

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

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №9**  
дисциплины «Основы программной инженерии»

Выполнил:  
Мелтонян Одиссей  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Проверил: Воронкин Р. А.

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г

## Тема: Работа со словарями в языке Python

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

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT 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.](#)

*Required fields are marked with an asterisk (\*).*

**Owner \*** **Repository name \***

odik8 / BSE9

✔ BSE9 is available.

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

**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:**

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

**Add .gitignore**

**.gitignore template:** Python

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

**Choose a license**

**License:** MIT License

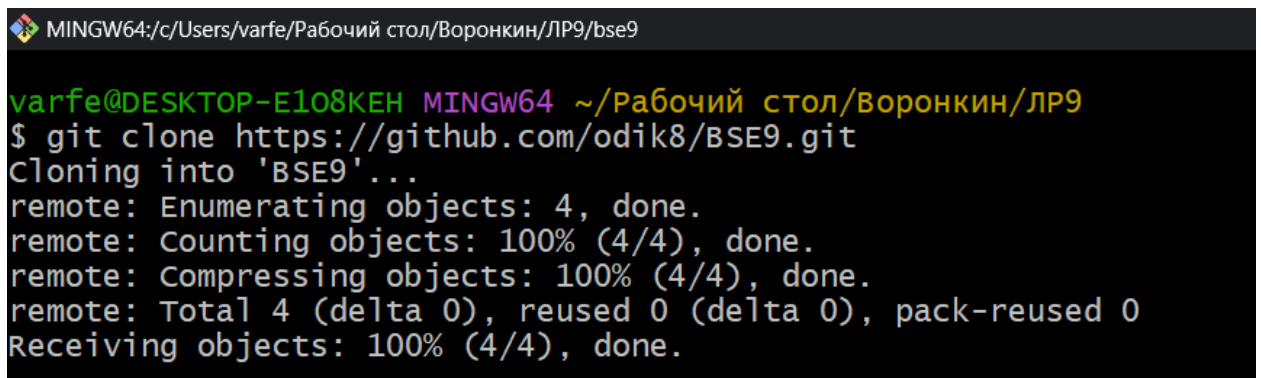
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

You are creating a public repository in your personal account.

