

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 ЛАБОРАТОРНОЙ РАБОТЕ №2.6
дисциплины «Введение в специальность»

Выполнил:
Шамс Вайсудин
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
очная форма обучения

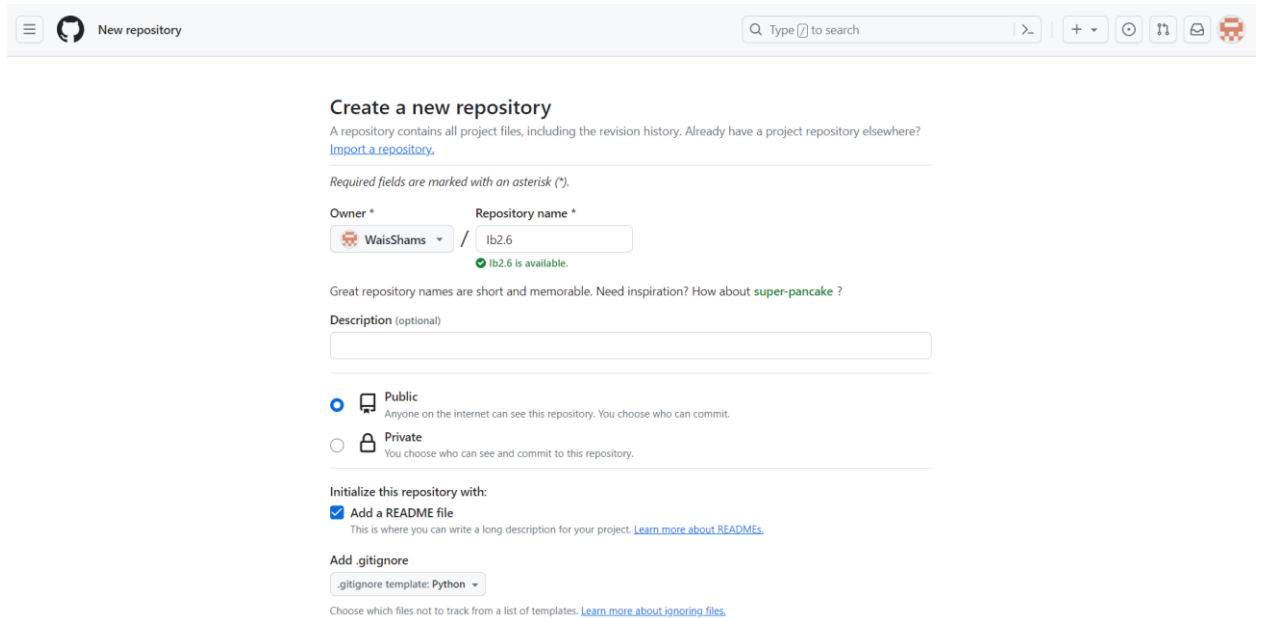
(подпись)

Отчет защищен с оценкой _____ Дата
защиты _____

Ставрополь, 202^г г.

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

1. Создал общедоступный репозиторий на GitHub с MIT



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 * Repository name *

WaisShams / lb2.6

lb2.6 is available.

Great repository names are short and memorable. Need inspiration? How about [super-pancake](#) ?

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.

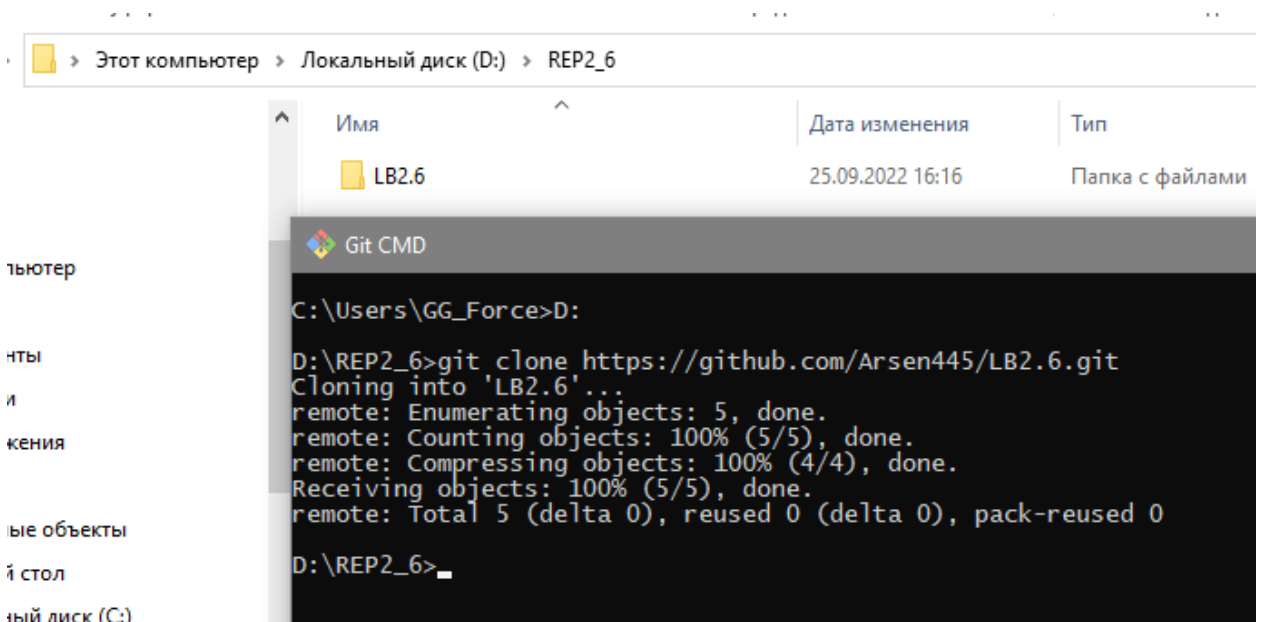
Initialize this repository with:

