

**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  
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-22-1

Душин Александр Владимирович.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2023

Тема: Работа со словарями в языке Python.

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

Ход выполнения работы:


1. Создать общедоступный репозиторий на GitHub с использованием лицензии 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 \*

 MrPlatynum ▾

Repository name \*

/ ProgrammEngineering9

✔ ProgrammEngineering9 is available.

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

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 общедоступного репозитория на GitHub с заданными настройками

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering9.git
Cloning into 'ProgrammEngineering9'...
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 созданного репозитория на локальный компьютер



Рисунок 3 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering9 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering9 (develop)
$ |
```

Рисунок 4 – организация репозитория в соответствии с моделью ветвления git flow

2. Проработать примеры лабораторной работы, оформляя код согласно PEP-8:

```
example1.py x
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        elif command == 'add':
18            # Запросить данные о работнике.
19            name = input("Фамилия и инициалы? ")
20            post = input("Должность? ")
21            year = int(input("Год поступления? "))
22            # Создать словарь.
23            worker = {
24                'name': name,
25                'post': post,
26                'year': year,
27            }
28            # Добавить словарь в список.
29            workers.append(worker)
30            # Отсортировать список в случае необходимости.
31            if len(workers) > 1:
32                workers.sort(key=lambda item: item.get('name', ''))
33        elif command == 'list':
34            # Заголовок таблицы.
35            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
36                '-' * 4,
37                '-' * 30,
38                '-' * 20,
39                '-' * 8
40            )
41            print(line)
42            print(
```

```

43         '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
44             "№",
45             "Ф.И.О.",
46             "Должность",
47             "Год"
48         )
49     )
50     print(line)
51     # Вывести данные о всех сотрудниках.
52     for idx, worker in enumerate(workers, 1):
53         print(
54             '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
55                 idx,
56                 worker.get('name', ''),
57                 worker.get('post', ''),
58                 worker.get('year', 0)
59             )
60         )
61     print(line)
62     elif command.startswith('select '):
63         # Получить текущую дату.
64         today = date.today()
65         # Разбить команду на части для выделения номера года.
66         parts = command.split(' ', maxsplit=1)
67         # Получить требуемый стаж.
68         period = int(parts[1])
69         # Инициализировать счетчик.
70         count = 0
71         # Проверить сведения работников из списка.
72         for worker in workers:
73             if today.year - worker.get('year', today.year) >= period:
74                 count += 1
75                 print(
76                     '{:>4}: {}'.format(count, worker.get('name', ''))
77                 )
78         # Если счетчик равен 0, то работники не найдены.
79         if count == 0:
80             print("Работники с заданным стажем не найдены.")
81     elif command == 'help':