**Create repository**

Рисунок 1 – Создание репозитория

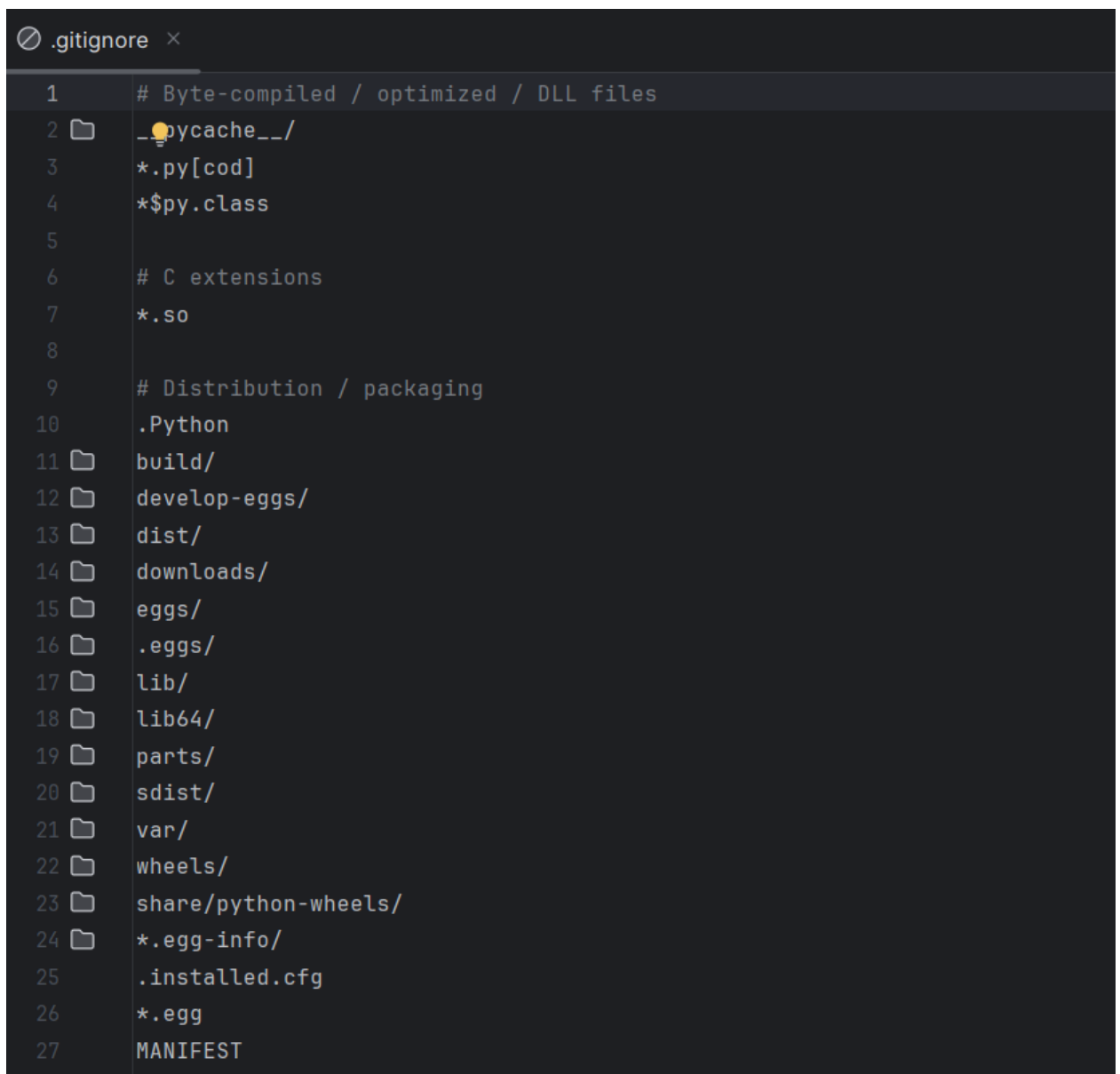
3. Выполнил клонирование созданного репозитория.



```
MINGW64:/c:/Users/varfe/Рабочий стол/Воронкин/ЛР9/bse9
varfe@DESKTOP-E108KEH MINGW64 ~/Рабочий стол/Воронкин/ЛР9
$ git clone https://github.com/odik8/BSE9.git
Cloning into 'BSE9'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2 – Клонирование репозитория

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



```
.gitignore x
1 # Byte-compiled / optimized / DLL files
2 _pypcache_/_/
3 *.py[cod]
4 *$py.class
5
6 # C extensions
7 *.so
8
9 # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
```

Рисунок 3 – Файл .gitignore

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

```
varfe@DESKTOP-E108KEN MINGW64 ~/Рабочий стол/Воронкин/ЛР9/bse9 (main)
$ git flow init

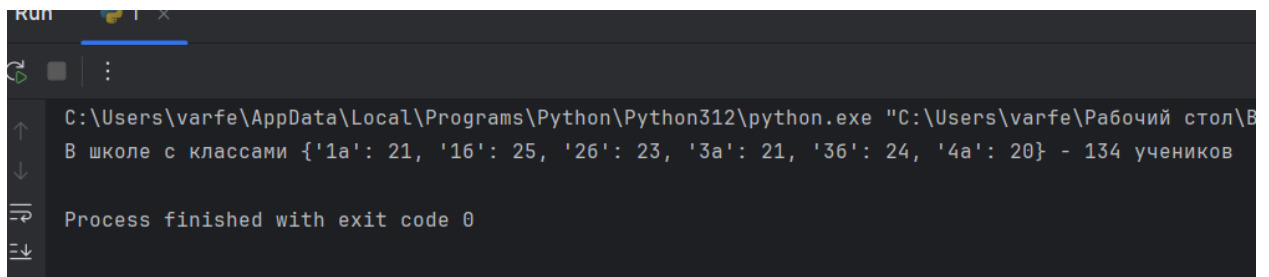
which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]
```

Рисунок 4 – Инициализация git-flow

6. Решил 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.

