

**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»

Выполнил:
студент 2 курса
группы Пиж-б-о-21-1
Коныжев Максим
Викторович

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.

1

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 MaxDrill ▾

Repository name *

/ lab2_6 ✓

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

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:

Skip this step if you're importing an existing repository.

☒ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore


Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▾

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

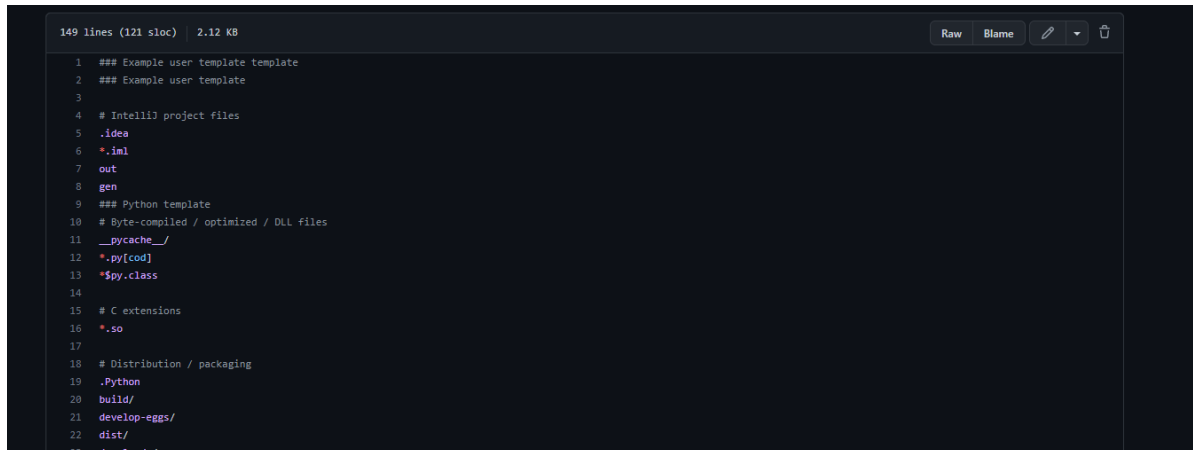
This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

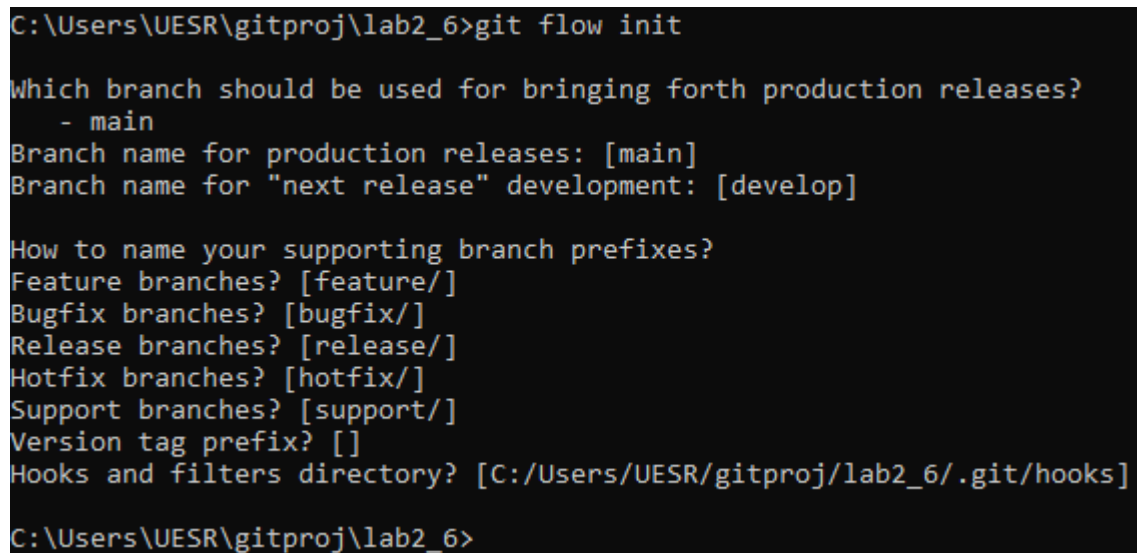
Рисунок

Создание репозитория

A screenshot of a code editor showing a .gitignore file. The file contains a list of patterns to be ignored by Git, including IDE files, Python templates, compiled files, C extensions, and distribution files. The editor interface shows 149 lines and 2.12 KB. The patterns listed are: ### Example user template template, ### Example user template, # IntelliJ project files, .idea, *.iml, out, gen, ## Python template, # Byte-compiled / optimized / DLL files, __pycache_/, *.py[co], *.py.class, # C extensions, *.so, # Distribution / packaging, .Python, build/, develop-eggs/, dist/, and download/.

```
149 lines (121 sloc) | 2.12 KB
1  ### Example user template template
2  ### Example user template
3
4  # IntelliJ project files
5  .idea
6  *.iml
7  out
8  gen
9  ## Python template
10 # Byte-compiled / optimized / DLL files
11 __pycache_/
12 *.py[co]
13 *.py.class
14
15 # C extensions
16 *.so
17
18 # Distribution / packaging
19 .Python
20 build/
21 develop-eggs/
22 dist/
23 download/
```

