

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
«Основы программной инженерии»

Выполнил:

Разворотников Денис Сергеевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

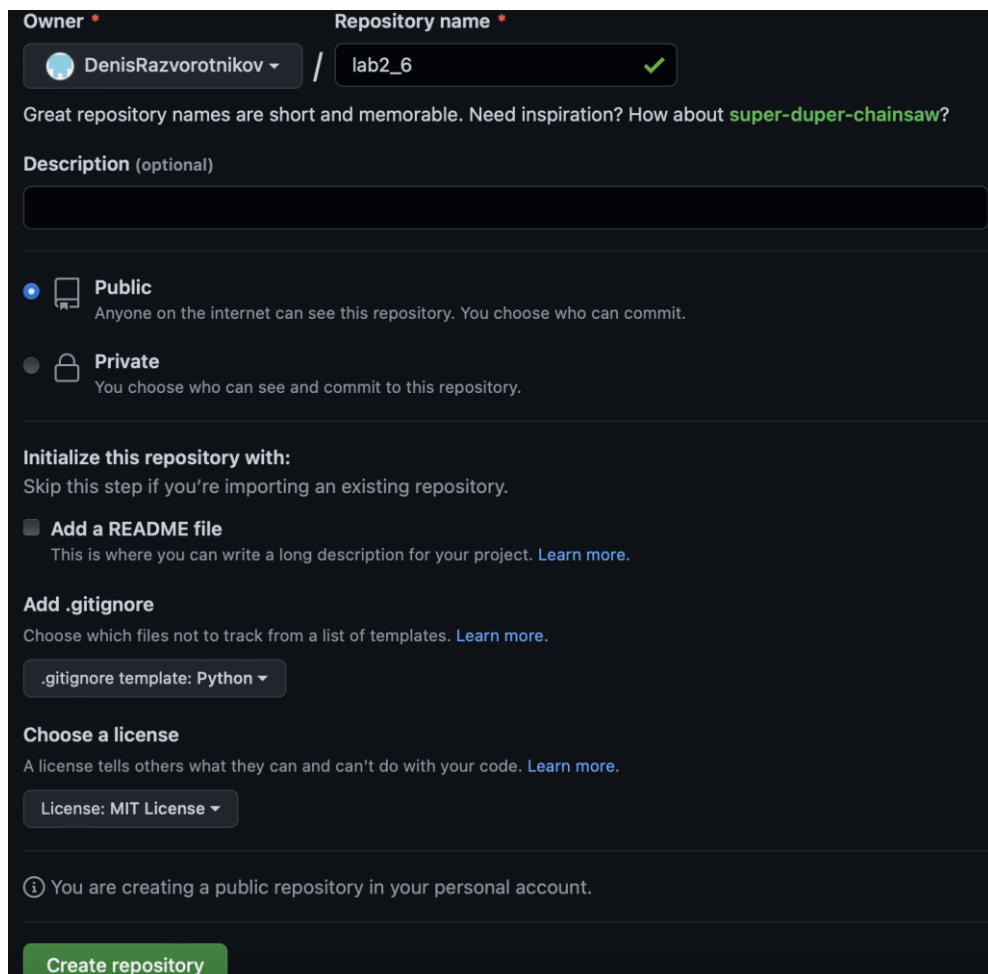
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

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.



Owner * DenisRazvorotnikov / Repository name * lab2_6 ✓

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

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 ▼

ⓘ You are creating a public repository in your personal account.

[Create repository](#)

Рисунок 1 – Создание репозитория

```
denisrazvorotnikov@Air-Denis desktop % git clone https://github.com/DenisRazv
tnikov/lab2_6.git
Cloning into 'lab2_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.
denisrazvorotnikov@Air-Denis desktop %
```

Рисунок 2 – Клонирование репозитория

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

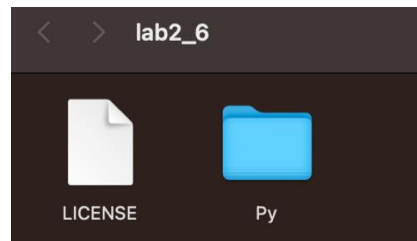


Рисунок 3 – Папка PyCharm для примеров

```
/Users/denisrazvorotnikov/Desktop/lab2_6/Py/bin/python /Users/denisrazvorotnikov/Desktop
>>> 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? Razvorotnikov D S
Должность? Gamer
Год поступления? 2001
>>> list
+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+
|  1 | Razvorotnikov D S      |      Gamer          |   2001  |
+-----+-----+-----+
>>> select 10
1: Razvorotnikov D S
>>> exit
```

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())}")
```

```
/Users/denisrazvorotnikov/Desktop/lab2_6/Py/bin/python /Us
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)
```

```
/Users/denisrazvorotnikov/Desktop/lab2_6/Py/
{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.

```
#!/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? Russia
Which number of the flight do you need? 5
Which type of plane do you need? Pasagers
>>> list
+-----+-----+-----+-----+
| № | Destination | Number of the flight | Type of the plane |
+-----+-----+-----+-----+
| 1 | Russia | 5 | Pasagers |
+-----+-----+-----+-----+

>>> select
choose type of the plane: Pasagers
1: Russia 5

>>> 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 8 – Результат работы программы

```

7/users/denisrazvorotnikov/desktop/lab2_6/1.py/bin/python 7/users/denisrazvorotnikov/desktop/lab2_6/1.py
>>> add
What destination do you need? Russia
Which number of the flight do you need? 5
Which type of plane do you need? passagers
>>> add
What destination do you need? Stav
Which number of the flight do you need? 3
Which type of plane do you need? Fast
>>> list
+-----+-----+-----+-----+
| № | Destination | Number of the flight | Type of the plane |
+-----+-----+-----+-----+
| 1 | Russia | 5 | passagers |
| 2 | Stav | 3 | Fast |
+-----+-----+-----+-----+

>>> select
choose type of the plane: Fast
1: Stav 3
>>>

```

Рисунок 9 – Результат работы программы с двумя рейсами

```

[denisrazvorotnikov@Air-Denis lab2_6 % git add .
[denisrazvorotnikov@Air-Denis lab2_6 % git commit -m "Tasks"
[develop 84e0b91] Tasks
4 files changed, 187 insertions(+)
create mode 100644 1.py
create mode 100644 2.py
create mode 100644 3.py
create mode 100644 idz.py
denisrazvorotnikov@Air-Denis lab2_6 %

```

Рисунок 10 – Коммит изменений

```
denisrazvorotnikov@Air-Denis lab2_6 % git merge develop
Updating 84fced9..84e0b91
Fast-forward
 1.py | 90 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 2.py | 14 ++++++++
 3.py | 8 +++++
 idz.py | 75 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 4 files changed, 187 insertions(+)
 create mode 100644 1.py
 create mode 100644 2.py
 create mode 100644 3.py
 create mode 100644 idz.py
```

Рисунок 11 – Слияние веток main и develop

```
denisrazvorotnikov@Air-Denis lab2_6 % git push git@github.com:DenisRazvorotnikov/lab2_6.git
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 2.95 KiB | 2.95 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:DenisRazvorotnikov/lab2_6.git
 84fced9..84e0b91 main -> main
denisrazvorotnikov@Air-Denis lab2_6 %
```

Рисунок 12 – Пуш изменений на удаленный сервер

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