```
1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  1 usage  o odyssey
5  def main():
6      school = {"1a": 20, "1б": 25, "2а": 18, "2б": 23, "3а": 21, "3б": 24, }
7      #изменение_количество_учащихся
8      school["1a"] += 1
9
10     #Добавление нового класса
11     school["4а"] = 20
12
13     #Удаление одного класса
14     del school["2а"]
15
16     count_of_students = sum(school.values())
17     return f"В школе с классами {school} - {count_of_students} учеников"
18
19 if __name__ == "__main__":
20     print(main())
```

Рисунок 5 – Код решения первой задачи



```
C:\Users\varfe\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\varfe\Рабочий стол\В школе с классами {'1a': 21, '16': 25, '26': 23, '3a': 21, '36': 24, '4a': 20} - 134 учеников"
Process finished with exit code 0
```

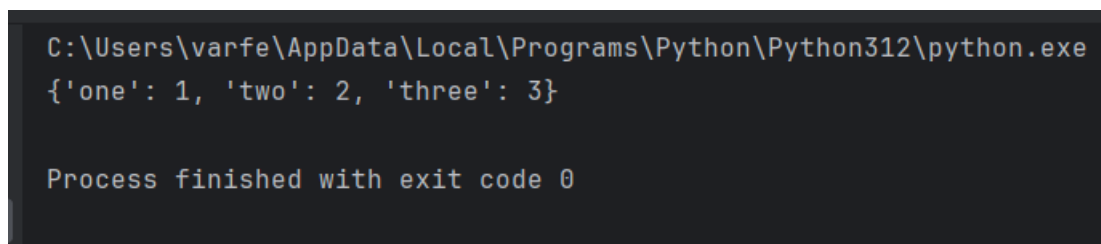
Рисунок 6 – Результат выполнения первой программы

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



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  1 usage new *
5  def main():
6      dict1 = {1: "one", 2: "two", 3: "three"}
7
8      reversed_dict = {value: key for key, value in dict1.items()}
9      return reversed_dict
10
11
12  if __name__ == "__main__":
13      print(main())
```

Рисунок 7 – Код решения второй задачи



```
C:\Users\varfe\AppData\Local\Programs\Python\Python312\python.exe
{'one': 1, 'two': 2, 'three': 3}
Process finished with exit code 0
```

Рисунок 8 – Результат выполнения второй программы

## Индивидуальное задание

Вариант 11. 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 датам рождения; вывод на экран информации о человеке, номер телефона которого введен с клавиатуры; если такого нет, выдать на дисплей соответствующее сообщение.

### Код решения:

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

from datetime import datetime

def get_birthdate():
    while True:
        try:
            date_str = input("Введите дату рождения в формате ДД.ММ.ГГГГ: ")
            birthdate = datetime.strptime(date_str, "%d.%m.%Y").date()
            return birthdate
        except ValueError:
            print("Ошибка Неправильный формат даты. Попробуйте снова.")

def add_person(list_of_people):
    print("\nДобавление нового человека:")
    last_name = input("Введите фамилию: ")
    first_name = input("Введите имя: ")
    phone_number = input("Введите номер телефона: ")
    birthdate = get_birthdate()

    person = {
        'фамилия': last_name,
        'имя': first_name,
        'номер телефона': phone_number,
        'дата рождения': birthdate
    }

    list_of_people.append(person)
    list_of_people.sort(key=lambda x: x['дата рождения'])
    print("Человек добавлен\n")

def find_person_by_phone(people, phone):
    for person in people:
        match person['номер телефона']:
            case phone:
                return person
    return None
```

```

def print_person_info(list_of_people):
    match list_of_people:
        case []:
            print("Человек не найден.\n")
        case {'фамилия': f, 'имя': i, 'номер телефона': nt, 'дата рождения':
dr}:
            print("\nИнформация о человеке:")
            print(f"Фамилия: {f}")
            print(f"Имя: {i}")
            print(f"Номер телефона: {nt}")
            print(f"Дата рождения: {dr.strftime('%d.%m.%Y')}\n")

def main():
    list_of_people = []

    while True:
        print("1. Добавить человека")
        print("2. Найти человека по номеру телефона")
        print("3. Вывести список людей")
        print("4. Выйти")
        choice = input("Выберите действие (1/2/3/4): ")

        match choice:
            case '1':
                add_person(list_of_people)
            case '2':
                phone_to_find = input("Введите номер телефона для поиска: ")
                found_person = find_person_by_phone(list_of_people,
phone_to_find)
                print_person_info(found_person)
            case '3':
                for _ in list_of_people: print_person_info(_)
            case '4':
                print("Программа завершена.")
                break
            case _:
                print("Некорректный ввод. Попробуйте снова.")