Рисунок 2 – Изменение gitignore

A screenshot of a terminal window showing the output of the 'git flow init' command. The command prompts the user for various branch names and prefixes, which are then entered. The final output shows the directory for hooks and filters.

```
C:\Users\UESR\gitproj\lab2_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:/Users/UESR/gitproj/lab2_6/.git/hooks]

C:\Users\UESR\gitproj\lab2_6>
```

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

```
C:\Users\UESR\Desktop\2.1\proj\venv\Scripts\python.exe C:/Users/UESR/Desktop/2.1
>>> add
Фамилия и инициалы? Max K
Должность? director
Год поступления? 2000
>>> list
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Max K                    | director           |      2000     |
+-----+-----+-----+-----+
>>> select 5
      1: Max K
>>> exit

Process finished with exit code 0
```

Рисунок 5 – Результат работы программы

3. Выполнил 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 7

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 времени отправления поезда; вывод на экран информации о поездах, направляющихся в пункт, название которого введено с клавиатуры; если таких поездов нет, выдать на дисплей соответствующее сообщение.

```

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

import sys

if __name__ == '__main__':
    train = []
    while True:
        info = input(">>> ").lower()
        if info == 'exit':
            break

        elif info == 'add':
            dest = input("What destination do you need? ")
            numb = int(input("What number of the train? "))
            date = float(input("What time do you need? "))
            trainDct = {
                'destination': dest,
                'number_train': numb,
                'flight_date': date,
            }
            train.append(trainDct)
            if len(train) > 1:
                train.sort(key=lambda item: item.get('flight_date', ''))

        elif info == 'list':
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 18
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^18} |'.format(
                    "№",
                    "Destination",
                    "Number of the train",
                    "Flight Date"
                )
            )
            print(line)
            for i, item in enumerate(train, 1):
                print(
                    '| {:>4} | {:<30} | {:<20} | {:>18} |'.format(
                        i,
                        item.get('destination', ''),
                        item.get('number_train', ''),
                        item.get('flight_date', 0)
                    )
                )
            print(line)

        elif info.startswith('select'):
            t = input("enter your destination: ")
            count = 0
            for i in train:
                if i.get('destination') == t:
                    count += 1
            print(

```

```

        '{:>4}: {} {} {}'.format(
            count,
            i.get('destination', ''),
            i.get('flight_date'),
            i.get("number_train")
        )
    )

    if count == 0:
        print("We couldn't find this type of plane")

    elif info == 'help':
        print("command list:\n")
        print("add - add information about a flight;")
        print("list - display the flight schedule;")
        print("select <type> - select destination;")
        print("help - show reference;")
        print("exit - leave a program.")
    else:
        print(f"unknown command {info}", file=sys.stderr)

```

```

>>> add
What destination do you need? london
What number of the train? 15
What time do you need? 22.12
>>> add
What destination do you need? mos
What number of the train? 16
What time do you need? 23.12
>>> add
What destination do you need? keln
What number of the train? 15
What time do you need? 21.12
>>>
unknown command
>>> list
+-----+-----+-----+-----+
| № | Destination | Number of the train | Flight Date |
+-----+-----+-----+-----+
| 1 | keln | 15 | 21.12 |
| 2 | london | 15 | 22.12 |
| 3 | mos | 16 | 23.12 |
+-----+-----+-----+-----+
>>> select
enter your destination: london
1: london 22.12 15
>>> exit

```

Рисунок 8 – Результат работы программы с двумя рейсами

```

C:\Users\UESR\gitproj\lab2_6>git add .

C:\Users\UESR\gitproj\lab2_6>git commit -m "Add proj"
[develop aa04d41] Add proj
4 files changed, 201 insertions(+)
create mode 100644 proj/idz9.py
create mode 100644 proj/lab9_obsh.py
create mode 100644 proj/num1_lr9.py
create mode 100644 proj/num2_lr9.py

C:\Users\UESR\gitproj\lab2_6>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\UESR\gitproj\lab2_6>

```

Рисунок 9 – Коммит изменений

```
C:\Users\UESR\gitproj\lab2_6>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\UESR\gitproj\lab2_6>git merge develop
Updating 78162c5..aa04d41
Fast-forward
 proj/idz9.py      | 76 +++++
 proj/lab9_obsh.py | 103 +++++
 proj/num1_lr9.py  | 14 +++++
 proj/num2_lr9.py  |  8 +++++
 4 files changed, 201 insertions(+)
 create mode 100644 proj/idz9.py
 create mode 100644 proj/lab9_obsh.py
 create mode 100644 proj/num1_lr9.py
 create mode 100644 proj/num2_lr9.py

C:\Users\UESR\gitproj\lab2_6>
```

Рисунок 10 – Слияние веток main и develop

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