☒ 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.](#)

2. Выполнил клонирование созданного репозитория.



3. Организовал свой репозиторий в соответствии с моделью ветвления git-flow. (Перешел с главной main на develop)

```

D:\REP2_6\LB2.6>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? [D:/REP2_6/LB2.6/.git/hooks]

D:\REP2_6\LB2.6> git status
On branch develop
nothing to commit, working tree clean

D:\REP2_6\LB2.6>

```

4. Создал проект PyCharm в папке репозитория.

Этот компьютер > Локальный диск (D:) > REP2_6 > LB2.6 > 111 >

	Имя	Дата изменения	Тип	Размер
Директория Python	.idea	25.09.2022 17:07	Папка с файлами	
	venv	25.09.2022 17:07	Папка с файлами	
	main.py	25.09.2022 17:07	Python File	1 КБ

5. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

(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": 8, "1б": 12, "2б": 12, "6а": 4, "7в": 19}
    print(school, "количество учащихся в разных классах")
    school["1а"] = 22
    print(school, "1 изменение")
    school["1г"] = 26
    print(school, "новый класс")
    del school["2б"]
    print(school, "класс расформирован")
    k = sum(school.values())
    print(k, "кол-во учеников")
```

IDLE Shell 3.9.13

File Edit Shell Debug Options Window Help

Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: D:\REP2 6\LB2.6\1zad.py =====

{'1а': 8, '1б': 12, '2б': 12, '6а': 4, '7в': 19} количество учащихся в разных классах

{'1а': 22, '1б': 12, '2б': 12, '6а': 4, '7в': 19} 1 изменение

{'1а': 22, '1б': 12, '2б': 12, '6а': 4, '7в': 19, '1г': 26} новый класс

{'1а': 22, '1б': 12, '6а': 4, '7в': 19, '1г': 26} класс расформирован

83 кол-во учеников



8. Зафиксируйте сделанные изменения в репозитории.

```

D:\REP2_6\LB2.6>git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   1.png
        new file:   1zad.py
        new file:   2zad.py

D:\REP2_6\LB2.6>git commit -m "new"
[develop 981ed95] new
 3 files changed, 10 insertions(+)
 create mode 100644 1.png
 create mode 100644 1zad.py
 create mode 100644 2zad.py

D:\REP2_6\LB2.6>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 14.62 KiB | 14.62 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Arsen445/LB2.6.git
 d6888d4..981ed95  develop -> develop

D:\REP2_6\LB2.6>
  
```

9. 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.

```

sid 1
Mani 2
Shiba 3
>>>
===== RESTART: D:
sid 1
Mani 2
Shiba 3
{1: 'sid', 2: 'Mani', 3: 'Shiba'}
>>>
===== RESTART: D:
{1: 'sid', 2: 'Mani', 3: 'Shiba'}
{'sid': 1, 'Mani': 2, 'Shiba': 3}
>>>

```

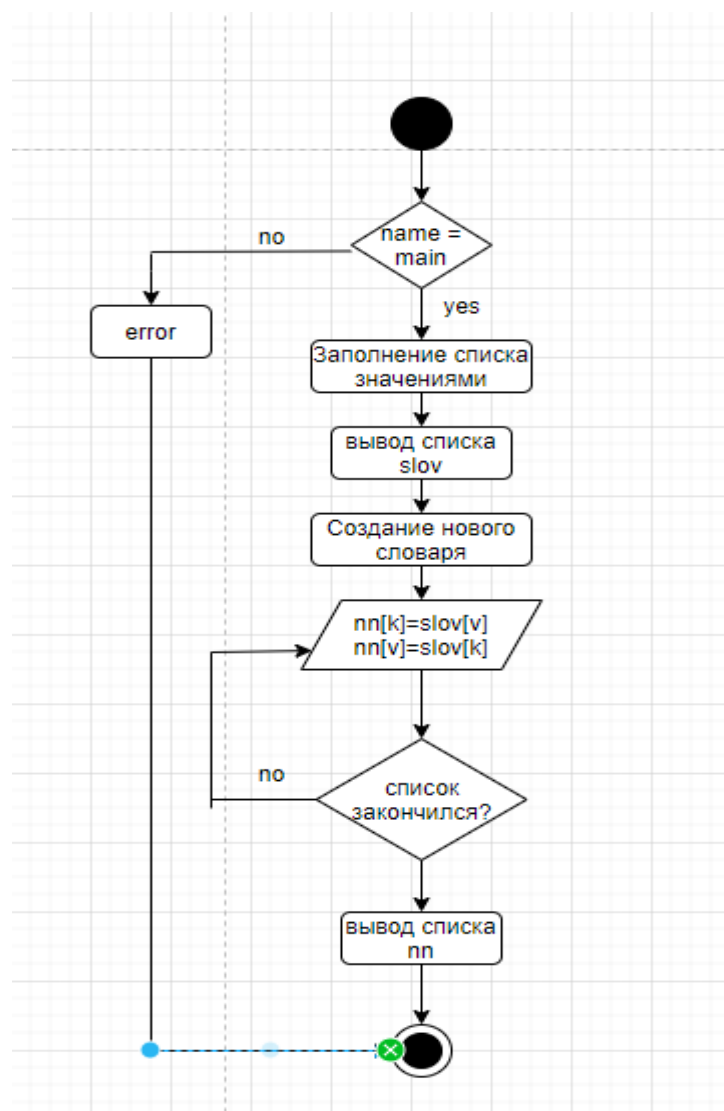
```

