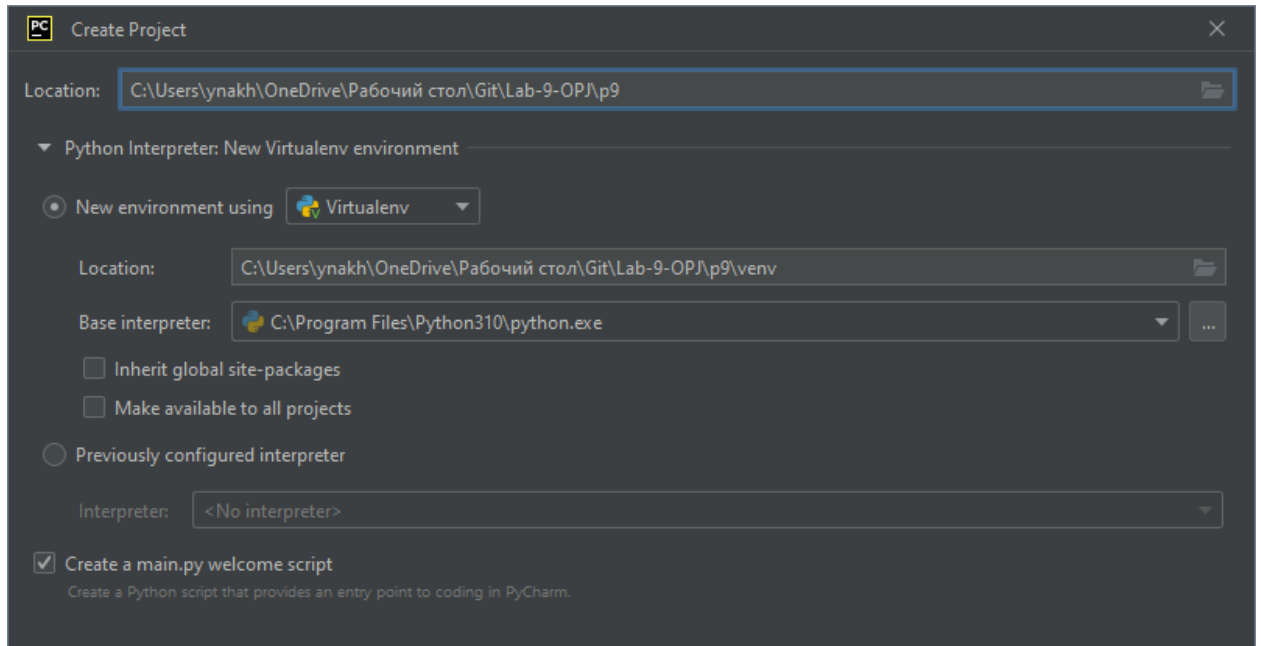


Рисунок 5 – Создание проекта PyCharm в папке репозитория

7. Проработайте пример лабораторной работы. Создайте для него отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

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.

Код:

```
#!/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.startswith('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)

```

8. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных, вводимых с клавиатуры.

```
C:\Windows\py.exe
>>> add
Фамилия и инициалы? Поляков ЕМ
Должность? программист
Год поступления? 2017
>>> add
Фамилия и инициалы? Петров
Должность? директор
Год поступления? 2004
>>> add
Фамилия и инициалы? Смирнов НД
Должность? менеджер
Год поступления? 2013
>>> add
Фамилия и инициалы? Романов ПР
Должность? бухгалтер
Год поступления? 1999
>>> add
Фамилия и инициалы? Дмитриев ПЕ
Должность? программист
Год поступления? 2008
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Дмитриев ПЕ | программист | 2008 |
| 2 | Петров | директор | 2004 |
| 3 | Поляков ЕМ | программист | 2017 |
| 4 | Романов ПР | бухгалтер | 1999 |
| 5 | Смирнов НД | менеджер | 2013 |
+-----+-----+-----+-----+
>>> select 10
1: Дмитриев ПЕ
2: Петров
3: Романов ПР
>>> select 30
Работники с заданным стажем не найдены.
>>> _
```

Рисунок 6 – Результат работы программы

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.