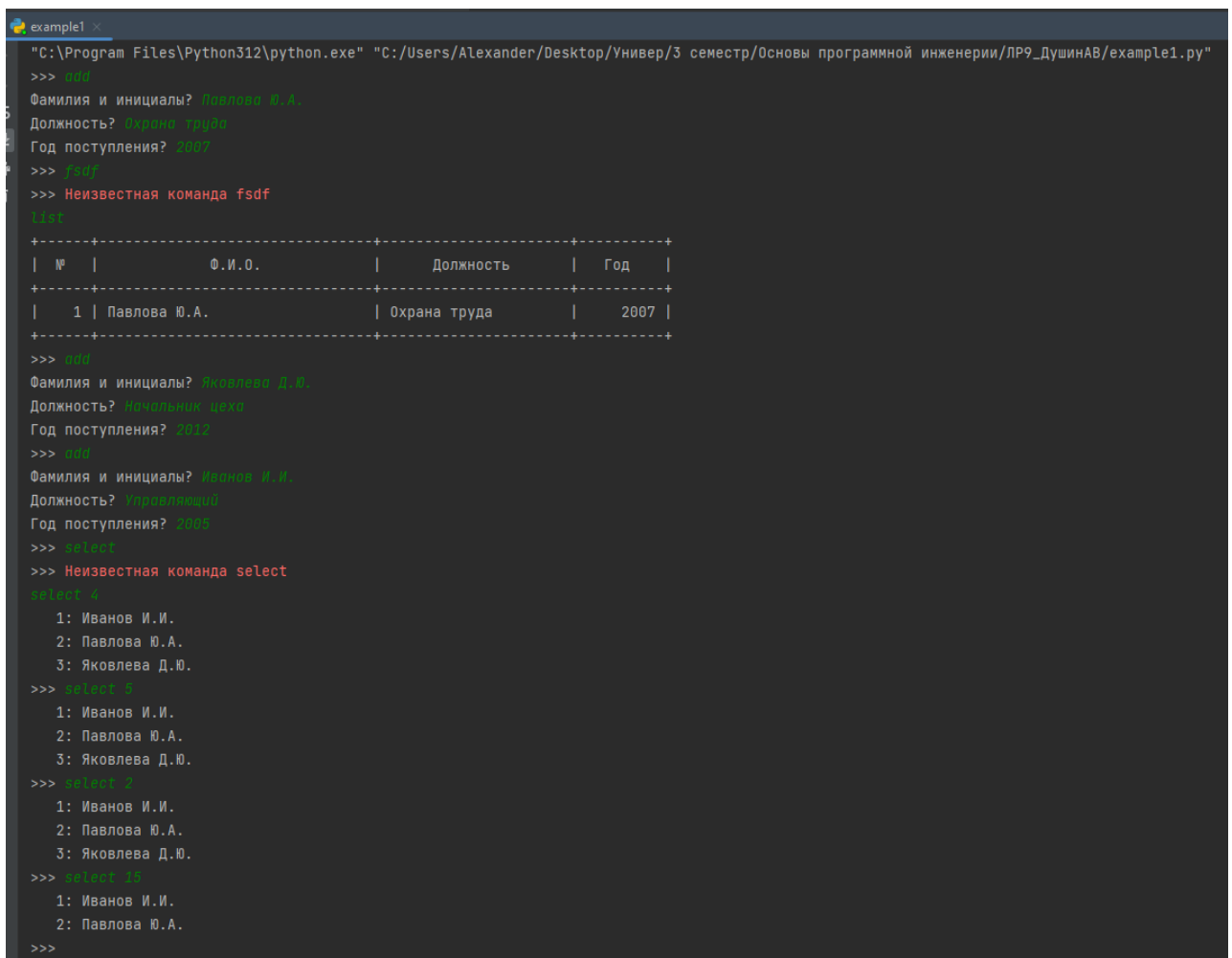
```

```

82         # Вывести справку о работе с программой.
83         print("Список команд:\n")
84         print("add - добавить работника;")
85         print("list - вывести список работников;")
86         print("select <стаж> - запросить работников со стажем;")
87         print("help - отобразить справку;")
88         print("exit - завершить работу с программой.")
89     else:
90         print(f"Неизвестная команда {command}", file=sys.stderr)
91

```

Рисунок 5 – Пример 1



