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

Кучеренко С. Ю. « » 2022г.

Подпись студента_____

Работа защищена « »_____2022г.

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.

Выполнение работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT и язык программирования Python.

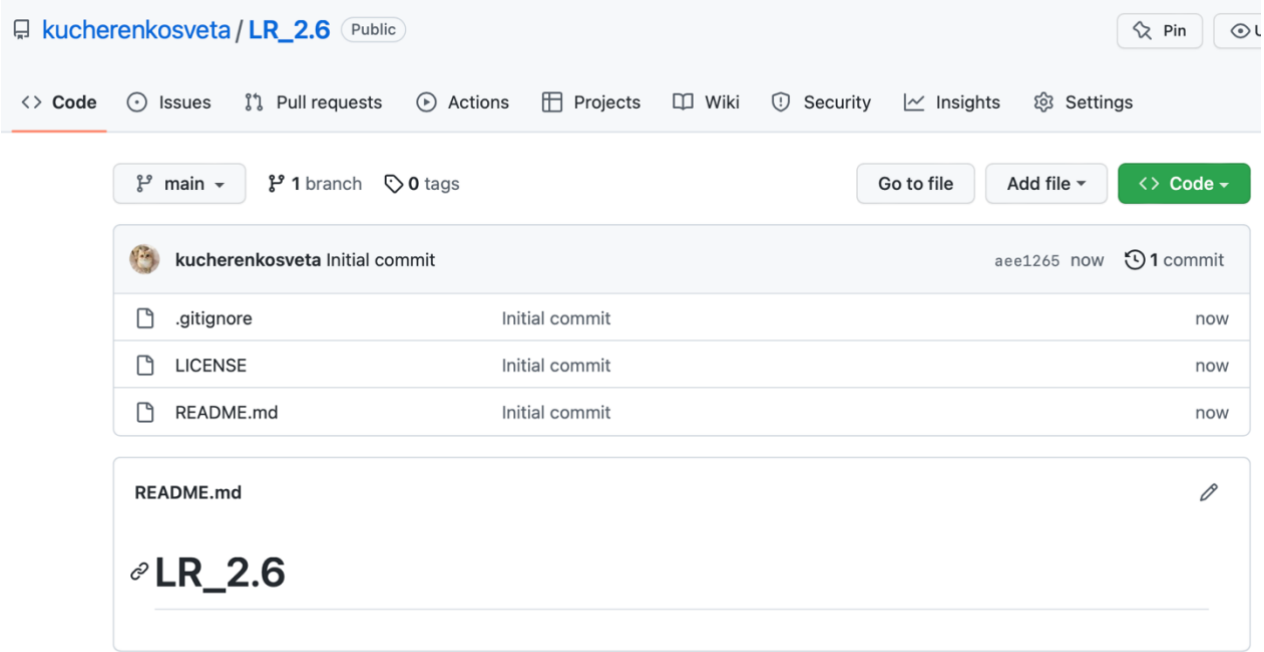

Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
(base) svetik@MacBook-Air-Svetik Laba9 % git clone https://github.com/kucherenko
sveta/LR_2.6.git
Cloning into 'LR_2.6'...
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.
```

Рисунок 2 – Клонирование репозитория

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

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