Код:

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

```

import sys

if __name__ == '__main__':
    school = {}
    # 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 == 'mod':
            # Запросить данные об изменениях в классе.
            cl = input("Enter class: ")
            students = int(input("Enter number of students: "))

            # Проверка на существование
            if cl in school.keys():
                # Изменяем запись
                school[cl] = students
            else:
                print("Введен несуществующий класс")

        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+'.format(
                '-' * 8,
                '-' * 25
            )
            print(line)
            print(
                '| {:^8} | {:^25} |'.format(
                    "Класс",
                    "Количество учащихся"
                )
            )
            print(line)

            # Вывести данные о всех сотрудниках.
            for cl, std in school.items():
                print(
                    '| {:<8} | {:<25} |'.format(
                        cl,
                        std
                    )
                )

            print(line)

        elif command == 'new':
            # Запросить данные о новом классе.
            cl = input("Enter class: ")
            students = int(input("Enter number of students: "))

            # Добавить запись
            school.setdefault(cl, students)

        elif command == 'count':
            # Подсчитаем общее количество учеников
            all_std = sum(s for s in school.values())
            print(f'Total number of students: {all_std}')

```



```

elif command == 'del':
    # Запросить данные о классе.
    cl = input("Enter class: ")
    # Удалить запись
    del school[cl]

elif command == 'help':
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("new - создать запись о новом классе;")
    print("mod - изменить сведения о кол-ве обучающихся;")
    print("del - удалить класс;")
    print("list - вывести список классов;")
    print("count - вывести общее количество учащихся;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

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

```

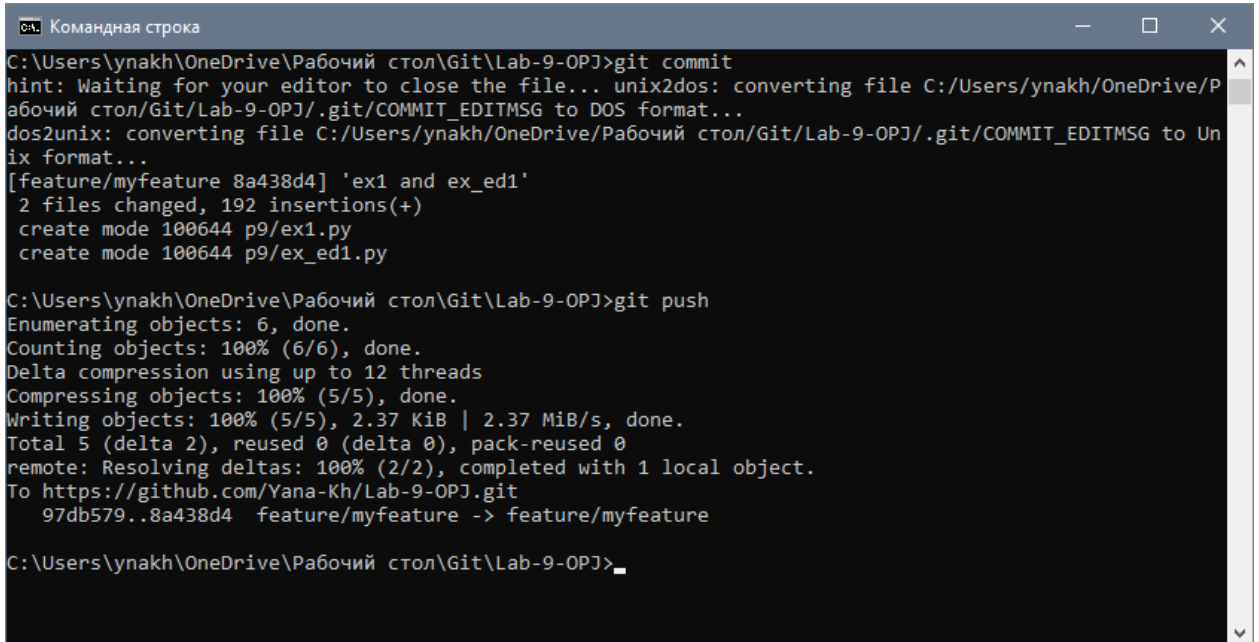
```

>>> new
Enter class: 2a
Enter number of students: 34
>>> new
Enter class: 8b
Enter number of students: 13
>>> new
Enter class: 9c
Enter number of students: 21
>>> new
Enter class: 11a
Enter number of students: 28
>>> list
+-----+-----+
|  Класс  |  Количество учащихся  |
+-----+-----+
|  2a     |  34                   |
|  8b     |  13                   |
|  9c     |  21                   |
|  11a    |  28                   |
+-----+-----+
>>> mod
Enter class: 11a
Enter number of students: 18
>>> list
+-----+-----+
|  Класс  |  Количество учащихся  |
+-----+-----+
|  2a     |  34                   |
|  8b     |  13                   |
|  9c     |  21                   |
|  11a    |  18                   |
+-----+-----+
>>> del
Enter class: 2a
>>> list
+-----+-----+
|  Класс  |  Количество учащихся  |
+-----+-----+
|  8b     |  13                   |
|  9c     |  21                   |
|  11a    |  18                   |
+-----+-----+
>>>

