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 ПРАКТИЧЕСКОЙ РАБОТЕ №9
дисциплины «Программирование на Python»

Выполнил:
Лейс Алексей Вячеславович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем»

_____
(подпись)

Руководитель практики: кандидат тех. наук доцент кафедры инфокоммуникаций: 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 версии 3.x.

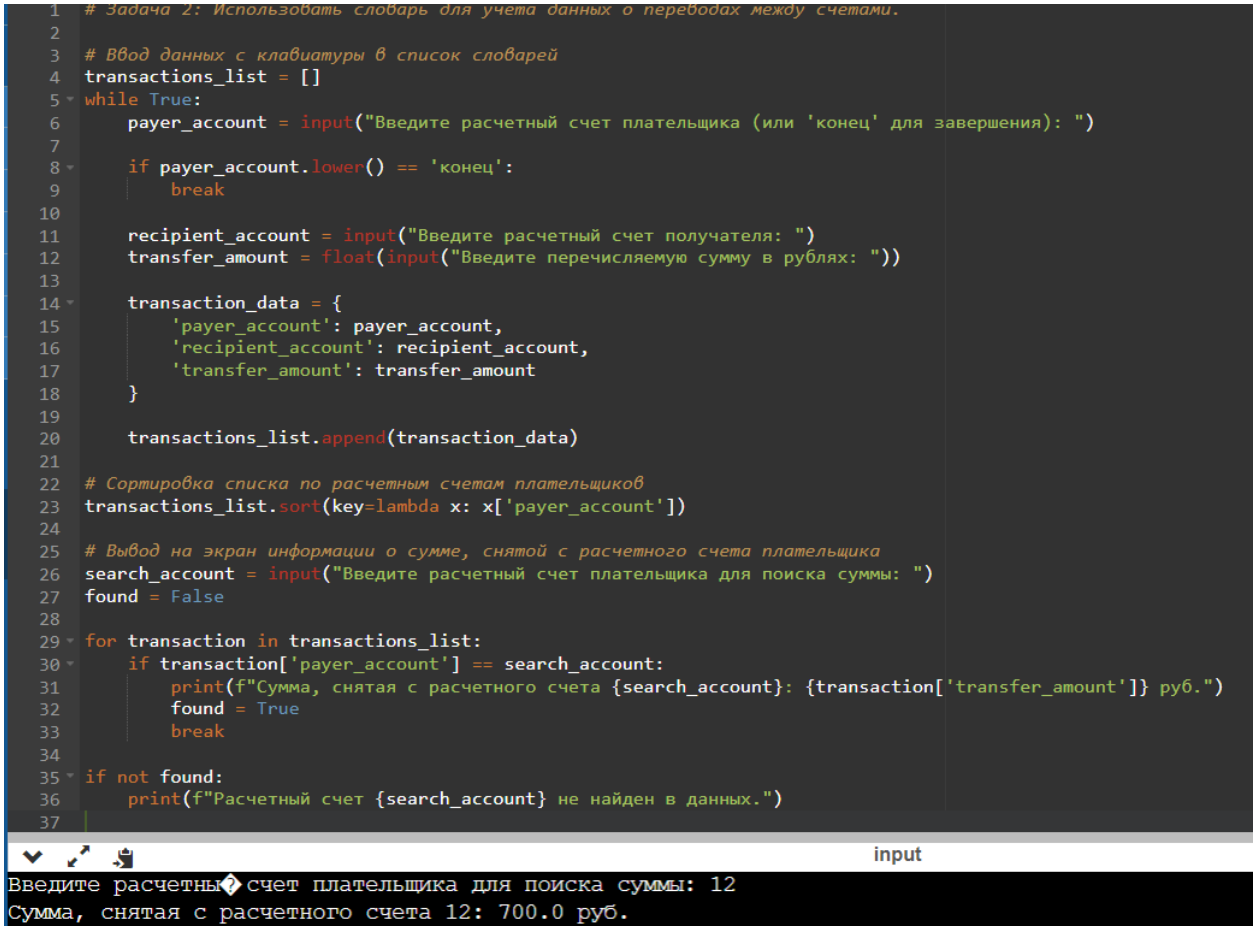
Порядок выполнения работы:

Пример 1:

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


import re
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.
        match command:
            case 'exit':
                break

            case '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', ''))

            case 'list':
                # Заголовок таблицы.
                line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                    '-' * 4,
                    '-' * 30,
                    '-' * 20,
                    '-' * 8
```

Индивидуальное задание:

Использовать словарь, содержащий следующие ключи: расчетный счет плательщика; расчетный счет получателя; перечисляемая сумма в руб. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть размещены в алфавитном порядке по расчетным счетам плательщиков; вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры; если такого расчетного счета нет, выдать на дисплей соответствующее сообщение.



```
# Задача 2: Использовать словарь для учета данных о переводах между счетами.

# Ввод данных с клавиатуры в список словарей
transactions_list = []
while True:
    payer_account = input("Введите расчетный счет плательщика (или 'конец' для завершения): ")

    if payer_account.lower() == 'конец':
        break

    recipient_account = input("Введите расчетный счет получателя: ")
    transfer_amount = float(input("Введите перечисляемую сумму в рублях: "))

    transaction_data = {
        'payer_account': payer_account,
        'recipient_account': recipient_account,
        'transfer_amount': transfer_amount
    }

    transactions_list.append(transaction_data)

# Сортировка списка по расчетным счетам плательщиков
transactions_list.sort(key=lambda x: x['payer_account'])

# Вывод на экран информации о сумме, снятой с расчетного счета плательщика
search_account = input("Введите расчетный счет плательщика для поиска суммы: ")
found = False

for transaction in transactions_list:
    if transaction['payer_account'] == search_account:
        print(f"Сумма, снятая с расчетного счета {search_account}: {transaction['transfer_amount']} руб.")
        found = True
        break

if not found:
    print(f"Расчетный счет {search_account} не найден в данных.")
```

```
input
Введите расчетны счет плательщика для поиска суммы: 12
Сумма, снятая с расчетного счета 12: 700.0 руб.
```

Ответы на вопросы:

1. **Что такое словари в языке Python?**

- Словари в Python - это изменяемые коллекции, представляющие собой набор пар "ключ-значение". Они используются для хранения данных, к которым можно обращаться по ключу.

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.

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

- Для обхода словарей можно использовать циклы, такие как **for key in my_dict:**. Также есть методы, такие как **keys()**, **values()**, и **items()**, которые возвращают представления 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]**.

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**.

6. **Что такое словарь включений?**

- Словарь включений (или словарное включение) - это компактный способ создания словаря в одной строке с использованием синтаксиса, например: **my_dict = {key: value for key, value in iterable}**.

7. **Самостоятельно изучите возможности функции zip() и приведите примеры ее использования.**

- Функция **zip()** используется для объединения элементов из нескольких итерируемых объектов. Пример: **list_of_keys = ['a', 'b', 'c']**, **list_of_values = [1, 2, 3]**, **my_dict = dict(zip(list_of_keys, list_of_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. Он включает функционал для создания, форматирования и арифметики с датами и временем, а также поддерживает различные календарные системы. Примеры включают **datetime.datetime** для представления даты и времени, и методы для работы с ними, такие как **strftime()**, **strptime()**, и многое другое.

Вывод В ходе работы приобрел навыки по работе со словарями при написании программ с помощью языка программирования Python версии 3.x.