

**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: «Работа со словарями»

Выполнила:  
студентка 2 курса  
группы Пиж-б-о-21-1  
Джолдошова Мээрим  
Бекболотовна

Ставрополь 2022

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

1. Был создан репозиторий в Github в который были добавлены правила gitignore для работы IDE PyCharm, была выбрана лицензия MIT, сам репозиторий был клонирован на локальный сервер и был организован в соответствии с моделью ветвления git-flow.

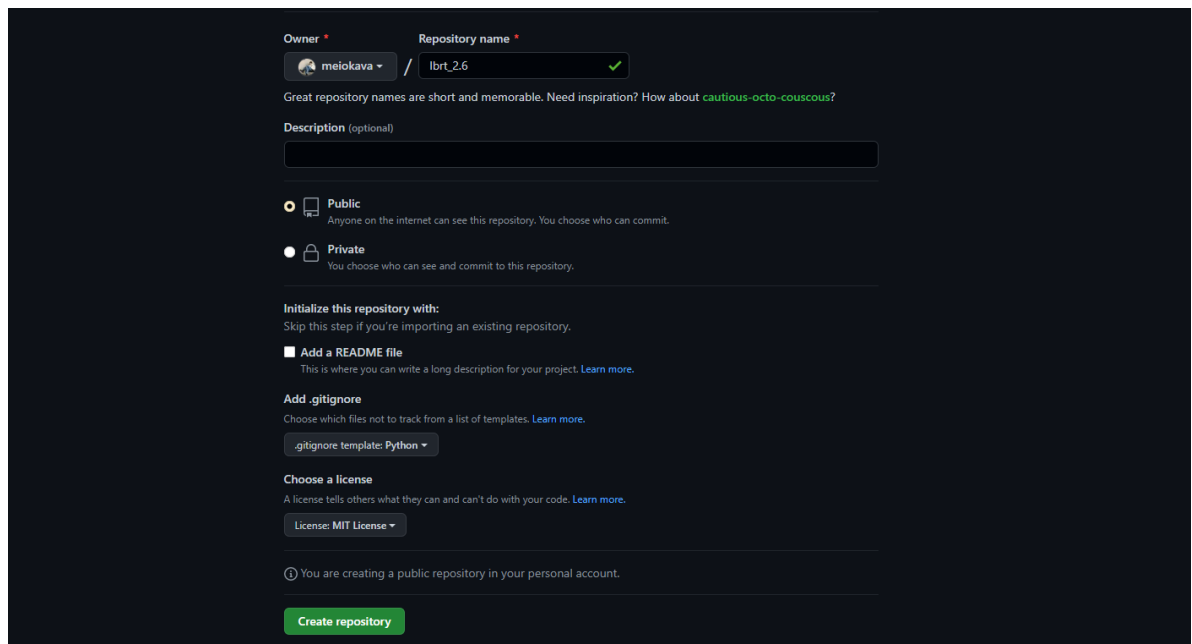


Рисунок 1 – Создание репозитория

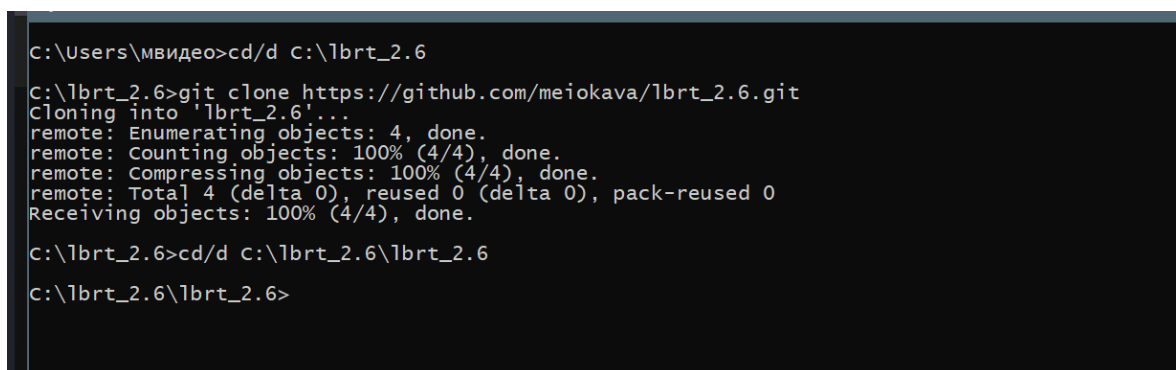


Рисунок 2 – Клонирование репозитория



Рисунок 3 – Изменение gitignore

```
C:\Users\МВидео>cd/d c:\lbrt_2.6
C:\lbrt_2.6>git clone https://github.com/meiokava/lbrt_2.6.git
Cloning into 'lbrt_2.6'...
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:\lbrt_2.6>cd/d c:\lbrt_2.6\lbrt_2.6
C:\lbrt_2.6\lbrt_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? [c:\lbrt_2.6\lbrt_2.6\.git\hooks]
```

Рисунок 4 – Организация репозитория в соответствии с моделью ветвления git-flow

2. Был создана папка PyCharm в которой хранятся примеры из лабораторной работы.

Имя	Дата изменения	Тип	Раз
.git	25.11.2022 21:36	Папка с файлами	