```

Рисунок 7 – Результат работы программы

10. Зафиксируйте сделанные изменения в репозитории.



```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-9-OPJ>git commit
hint: Waiting for your editor to close the file... unix2dos: converting file C:/Users/ynakh/OneDrive/Рабочий стол\Git\Lab-9-OPJ/.git/COMMIT_EDITMSG to DOS format...
dos2unix: converting file C:/Users/ynakh/OneDrive\Рабочий стол\Git\Lab-9-OPJ/.git/COMMIT_EDITMSG to Unix format...
[feature/myfeature 8a438d4] 'ex1 and ex_ed1'
 2 files changed, 192 insertions(+)
 create mode 100644 p9/ex1.py
 create mode 100644 p9/ex_ed1.py

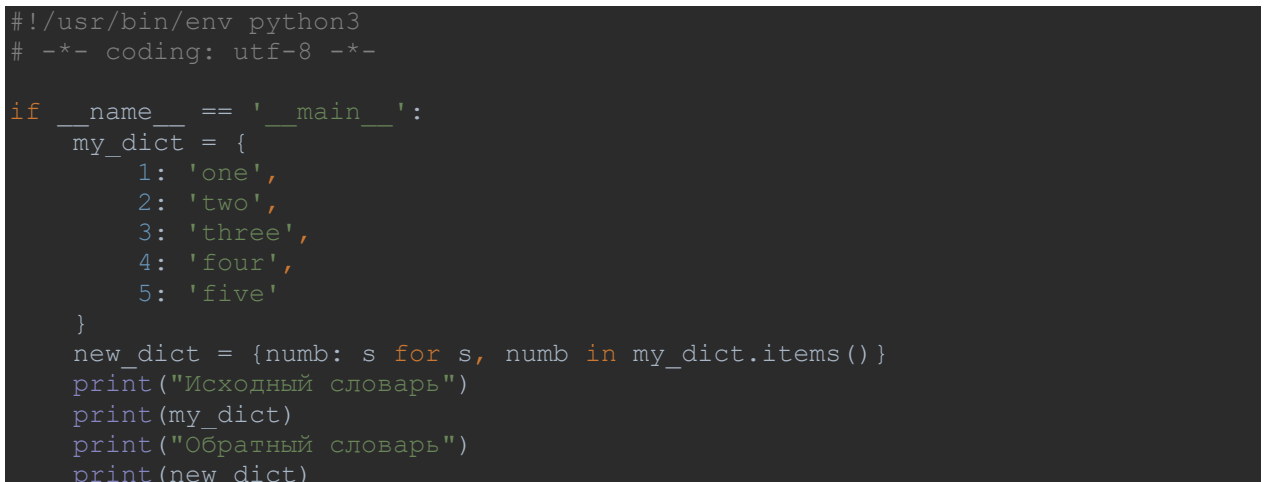
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-9-OPJ>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.37 KiB | 2.37 MiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/Yana-Kh/Lab-9-OPJ.git
 97db579..8a438d4  feature/myfeature -> feature/myfeature

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-9-OPJ>
```

Рисунок 8 – Фиксирование изменений в репозитории

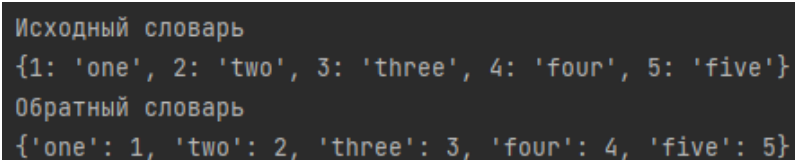
11. Решите задачу: создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод `items()`, с помощью полученного объекта `dict_items` создайте новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.