```

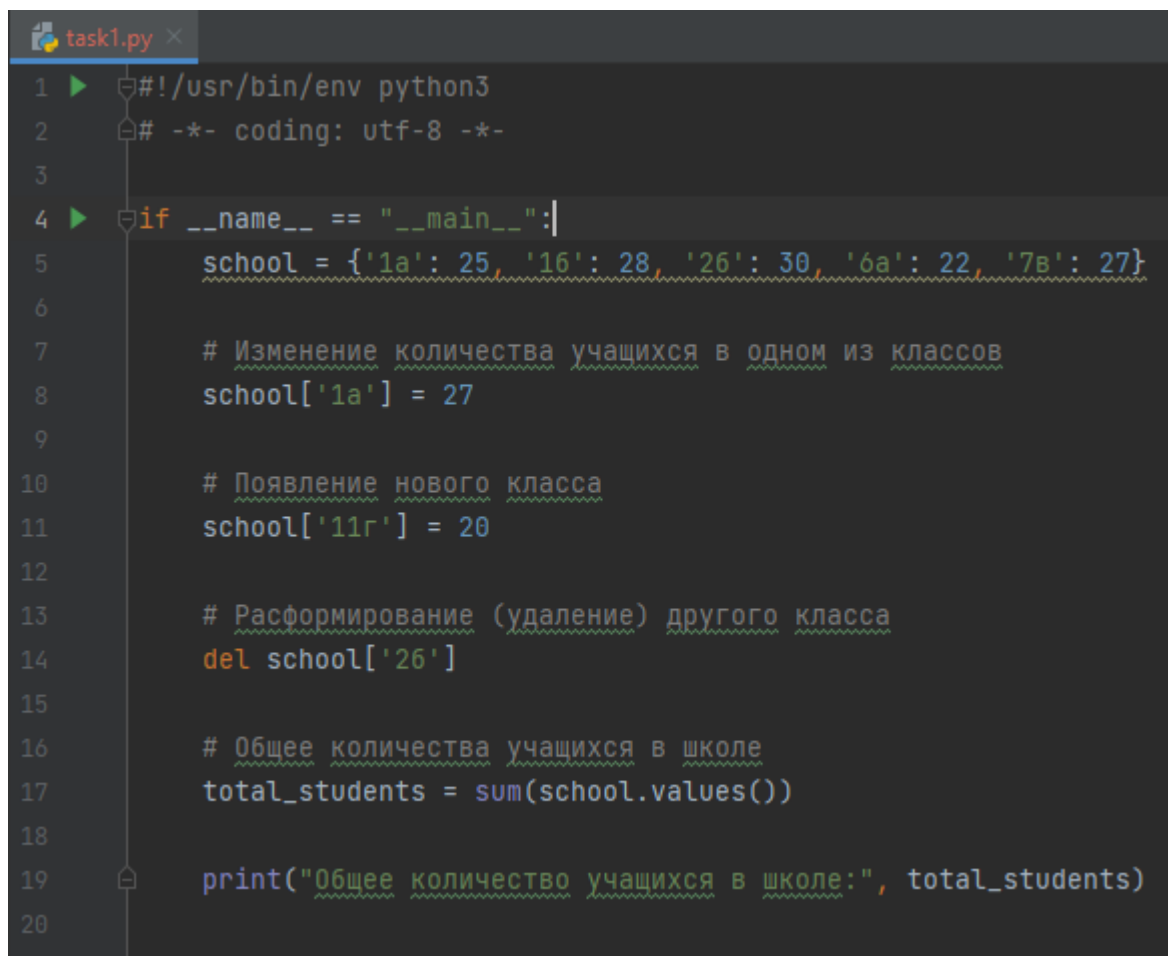
example1
"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы программной инженерии/ЛР9_ДушинАВ/example1.py"
>>> add
Фамилия и инициалы? Павлова Ю.А.
Должность? Охрана труда
Год поступления? 2007
>>> list
Неизвестная команда fsdf
list
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Павлова Ю.А.             | Охрана труда        |      2007     |
+-----+-----+-----+-----+
>>> add
Фамилия и инициалы? Яковлева Д.Ю.
Должность? Начальник цеха
Год поступления? 2012
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? Управляющий
Год поступления? 2005
>>> select
Неизвестная команда select
select 4
1: Иванов И.И.
2: Павлова Ю.А.
3: Яковлева Д.Ю.
>>> select 5
1: Иванов И.И.
2: Павлова Ю.А.
3: Яковлева Д.Ю.
>>> select 2
1: Иванов И.И.
2: Павлова Ю.А.
3: Яковлева Д.Ю.
>>> select 15
1: Иванов И.И.
2: Павлова Ю.А.
>>>

```

Рисунок 6 – Вывод программы (Пример 1)

3. Решим 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 = {'1a': 25, '16': 28, '26': 30, '6a': 22, '7в': 27}
6
7      # Изменение количества учащихся в одном из классов
8      school['1a'] = 27
9
10     # Появление нового класса
11     school['11г'] = 20
12
13     # Расформирование (удаление) другого класса
14     del school['26']
15
16     # Общее количества учащихся в школе
17     total_students = sum(school.values())
18
19     print("Общее количество учащихся в школе:", total_students)
20
```