PyCharm	25.11.2022 21:40	Папка с файлами	
.gitignore	25.11.2022 21:30	Исходный файл Git I...	
LICENSE	25.11.2022 21:30	Файл	

Рисунок 4 – Папка PyCharm для примеров

```
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Фамилия и инициалы? Mei O S
Должность? Owner
Год поступления? 2005
>>> list
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
| 1 | Mei O S                  | Owner               | 2005          |
+-----+-----+-----+-----+
>>> select 10
1: Mei O S
>>> exit

Process finished with exit code 0
```

Рисунок 5 – Результат работы программы

### 3. Были выполнены задания.

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.

Код программы:

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

if __name__ == '__main__':
    school = {
        "1a": 30, "1b": 31, "2a": 29, "2b": 28, "3a": 26, "3b": 16,
        "4a": 18, "4b": 34, "5a": 29, "5b": 30, "6a": 30, "6b": 28,
        "7a": 27, "7b": 26, "8a": 25, "8b": 28, "9a": 29, "9b": 27,
        "10a": 29, "10b": 32, "11a": 25, "11b": 24, "5a": 18, "1c": 13
    }

    school["2a"] = 16
    school["1c"] = 21
    del school["8b"]
    print(f"The total number of students enrolled in school is:
    {sum(school.values())}")
```