Код:



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

if __name__ == '__main__':
    my_dict = {
        1: 'one',
        2: 'two',
        3: 'three',
        4: 'four',
        5: 'five'
    }
    new_dict = {numb: s for s, numb in my_dict.items()}
    print("Исходный словарь")
    print(my_dict)
    print("Обратный словарь")
    print(new_dict)
```



```
Исходный словарь
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
Обратный словарь
{'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}
```

Рисунок 9 – Результат работы программы

12. Зафиксируйте сделанные изменения в репозитории.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-9-OPJ>git commit
hint: Waiting for your editor to close the file... unix2dos: converting file C:/Users/ynakh/OneDrive/Рабочий стол\Git\Lab-9-OPJ/.git/COMMIT_EDITMSG to DOS format...
dos2unix: converting file C:/Users/ynakh/OneDrive\Рабочий стол\Git\Lab-9-OPJ/.git/COMMIT_EDITMSG to Unix format...
[feature/myfeature 53406a0] ex_ed2
1 file changed, 18 insertions(+)
create mode 100644 p9/ex_ed2.py

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-9-OPJ>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 582 bytes | 582.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Yana-Kh/Lab-9-OPJ.git
8a438d4..53406a0 feature/myfeature -> feature/myfeature

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-9-OPJ>
```

Рисунок 10 – Фиксирование изменений в репозитории

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

Вариант 13(32). 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 упорядочены по трем первым цифрам номера телефона; вывод на экран информации о человеке, чья фамилия введена с клавиатуры; если такого нет, выдать на дисплей соответствующее сообщение.

Код:

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

import datetime
import sys

if __name__ == '__main__':
    # Список работников.
    peopls = []

    # 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 человеке.
        name = input("Фамилия и имя: ")
        phone = int(input("Номер телефона: +7"))
        bday = list(map(int, input("Дата рождения: ").split('.')))
        d_bday = datetime.date(bday[2], bday[1], bday[0])
        # Создать словарь.
        human = {
            'name': name,
            'phone': phone,
            'birthday': d_bday
        }

        # Добавить словарь в список.
        peopls.append(human)
        # Отсортировать список в случае необходимости.
        if len(peopls) > 1:
            peopls.sort(key=lambda item: item.get('phone', 0))

    elif command == 'list':
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 15,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^15} | {:^15} |'.format(
                "№",
                "Ф и И.",
                "Телефон",
                "День рождения"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, human in enumerate(peopls, 1):
            print(
                f'| {idx:>4} |'
                f' {human.get("name", ""):<30} |'
                f' {human.get("phone", 0):<15} |'
                f' {human.get("birthday")} |'
            )

            print(line)

    elif command == 'find':
        f = input('Введите фамилию: ')
        for human in peopls:
            if f in str(human.values()):
                print('Запись найдена:')
                print(
                    f"Имя: {human.get('name', '')} \n"
                    f"Номер: {human.get('phone', 0)} \n"
                    f"День рождения: {human.get('birthday')} \n"
                )
                continue
            else:
                print('Запись не найдена')

