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 №2.6

Дисциплина: «Основы программной инженерии»

Тема: «Условные операторы и циклы в языке Python»

Выполнила:

студентка 2 курса

группы Пиж-б-о-21-1

Логвинов Иван

Васильевич

Ставрополь 2022

1. Был создан общедоступного репозиторий в GitHub в котором были добавлены gitignore, правила для работы с IDE PyCharm с ЯП Python и лицензия MIT, репозиторий был клонировал на локальный сервер и организован в соответствие с моделью ветвления git-flow.

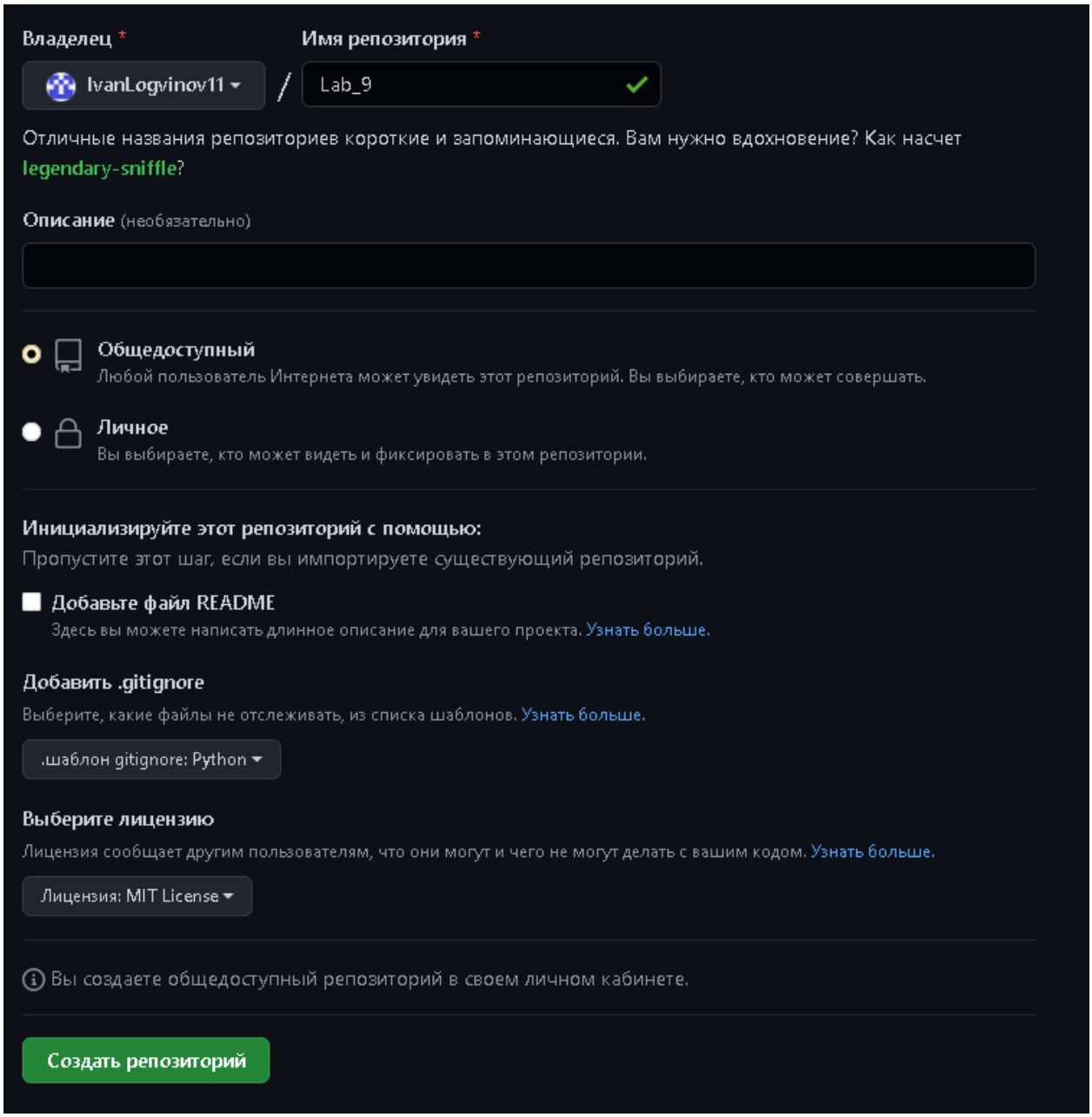

Рисунок 1.1 – Создание общедоступного репозитория

```
Microsoft Windows [Version 10.0.18363.418]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\Иван>cd desktop

C:\Users\Иван\Desktop>git clone https://github.com/IvanLogvinov11/Lab_9.git
Cloning into 'Lab_9'...
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.

C:\Users\Иван\Desktop>
```

Рисунок 1.2 – Клонирование созданного репозитория

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

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
            )
```

Рисунок 1.3 - Код примера

```
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                    "No",