```
The total number of students enrolled in school is: 599

Process finished with exit code 0
```

Рисунок 6 – Результат работы программы

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.

Код программы:

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

if __name__ == '__main__':
    d = {5: 'five', 8: 'eight', 10: 'ten'}
    swapped = {value: keys for keys, value in d.items()}
    print(d)
    print(swapped)
```

```
C:\Users\мвидео\PycharmProjects\pythonProject48\venv\Scr
{5: 'five', 8: 'eight', 10: 'ten'}
{'five': 5, 'eight': 8, 'ten': 10}

Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

## Индивидуальное задание

### Вариант 5

Использовать словарь, содержащий следующие ключи: название пункта назначения рейса; номер рейса; тип самолета. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения; вывод на экран

пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры; если таких рейсов нет, выдать на дисплей соответствующее сообщение.

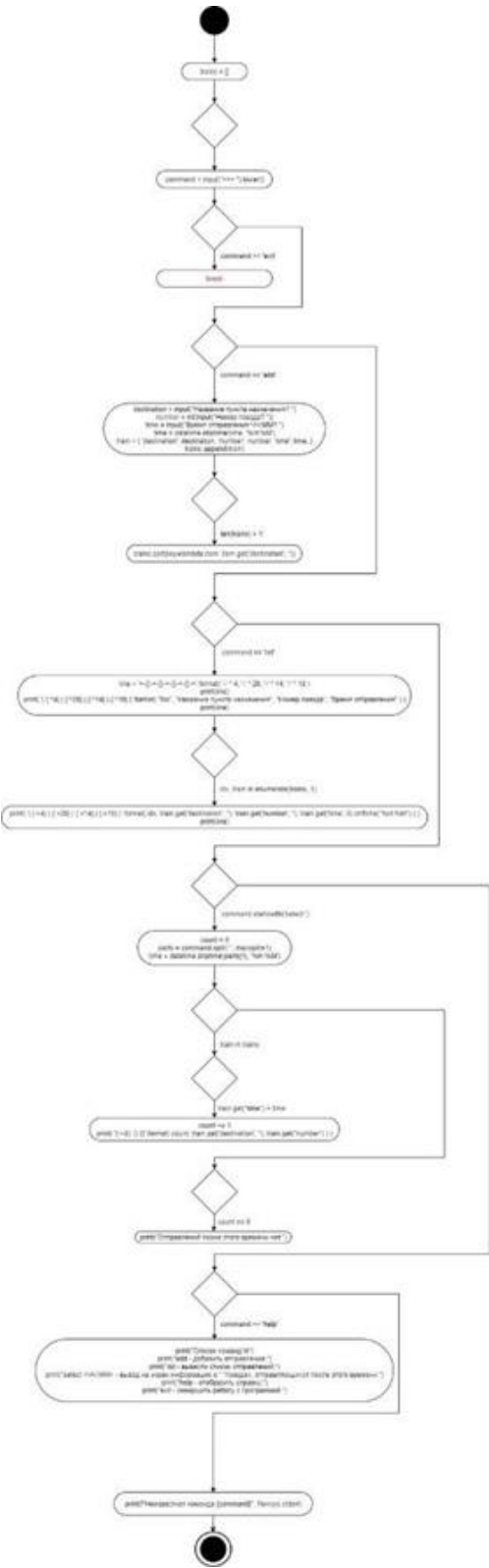


Рисунок 8 – UML диаграмма

```

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

import sys

if __name__ == '__main__':
    flights = []
    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break
        elif command == 'add':
            dst = input("What destination do you need? ")
            nmb = int(input("Which number of the flight do you need? "))
            tpe = input("Which type of plane do you need? ")
            flight = {
                'destination': dst,
                'number_flight': nmb,
                'type_plane': tpe,
            }
            flights.append(flight)
            if len(flights) > 1:
                flights.sort(key=lambda item: item.get('destination', ''))
        elif command == 'list':
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 18
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^18} |'.format(
                    "№",
                    "Destination",
                    "Number of the flight",
                    "Type of the plane"
                )
            )
            print(line)
            for idx, flight in enumerate(flights, 1):
                print(
                    '| {:>4} | {:<30} | {:<20} | {:>18} |'.format(
                        idx,
                        flight.get('destination', ''),
                        flight.get('number_flight', ''),
                        flight.get('type_plane', 0)
                    )
                )
            print(line)

        elif command.startswith('select'):
            t = input("choose type of the plane: ")
            count = 0
            for flight in flights:
                if flight.get('type_plane') == t:
                    count += 1
                    print(
                        '{:>4}: {} {}'.format(
                            count,
                            flight.get('destination', ''),
                            flight.get("number_flight")
                        )
                    )

```

```

        if count == 0:
            print("We couldn't find this type of plane")
    elif command == 'help':
        print("command list:\n")
        print("add - add information about a flight;")
        print("list - display the flight schedule;")
        print("select <type> - select the type of the plane;")
        print("help - show reference;")
        print("exit - leave a program.")
    else:
        print(f"unknown command {command}", file=sys.stderr)

```

```

>>> add
What destination do you need? canada
Which number of the flight do you need? 4
Which type of plane do you need? passangers
>>> list
+-----+-----+-----+-----+
| № | Destination | Number of the flight | Type of the plane |
+-----+-----+-----+-----+
| 1 | canada | 4 | passangers |
+-----+-----+-----+-----+
>>> select
choose type of the plane: passangers
1: canada 4
>>> help
command list:

add - add information about a flight;
list - display the flight schedule;
select <type> - select the type of the plane;
help - show reference;
exit - leave a program.
>>> exit
Process finished with exit code 0

```

Рисунок 9 – Результат работы программы