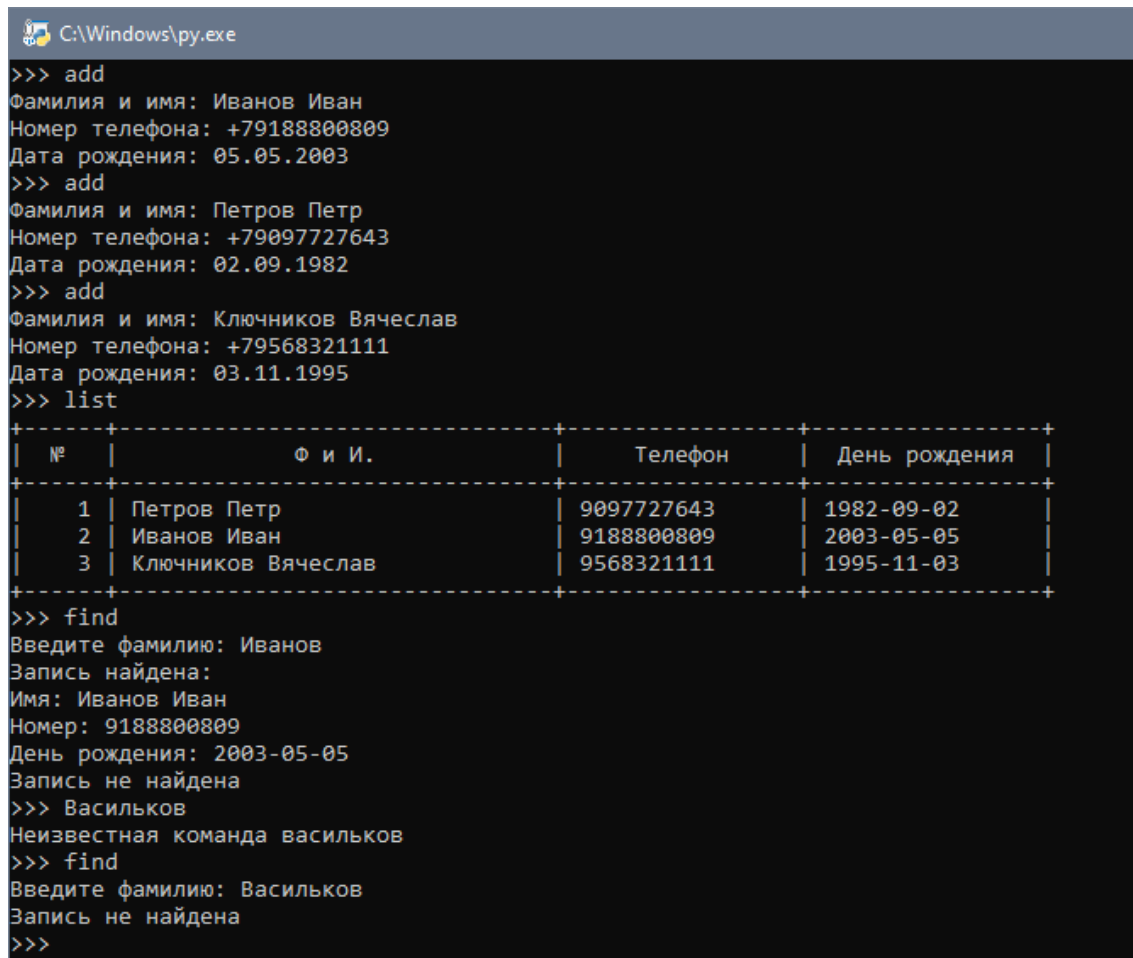
```

```

elif command == 'help':
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить человека;")
    print("list - вывести список людей;")
    print("find - поиск по фамилии;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