Рисунок 7 – задание №1



```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering9/task1.py
Общее количество учащихся в школе: 124
```

Рисунок 8 – Вывод задания №1

4. 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.

```
task2.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     original_dict = {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
6
7     # Получение объекта dict_items
8     dict_items = original_dict.items()
9
10    # Создание "обратного" словаря
11    inverted_dict = {value: key for key, value in dict_items}
12
13    print("Обратный словарь:")
14    print(inverted_dict)
15
```

Рисунок 9 – Задание №2

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering9/task2.py
Обратный словарь:
{'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}
```

Рисунок 10 – Вывод задания №2

5. Выполним индивидуальные задания:

Листинг программы:

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

if __name__ == '__main__':
    # Список поездов.
    trains = []
    # 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 поезде.
            destination = input("Название пункта назначения: ")
            train_number = input("Номер поезда: ")
            departure_time = input("Время отправления (в формате ЧЧ:ММ): ")

            # Создать словарь для поезда.
            train = {
                'название пункта назначения': destination,
```



```

        'номер поезда': train_number,
        'время отправления': departure_time,
    }

    # Добавить словарь в список поездов.
    trains.append(train)
    # Отсортировать список по названиям пунктов назначения.
    trains.sort(key=lambda x: x['название пункта назначения'])

elif command == 'list':
    line = f'+-{"-" * 35}-+-{"-" * 15}-+-{"-" * 25}-+'
    print(line)
    print(f'| {"Название пункта назначения":^35} | {"Номер поезда":^15} | {"Время отправления":^25} |')

    for train in trains:
        print(line)
        print(
            f'| {train["название пункта назначения":^35} | {train["номер поезда":^15} | {train["время отправления":^25} |')
        print(line)

elif command.startswith('select '):
    # Получить текущее время для сравнения.
    search_time = command.split(' ')[1]
    found = False

    # Вывести поезда, отправляющиеся после введенного времени.
    print(f"Поезда, отправляющиеся после {search_time}:")
    for train in trains:
        if train['время отправления'] > search_time:
            print(f"Название пункта назначения: {train['название пункта назначения']}, "
                  f"Номер поезда: {train['номер поезда']}, "
                  f"Время отправления: {train['время отправления']}")
            found = True

    if not found:
        print("Нет поездов, отправляющихся после указанного времени.")

elif command == 'help':
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить информацию о поезде;")
    print("list - вывести список всех поездов;")
    print("select <время> - вывести поезда, отправляющиеся после указанного времени;")
    print("exit - завершить работу с программой.")
else:
    print(f"Неизвестная команда {command}")

```

```

"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering9/individual1.py
>>> add
Название пункта назначения: Москва
Номер поезда: 13
Время отправления (в формате ЧЧ:ММ): 13:00
>>> add
Название пункта назначения: Омск
Номер поезда: 116
Время отправления (в формате ЧЧ:ММ): 13:00
>>> add
Название пункта назначения: 5
Номер поезда: fd
Время отправления (в формате ЧЧ:ММ): 13:00
>>> list
+-----+-----+-----+
| Название пункта назначения | Номер поезда | Время отправления |
+-----+-----+-----+
| 5 | fd | 13:00 |
+-----+-----+-----+
| Москва | 13 | 13:00 |
+-----+-----+-----+
| Омск | 116 | 13:00 |
+-----+-----+-----+
>>> select 13:00
Поезда, отправляющиеся после 13:00:
Нет поездов, отправляющихся после указанного времени.
>>> select 12:00
Поезда, отправляющиеся после 12:00:
Название пункта назначения: 5, Номер поезда: fd, Время отправления: 13:00
Название пункта назначения: Москва, Номер поезда: 13, Время отправления: 13:00
Название пункта назначения: Омск, Номер поезда: 116, Время отправления: 13:00
>>> exit

```

Рисунок 11 – Вывод программы

6. Зафиксируем сделанные изменения, сольем ветки и отправим на удаленный репозиторий:

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering9 (develop)
$ git log
commit b90969c5e93d8870fdb5907b6ef3dd9411c6361 (HEAD -> develop)
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Sun Nov 26 21:48:21 2023 +0300