```
(base) svetik@MacBook-Air-Svetik LR_2.6 % 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? [/Users/svetik/Desktop/Laba9/LR_2.6/.git/hooks]
(base) svetik@MacBook-Air-Svetik LR_2.6 %
```

Рисунок 3 – Организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

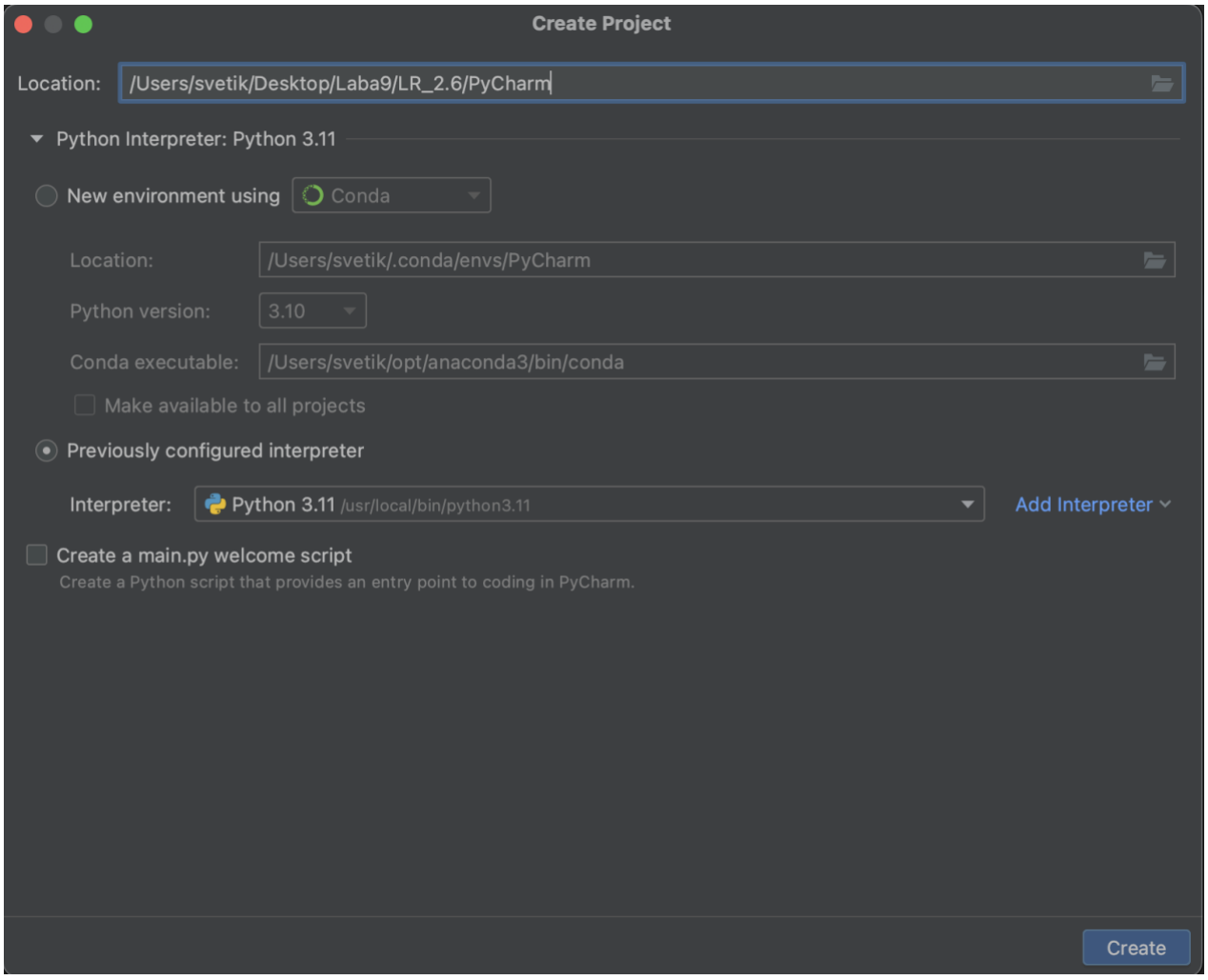

Рисунок 4 – Создание проекта PyCharm в папке репозитория

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.

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

import sys
from datetime import date

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

    # Организовать бесконечный цикл запроса команд.     while True:
    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.

```
/usr/local/bin/python3.11 /Users/svetik/Desktop/Laba9/LR_2.6/PyCharm/pr1.py
>>> add
Фамилия и инициалы? Кучеренко С.Ю.
Должность? Старший куратор
Год поступления? 2022
>>> add
Фамилия и инициалы? Трушева В.О.
Должность? Руководитель Пресс-Центра
Год поступления? 2022
>>> add
Фамилия и инициалы? Гасанов Г.М.
Должность? Штабист
Год поступления? 2014
>>> list
+------+--------------------------------+----------------------+----------+
|  №   |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Гасанов Г.М.                   | Штабист              |     2014 |
|    2 | Кучеренко С.Ю.                 | Старший куратор      |     2022 |
|    3 | Трушева В.О.                   | Руководитель Пресс-Центра |     2022 |
+------+--------------------------------+----------------------+----------+
>>> selsect 2
>>> Неизвестная команда selsect 2
select 2
   1: Гасанов Г.М.
>>> select 10
Работники с заданным стажем не найдены.
```

Рисунок 5 – 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.

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

if __name__ == '__main__':
    school = {'1А': 17, '2В': 30, '6Б': 26, '11': 16, '10': 18}
    print('Первоначальные классы:', school)
    # В одном из классов изменилось количество учащихся
    school['1А'] = 40

    # В школе появился новый класс
    school['5В'] = 20

    # В школе был расформирован (удален) другой класс.
    del school['1А']
    print('Классы после изменений:', school)

    # Общее количество учащихся в школе
    sum = sum([i for i in school.values()])
    print('Общее количество учащихся в школе:', sum)
```

```
/usr/local/bin/python3.11 /Users/svetik/Desktop/Laba9/LR_2.6/PyCharm/ind.p
Первоначальные классы: {'1А': 17, '2В': 30, '6Б': 26, '11': 16, '10': 18}
Классы после изменений: {'2В': 30, '6Б': 26, '11': 16, '10': 18, '5В': 20}
Общее количество учащихся в школе: 110
```

Рисунок 6 – Результат работы программы

10. Зафиксируйте сделанные изменения в репозитории.

```
(base) svetik@MacBook-Air-Svetik LR_2.6 % git add .
(base) svetik@MacBook-Air-Svetik LR_2.6 % git commit -m 'add ind1'
[develop ded299f] add ind1
 8 files changed, 158 insertions(+)
 create mode 100644 PyCharm/.idea/.gitignore
 create mode 100644 PyCharm/.idea/PyCharm.iml
 create mode 100644 PyCharm/.idea/inspectionProfiles/profiles_settings.xml
 create mode 100644 PyCharm/.idea/misc.xml
 create mode 100644 PyCharm/.idea/modules.xml
 create mode 100644 PyCharm/.idea/vcs.xml
 create mode 100644 PyCharm/ind.py
 create mode 100644 PyCharm/pr1.py
(base) svetik@MacBook-Air-Svetik LR_2.6 %
```

Рисунок 7 – Фиксирование изменений в репозитории

11. Решите задачу: создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод items(), с с помощью

полученного объекта dict_items создайте новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.