                    "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', ''))
                    )
```

Рисунок 1.4 – Код примера

```
    # 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)
```

Рисунок 1.5 – Код примера

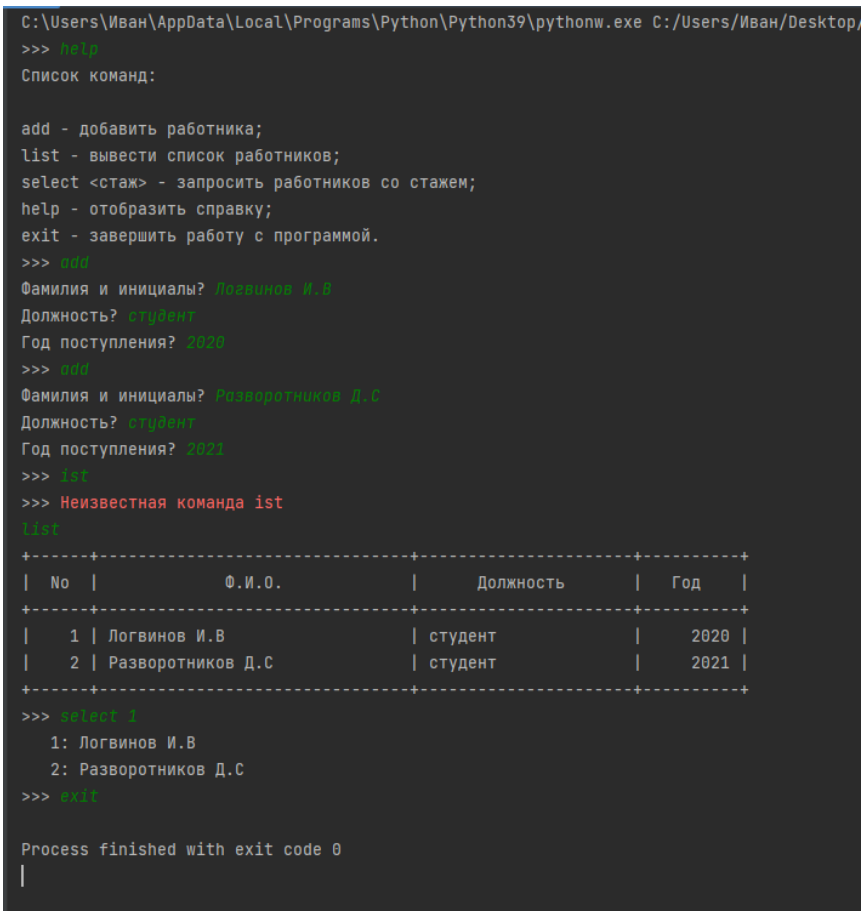

Рисунок 1.6 – Результат работы примера

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.

Код программы:

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

```
import sys


if __name__ == '__main__':

    # List for dictionaries with bank accounts
    requisites = []

    # Endless program call
    while True:

        # Entering the required command
        command = input ("Enter Command: ").lower()

        # Command to end the program
        if command == "exit":
            break

        # Command to add bank accounts
        elif command == "add":

            while True:
                s_b_a = input("Enter the sender's bank account: ")

                # Checking the spelling of a bank account
                if len(s_b_a) != 20 or s_b_a.isdigit() is False:
                    print("Incorrect bank account!")
                else:
                    break

            while True:
                b_a = input("Enter the beneficiary's account: ")