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

```



```

C:\Windows\py.exe
>>> add
Фамилия и имя: Иванов Иван
Номер телефона: +79188800809
Дата рождения: 05.05.2003
>>> add
Фамилия и имя: Петров Петр
Номер телефона: +79097727643
Дата рождения: 02.09.1982
>>> add
Фамилия и имя: Ключников Вячеслав
Номер телефона: +79568321111
Дата рождения: 03.11.1995
>>> list
+-----+-----+-----+-----+
| № | Ф и И. | Телефон | День рождения |
+-----+-----+-----+-----+
| 1 | Петров Петр | 9097727643 | 1982-09-02 |
| 2 | Иванов Иван | 9188800809 | 2003-05-05 |
| 3 | Ключников Вячеслав | 9568321111 | 1995-11-03 |
+-----+-----+-----+-----+
>>> find
Введите фамилию: Иванов
Запись найдена:
Имя: Иванов Иван
Номер: 9188800809
День рождения: 2003-05-05
Запись не найдена
>>> Васильков
Неизвестная команда васильков
>>> find
Введите фамилию: Васильков
Запись не найдена
>>>

```

Рисунок 11 – Результат работы программы

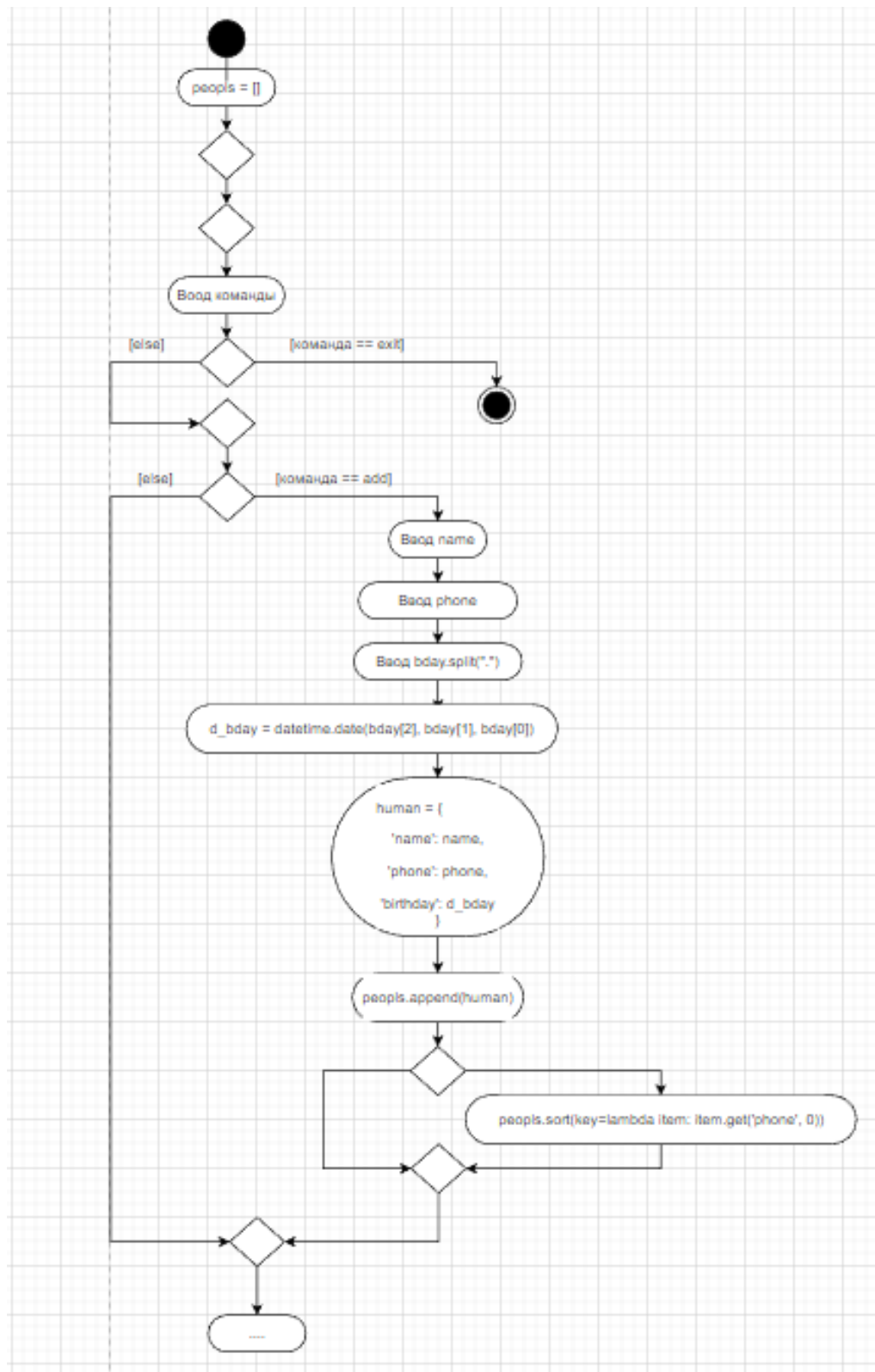