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

if __name__ == '__main__':
    dct = {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
    print('Исходный словарь:', dct)
    d = dct.items()
    print('Применение метода items():', d)
    swapped = dict(map(reversed, d))
    print('Обратный словарь:',swapped)
```

```
/usr/local/bin/python3.11 /Users/svetik/Desktop/Laba9/LR_2.6/PyCharm/ind2.
Исходный словарь: {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
Применение метода items(): dict_items([(1, 'one'), (2, 'two'), (3, 'three'
Обратный словарь: {'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}
```

Рисунок 8 – Результат работы программы

12. Зафиксируйте сделанные изменения в репозитории.

```
(base) svetik@MacBook-Air-Svetik LR_2.6 % git add .
(base) svetik@MacBook-Air-Svetik LR_2.6 % git commit -m 'add ind2'
[develop 1e4eb78] add ind2
 2 files changed, 10 insertions(+)
 rename PyCharm/{ind.py => ind1.py} (100%)
 create mode 100644 PyCharm/ind2.py
```

Рисунок 9 – Фиксирование изменений в репозитории

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

Вариант 9. Использовать словарь, содержащий следующие ключи: название начального пункта маршрута; название конечного пункта маршрута; номер маршрута. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в список, состоящий из словарей заданной структуры;
- записи должны быть упорядочены по номерам маршрутов;

- вывод на экран информации о маршруте, номер которого введен с клавиатуры; если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

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

import sys

if __name__ == '__main__':
    # Маршруты
    ways = []

    # 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 маршруте.
            start = input('Название начального маршрута: ')
            finish = input('Название конечного маршрута: ')
            num = int(input('Номер маршрута: '))

            # Создать словарь.
            way = {
                'start': start,
                'finish': finish,
                'num': num,
            }

            # Добавить словарь в список.
            ways.append(way)
            # Отсортировать список в случае необходимости.
            if len(way) > 1:
                ways.sort(key=lambda item: item.get(num, ' '))

        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 20,
                '-' * 20,
                '-' * 15
                )
            print(line)
            print(
                '| {:^4} | {:^20} | {:^20} | {:^15} |'.format(
                    "No",
                    "Название начального маршрута",
                    "Название конечного маршрута",
                    "Номер маршрута"
                )
            )
            print(line)

            # Вывести данные о всех маршрутах
```

```
            for idx, way in enumerate(ways, 1):
                print(
                    '| {:>4} | {:<20} | {:<20} | {:>15} |'.format(
                        idx,
                        way.get('start', ''),
                        way.get('finish', ''),
                        way.get('num', 0)
                    )
                )
            print(line)


        elif command == 'find':
            f = input('Введите номер маршрута: ')
            for way in ways:
                flag = 1
                if f in str(way.values()):
                    flag = 0
                    print('Маршрут найден:')
                    print(
                        f"Название начального маршрута:
{way.get('start', '')} \n"
                        f"Название конечного маршрута:
{way.get('finish', 0)} \n"
                        f"Номер маршрута: {way.get('num', 0)}"
                    )
                    continue

            if flag == 1:
                print('Маршрут не найден')

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить маршрут;")
            print("list - вывести список маршрутов;")
            print("find - вывод информации о маршруте;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

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

```
>>> add
Название начального маршрута: Пушкина
Название конечного маршрута: Кулакова
Номер маршрута: 42
>>> add
Название начального маршрута: Серова
Название конечного маршрута: Учительский проезд
Номер маршрута: 29
>>> list
+------+--------------------------------+--------------------------------+-----------------+
|  No  |  Название начального маршрута  |  Название конечного маршрута   | Номер маршрута  |
+------+--------------------------------+--------------------------------+-----------------+
|    1 | Пушкина                        | Кулакова                       |              42 |
|    2 | Серова                         | Учительский проезд             |              29 |
+------+--------------------------------+--------------------------------+-----------------+
>>> help
Список команд:

add - добавить маршрут;
list - вывести список маршрутов;
find - вывод информации о маршруте;
help - отобразить справку;
exit - завершить работу с программой.
>>> find
Введите номер маршрута: 29
Маршрут найден:
Название начального маршрута: Серова
Название конечного маршрута: Учительский проезд
Номер маршрута: 29
>>> find 22
>>> Неизвестная команда find 22
find
Введите номер маршрута: 22
Маршрут не найден
```

Рисунок 10 – Результат работы программы

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-значения 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 nums.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?

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.

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.

romtimestamp(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

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