                # Checking the spelling of a bank account
                if len(b_a) != 20 or b_a.isdigit() is False:
                    print("Incorrect bank account!")
                else:
                    break

            t_a = input("Enter transfer amount in ₽: ")

            # Creating dictionary with bank account
            requisite = {
                "s_b_a": s_b_a,
                "b_a": b_a,
                "t_a": t_a,
            }

            # Adding to the list
            requisites.append(requisite)

            # Sorting
            if len(requisites) > 1:
                requisites.sort(key=lambda item: item.get("s_b_a", ""))

        # Command for displaying entered values
        elif command == "list":

            # Create a table header
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 2,
                '-' * 25,
                '-' * 25,
                '-' * 10
```

```
            )
            print(line)
            print(
                '| {:^2} | {:^25} | {:^25} | {:^10} |'.format(
                    "№",
                    "Sender bank account",
                    "beneficiary account",
                    "Amount",
                )
            )
            print(line)

            # Filling the table with entered values
            for ind, requisite in enumerate(requisites, 1):
                print(
                    '| {:^2} | {:^25} | {:^25} | {:^10} |'.format(
                        ind,
                        requisite.get('s_b_a'),
                        requisite.get('b_a'),
                        requisite.get('t_a'),
                    )
                )
                print(line)

        # The command to receive the withdrawn amount from bank account
        elif command.startswith("select "):

            # Splitting input into 2 words
            parts = command.split(" ", maxsplit=1)

            # Assign value to second word (number of bank account)
            bank_acc = int(parts[1])

            full_summa = 0
            for requisite in requisites:

                # Comparison of each dictionary from the list by key "s_b_a"
with the entered one
                if int(requisite.get("s_b_a")) == bank_acc:

                    # Calculation of the withdrawn amount from bank account
                    full_summa += float(requisite.get("t_a"))

            # Checking for the existence of such an account
            if full_summa == 0:
                print("This bank account does not exist")
            else:
                print("The sum of all transfers of the entered", end=" ")
                print(f"bank account is equal to {full_summa}")

        # Help Command
        elif command == 'help':
            print("Command List:\n")
            print("add - Add bank account;")
            print("list - Display a list of bank accounts;")
            print("select <bank account> -", end=" ")
            print("The withdrawn amount from account;")
            print("help - Display Help;")
            print("exit - End the program.")
            print("\n")