*2zad.py - D:\REP2_6\LB2.6\2zad.py (3.9.13)*
File Edit Format Run Options Window Help

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

if __name__ == '__main__':
    slov = {1:"sid", 2:"Mani", 3:"Shiba"}
    print (slov)
    nn = {}
    for k, v in slov.items():
        nn[v]=k
    print (nn)

```



10. Зафиксируйте сделанные изменения в репозитории.

```
D:\REP2_6\LB2.6>git add 2zad.py
D:\REP2_6\LB2.6>git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2zad.py

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   1zad.py

D:\REP2_6\LB2.6>git add 1zad.py
D:\REP2_6\LB2.6>git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   1zad.py
        modified:   2zad.py

D:\REP2_6\LB2.6>git commit -m "new3"
[develop 7cac933] new3
 2 files changed, 27 insertions(+), 10 deletions(-)
 rewrite 1zad.py (100%)

D:\REP2_6\LB2.6>git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 635 bytes | 635.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Arsen445/LB2.6.git
   ff6b54b..7cac933  develop -> develop

D:\REP2_6\LB2.6>git status
On branch develop
Your branch is up to date with 'origin/develop'.
```

11. Использовать словарь, содержащий следующие ключи: название пункта назначения; номер поезда; время отправления. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по номерам поездов; вывод на экран информации о поезде, номер которого введен с клавиатуры; если таких поездов нет, выдать на дисплей соответствующее сообщение.

12. Зафиксируйте сделанные изменения в репозитории.

```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2zad.py
        new file:   indiv.py

D:\REP2_6\LB2.6>git commit -m "new4"
[develop 1f77f53] new4
 2 files changed, 108 insertions(+), 4 deletions(-)
 create mode 100644 indiv.py

D:\REP2_6\LB2.6>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.53 KiB | 1.53 MiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Arsen445/LB2.6.git
   7cac933..1f77f53  develop -> develop

D:\REP2_6\LB2.6>
```

The screenshot shows a Python IDE with the file `indiv.py` open. The code is a menu-driven program for managing train schedules. It includes a `main` function with a `while True` loop that handles commands like `help`, `list`, `add`, and `exit`. The `add` function prompts the user for the number of trains, their destinations, departure times, and numbers, and then prints a formatted table of the schedule.

The terminal window on the right shows the execution of the program. It displays the `RESTART` message and the program's output, including the schedule table for 3 trains:

№	Едет в	№ поезда	Время отправления
1	gn	2	9
2	fdgn	3	4
3	fnh	3	7

13. Выполните слияние ветки для разработки с веткой main/master.

```

D:\REP2_6\LB2.6>git merge develop
Merge made by the 'ort' strategy.
 1.png      | Bin 0 -> 14887 bytes
 1zad.py    | 15 ++++++
 2.png      | Bin 0 -> 19557 bytes
 2zad.py    | 11 +++++
 indiv.py   | 89 ++++++
 main.py    | 144 ++++++
6 files changed, 259 insertions(+)
create mode 100644 1.png
create mode 100644 1zad.py
create mode 100644 2.png
create mode 100644 2zad.py
create mode 100644 indiv.py
create mode 100644 main.py

D:\REP2_6\LB2.6>git status
On branch main
Your branch is ahead of 'origin/main' by 8 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

D:\REP2_6\LB2.6>git commit -m "new7"
On branch main
Your branch is ahead of 'origin/main' by 8 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

D:\REP2_6\LB2.6>git push
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 222 bytes | 222.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Arsen445/LB2.6.git
   b673ffc..054fc12  main -> main
D:\REP2_6\LB2.6>

```

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

Вывод: Изучил Словари в python