if __name__ == "__main__":
    main()

```

## Результат выполнения кода:

```

C:\Users\varfe\AppData\Local\Programs\Python\Python312\python.exe
"C:\Users\varfe\Рабочий стол\Воронкин\ЛР9\BSE9\individual.py"
1. Добавить человека
2. Найти человека по номеру телефона
3. Вывести список людей
4. Выйти
Выберите действие (1/2/3/4): 1

Добавление нового контакта:
Введите фамилию: Депп
Введите имя: Джонни
Введите номер телефона: 345
Введите дату рождения в формате ДД.ММ.ГГГГ: 09.06.1963
Контакт успешно добавлен

1. Добавить человека

```

```
2. Найти человека по номеру телефона
3. Вывести список людей
4. Выйти
Выберите действие (1/2/3/4): 1

Добавление нового контакта:
Введите фамилию: Вихорьков
Введите имя: Игорь
Введите номер телефона: 123
Введите дату рождения в формате ДД.ММ.ГГГГ: 29.12.1960
Контакт успешно добавлен

1. Добавить человека
2. Найти человека по номеру телефона
3. Вывести список людей
4. Выйти
Выберите действие (1/2/3/4): 2
Введите номер телефона для поиска: 345

Информация о человеке:
Фамилия: Депп
Имя: Джонни
Номер телефона: 345
Дата рождения: 09.06.1963

1. Добавить человека
2. Найти человека по номеру телефона
3. Вывести список людей
4. Выйти
Выберите действие (1/2/3/4): 3

Информация о человеке:
Фамилия: Вихорьков
Имя: Игорь
Номер телефона: 123
Дата рождения: 29.12.1960

Информация о человеке:
Фамилия: Депп
Имя: Джонни
Номер телефона: 345
Дата рождения: 09.06.1963

1. Добавить человека
2. Найти человека по номеру телефона
3. Вывести список людей
4. Выйти
Выберите действие (1/2/3/4):
```

Вопросы для защиты работы:

1. **Что такое словари в языке Python?** – Словарь в Python - это неупорядоченная коллекция элементов, где каждый элемент представляет собой пару ключ-значение. Ключи уникальны в пределах



словаря, и они используются для доступа к соответствующим значениям.

2. **Может ли функция `len()` быть использована при работе со словарями?** – Да, функция `len()` может использоваться для определения количества элементов (пар ключ-значение) в словаре. Например, `len(my_dict)` вернет количество элементов в словаре `my_dict`.

3. **Какие методы обхода словарей Вам известны?** – Для обхода словарей в Python можно использовать циклы `for`. Например:

```
for key in my_dict:  
  
    print(key, my_dict[key])
```

Также можно использовать методы `keys()`, `values()`, и `items()` для получения ключей, значений и пар ключ-значение соответственно.

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

– Значения из словаря можно получить по ключу с использованием квадратных скобок, например: `value = my_dict[key]`. Можно также использовать метод `get()`, который вернет значение по ключу, или значение по умолчанию, если ключ отсутствует.

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

– Значение в словаре по ключу можно установить с использованием квадратных скобок, например: `my_dict[key] = value`. Также можно использовать метод `update()` для обновления нескольких значений за один раз.

6. **Что такое словарь включений?** – Словарь включений (или словарь comprehensions) в Python — это компактный способ создания словарей с использованием синтаксиса списков включений. Пример:

```
my_dict = {key: value for key, value in iterable}
```

7. Самостоятельно изучите возможности функции **zip()** приведите примеры ее использования. – Функция **zip()** используется для объединения элементов из нескольких итерируемых объектов в кортежи. Пример:

```
keys = ['a', 'b', 'c']
```

```
values = [1, 2, 3]
```

```
my_dict = dict(zip(keys, values))
```

8. Самостоятельно изучите возможности модуля **datetime**. Каким функционалом по работе с датой и временем обладает этот модуль?

– Модуль **datetime** предоставляет классы для работы с датой и временем. Некоторые из основных классов включают **datetime**, **date**, **time**, и **timedelta**. Модуль позволяет выполнять операции с датой и временем, форматировать и парсить строки, а также работать с интервалами времени. Примеры:

```
from datetime import datetime, timedelta
```

```
now = datetime.now()
```

```
print(now)
```

```
future_date = now + timedelta(days=7)
```

```
print(future_date)
```