```

"C:\Program Files\Python310\python.exe" C:\lbrt_2.6\lbrt_2.6\indiv\indiv1.py
>>> add
What destination do you need? canada
Which number of the flight do you need? 4
Which type of plane do you need? passangers
>>> add
What destination do you need? america
Which number of the flight do you need? 5
Which type of plane do you need? official
>>> list
+-----+-----+-----+-----+
| № | Destination | Number of the flight | Type of the plane |
+-----+-----+-----+-----+
| 1 | america | 5 | official |
| 2 | canada | 4 | passangers |
+-----+-----+-----+-----+
>>> select
choose type of the plane: official
1: america 5
>>> |

```

Рисунок 10 – Результат работы программы с двумя рейсами



```
c:\lbrt_2.6\lbrt_2.6>git add .
c:\lbrt_2.6\lbrt_2.6>git commit -m "new completed tasks"
[develop cb0c59e] new completed tasks
4 files changed, 196 insertions(+)
create mode 100644 PyCharm/example1.py
create mode 100644 indiv/indiv1.py
create mode 100644 tasks/task1.py
create mode 100644 tasks/task2.py
c:\lbrt_2.6\lbrt_2.6>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Рисунок 11 – Коммит изменений

```
c:\lbrt_2.6\lbrt_2.6>git merge develop
Updating 637f134..cb0c59e
Fast-forward
 PyCharm/example1.py | 99 +++++
 indiv/indiv1.py      | 75 +++++
 tasks/task1.py       | 14 +++++
 tasks/task2.py       | 8  +++++
4 files changed, 196 insertions(+)
create mode 100644 PyCharm/example1.py
create mode 100644 indiv/indiv1.py
create mode 100644 tasks/task1.py
create mode 100644 tasks/task2.py
c:\lbrt_2.6\lbrt_2.6>git push
To https://github.com/meiokava/lbrt_2.6.git
 ! [rejected]        main -> main (fetch first)
```

Рисунок 12 – Слияние веток main и develop

```
c:\lbrt_2.6\lbrt_2.6>git push
To https://github.com/meiokava/lbrt_2.6.git
```

Рисунок 13 – Пуш изменений на удаленный сервер

Вывод: в ходе лабораторной работы были приобретены навыки по работе со словарями при написании программ с помощью языка программирования Python версии 3.x.

#### Контрольные вопросы:

1. Что такое словари в языке Python?

Словари в Python – это изменяемые отображения ссылок на объекты, доступные по ключу.

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

Функция len() возвращает длину (количество элементов) в объекте. Аргумент может быть последовательностью, такой как строка, байты, кортеж,

список или диапазон или коллекцией (такой как словарь, множество или неизменяемое множество).

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

Самый очевидный вариант обхода словаря — это попытаться напрямую запустить цикл `for` по объекту словаря, так же как мы делаем это со списками, кортежами, строками и любыми другими итерируемыми объектами. `for something in currencies: print(something)`

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

С помощью метода `.get()`

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

С помощью функции `dict.update()`

### 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()`. Вот пример программы, которая делает именно это:

```
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)
```

Функция `zip` возвращает следующее:

```
[('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? `Datetime` — важный элемент любой программы, написанной на Python. 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

Как получить текущие дату и время?

```
import datetime  
  
dt_now = datetime.datetime.now()  
  
print(dt_now)
```

Результат:

2022-09-11 15:43:32.249588

Получить текущую дату:

```
from datetime import date  
  
current_date = date.today()  
  
print(current_date)
```

Результат:

2022-09-11

Получить текущее время:

```
import datetime  
  
current_date_time = datetime.datetime.now()  
  
current_time = current_date_time.time()
```

```
print(current_time)
```

Результат:

15:51:05.627643