    финальные изменения

commit c4850bb083655c24fda2dfac445895bcc50577fa (origin/main, origin/HEAD, main)
Author: MrPlatynum <71084177+MrPlatynum@users.noreply.github.com>
Date: Sun Nov 26 14:15:46 2023 +0300

    Initial commit

```

Рисунок 12 – Коммиты ветки develop во время выполнения лабораторной работы

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering9 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering9 (main)
$ git merge develop
Updating c4850bb..b90969c
Fast-forward
 .gitignore                  | 2 +-
 .idea/.gitignore            | 8 +++
 .idea/.name                 | 1 +
 .idea/ProgrammEngineering9.iml | 8 +++
 .idea/inspectionProfiles/profiles_settings.xml | 6 ++
 .idea/misc.xml              | 4 ++
 .idea/modules.xml           | 8 +++
 .idea/vcs.xml               | 6 ++
 example1.py                 | 90 +++++
 individual1.py              | 71 +++++
 task1.py                    | 19 +++++
 task2.py                    | 14 ++++
12 files changed, 236 insertions(+), 1 deletion(-)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/.name
create mode 100644 .idea/ProgrammEngineering9.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 example1.py
create mode 100644 individual1.py
create mode 100644 task1.py
create mode 100644 task2.py

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering9 (main)
$ |

```

Рисунок 13 – Слияние ветки develop в ветку main

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering9 (main)
$ git push origin main
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (16/16), 4.84 KiB | 4.84 MiB/s, done.
Total 16 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/MrPlatynum/ProgrammEngineering9.git
   c4850bb..b90969c  main -> main

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering9 (main)
$ |

```

Рисунок 14 – Отправка на удаленный репозиторий

Ответы на контрольные вопросы:

1. Что такое словари в языке Python?

Словари (dictionaries) в Python – это структуры данных, которые хранят коллекцию пар ключ-значение. Они предоставляют эффективный способ хранения и доступа к данным. Ключи словаря должны быть уникальными и неизменяемыми, а значения могут быть любого типа.

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

Да, функция len() может быть использована для определения количества элементов (пар ключ-значение) в словаре. Например:

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
length = len(my_dict)
print(length) # Выведет: 3
```

3. Какие методы обхода словарей Вам известны?

В Python есть несколько способов обхода словарей:

Цикл for для перебора ключей или элементов словаря.

Методы .keys(), .values() и .items() для получения ключей, значений или пар ключ-значение в виде итерируемых объектов.

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

Для получения значения из словаря по ключу можно использовать:

Оператор доступа к элементу по ключу (my\_dict[key]), который вернет значение, связанное с данным ключом.

Метод .get(key), который вернет значение по ключу или None, если ключ отсутствует.

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

Чтобы установить значение в словаре по ключу:

Просто присвойте значение ключу: my\_dict[key] = value.

Используйте метод .update() для обновления значений или добавления новых пар ключ-значение.

6. Что такое словарь включений?

Словарь включений (dictionary comprehensions) – это способ создания нового словаря с помощью компактного синтаксиса, используя циклы и условия. Например:

```
squares = {x: x*x for x in range(5)} # Создание словаря с квадратами чисел от 0 до 4
```

#### 7. Функция zip() и примеры ее использования.

Функция zip() в Python используется для объединения элементов из нескольких итерируемых объектов в один. Например:

```
names = ['Alice', 'Bob', 'Charlie']
ages = [25, 30, 35]
combined = zip(names, ages) # Объединение имен и возрастов в пары
combined_list = list(combined) # Преобразование объекта zip в список пар
```

#### 8. Модуль datetime и его функционал по работе с датой и временем.

Модуль datetime в Python предоставляет классы для работы с датой, временем и интервалами. Включает классы datetime, date, time и timedelta, позволяющие создавать, обрабатывать и оперировать датами и временем, вычислять разницу между датами, форматировать вывод и многое другое. Например:

```
from datetime import datetime, timedelta
current_time = datetime.now() # Получение текущей даты и времени
future_time = current_time + timedelta(days=7) # Добавление 7 дней к текущей дате
```