        # Incorrect command message
        else:
            print(f"Invalid command {command}", file=sys.stderr)
```

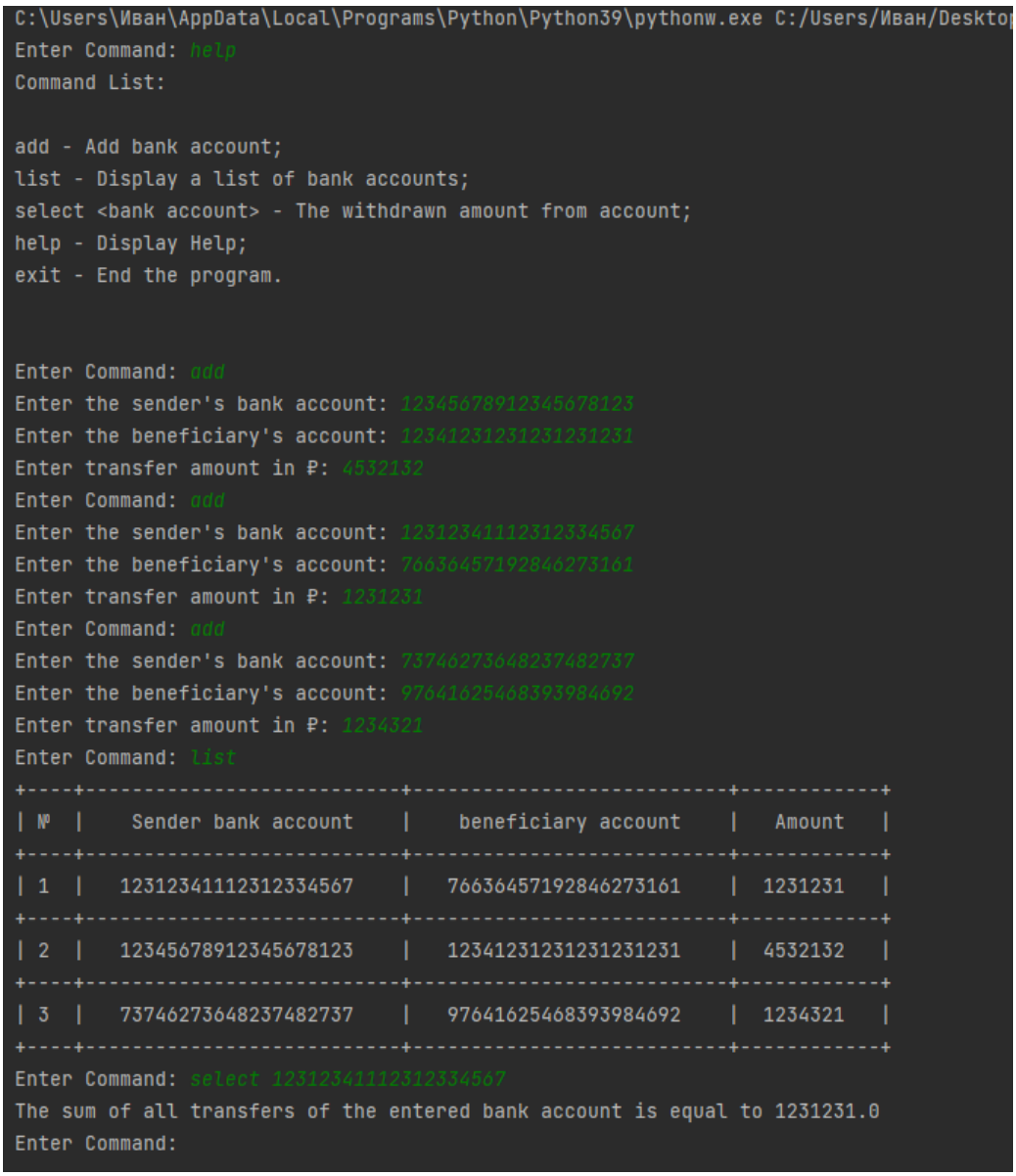

Рисунок 2.1 – Результат работы программы индивидуального задания

Контрольные вопросы

1. Что такое словари в языке Python?

Словарь (dict) представляет собой структуру данных (которая ещё называется ассоциативный массив), предназначенную для хранения произвольных объектов с доступом по ключу. Данные в словаре хранятся в формате ключ – значение.

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

Функция len() широко используется для определения размера объектов

в Python. В нашем случае передача объекта словаря этой функции вернет размер словаря, то есть количество пар ключ-значение, присутствующих в словаре.

3. Какие методы обхода словарей Вам известны?

Элементы словаря перебираются в цикле for также, как элементы других сложных объектов. Однако "по-умолчанию" извлекаются только ключи. С другой стороны у словаря как класса есть метод items(), который создает особую структуру, состоящую из кортежей. Каждый кортеж включает ключ и значение.

Методы словаря keys() и values() позволяют получить отдельно перечни ключей и значений. Так что если, например, надо перебрать только значения или только ключи, лучше воспользоваться одним из этих методов.

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

```
>>> for i in nums:
... print(nums[i])
...
one
two
three
```

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

С помощью метода setdefault(), при непосредственном обращении к ключу словарю.

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. У функции zip() множество сценариев применения. Например, она пригодится, если нужно создать набор словарей из двух массивов, каждый из которых содержит имя и номер сотрудника.

Функция zip() принимает итерируемый объект, например, список, кортеж,

множество или словарь в качестве аргумента. Затем она генерирует список

кортежей, которые содержат элементы из каждого объекта, переданного в

функцию. Предположим, что есть список имен и номером сотрудников, и их

нужно объединить в массив кортежей. Для этого можно использовать функцию zip().

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

Модуль datetime предоставляет классы для обработки времени и даты

разными способами. Поддерживается и стандартный способ представления

времени, однако больший упор сделан на простоту манипулирования датой,

временем и их частями.