Рисунок 12 – UML-диаграмма (часть 1)

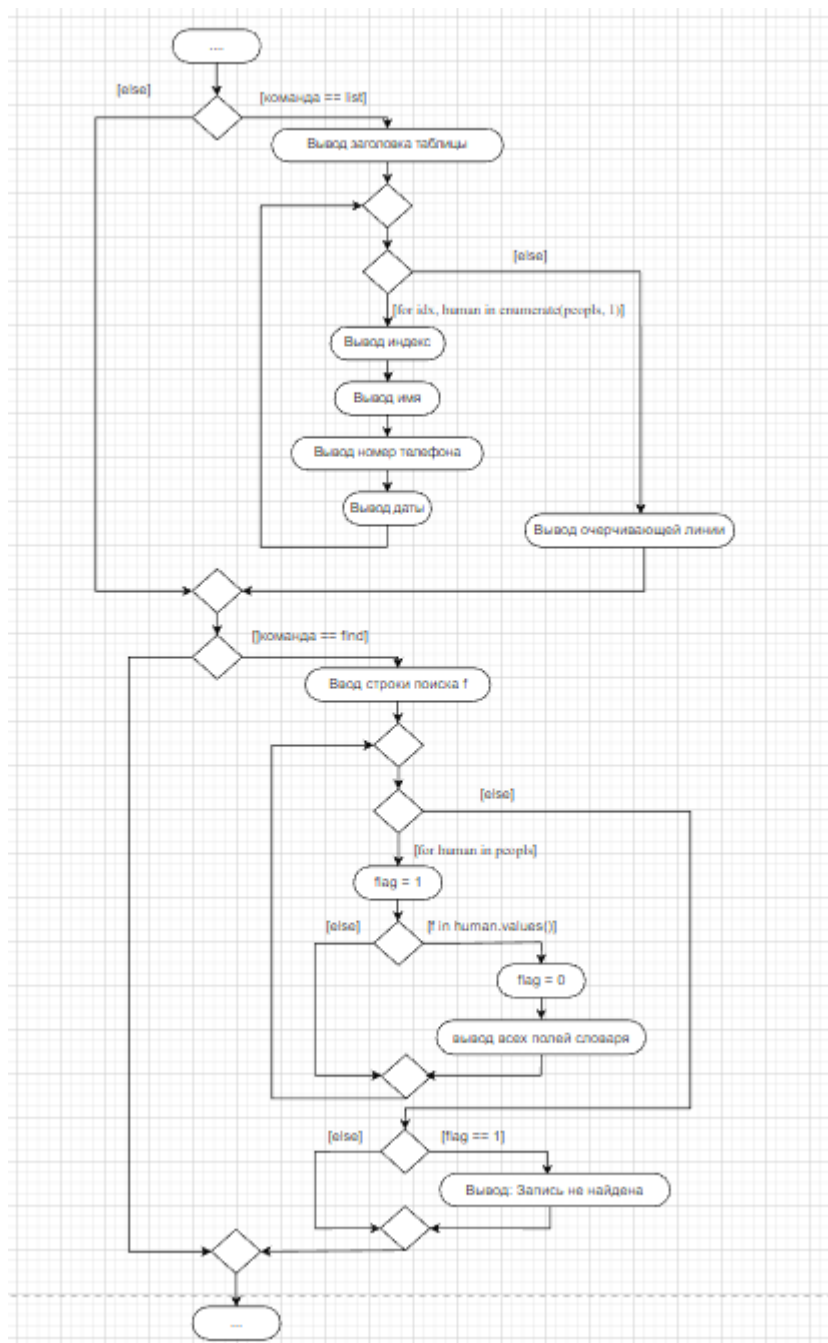


Рисунок 13 – UML-диаграмма (часть 2)

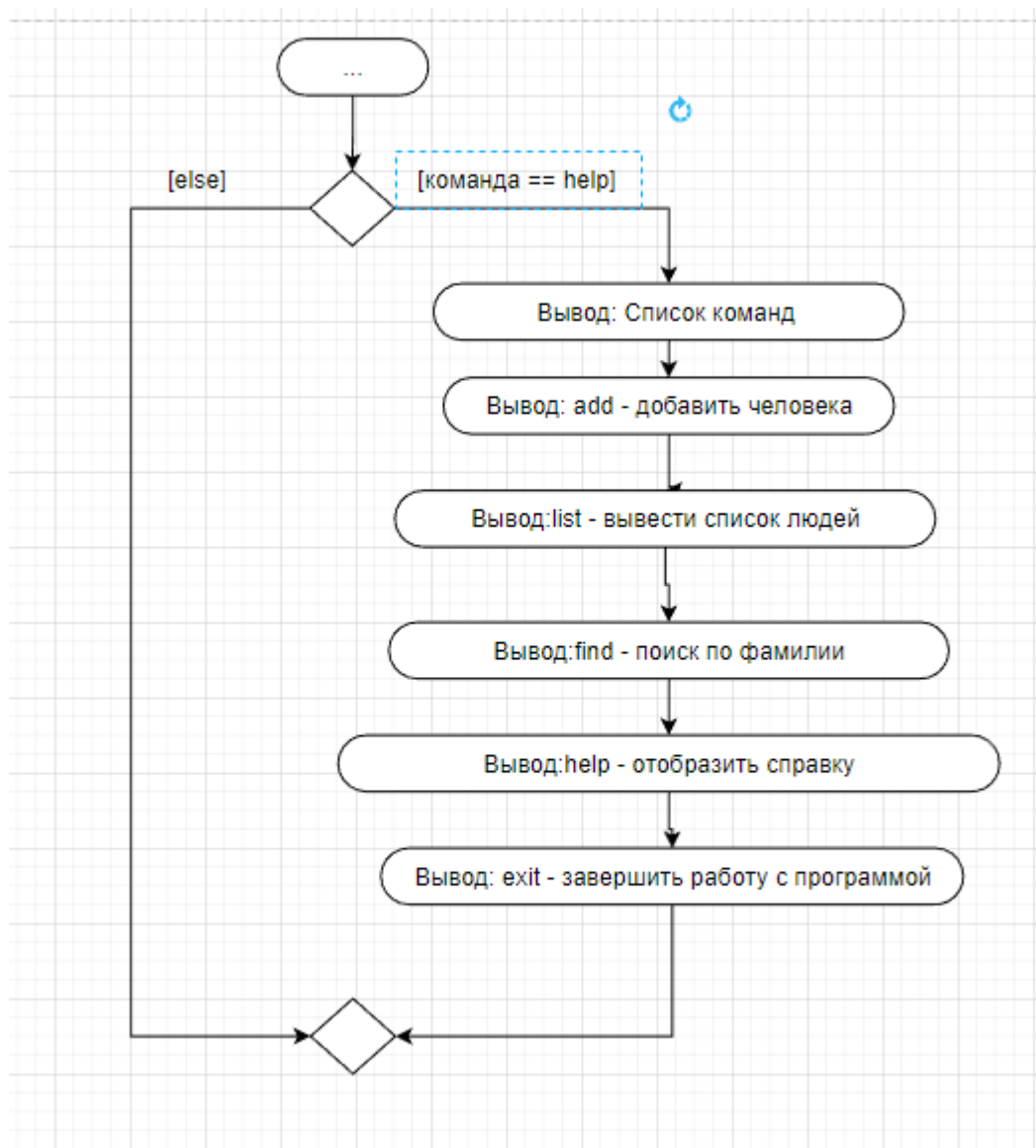


Рисунок 13 – UML-диаграмма (часть 3)

Вопросы для защиты работы

1. Что такое словари в языке Python?

Словари представляют собой структуры данных, в которых уникальные ключи отображают значения. Ключ и значение разделяются двоеточием, пары ключ-значение отделяются запятыми, а словарь целиком ограничивается фигурными скобками {}.

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

Да, она возвращает количество пар {key:value}

3. Какие методы обхода словарей Вам известны?

С помощью цикла:

```
num = {1: "one", 2: "two", ...}
```

```
for i in num:
```

```
    print(i) ///Выведет ключи
```

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

```
for i in num:
```

```
    print(num [i])
```

или

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

```
    print(key, 'is', value) // выведет пары ключ-значение
```

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

`x['one'] = 1`, где `one` – ключ, а `1` – значение

6. Что такое словарь включений?

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)
```

Вывод: [('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?

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.

Метод now() возвращает текущие дату и время с учетом локальных настроек.

today() - объект datetime из текущей даты и времени. Работает также, как и datetime.now() со значением tz=None.

fromtimestamp(timestamp) - дата из стандартного представления времени.

toordinal() - количество дней, прошедших с 01.01.1970

replace([year[, month[, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]]]])) - возвращает новый объект datetime с изменёнными атрибутами

fromordinal(ordinal) - дата из числа, представляющего собой количество дней, прошедших с 01.01.1970