

**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: «Программирование на Python»

Тема: «Работа со словарями в языке Python»

Выполнил: студент 2 курса

группы ИВТ-б-о-21-1

Урусов Максим Андреевич

Ставрополь 2022

## Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.6» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствии с моделью ветвления git-flow.

**Create a new repository**

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

---

**Owner \*** **Repository name \***

MaksimUrusov / Lab 2.6 ✓

Great repository names are short and to the point. Your new repository will be created as Lab-2.6. about [super-computing-machine?](#)

**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 1.1 Создание репозитория

```
C:\Users\den-n>git clone https://github.com/MaksimUrusov/Laba2.6.git
Cloning into 'Laba2.6'...
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 1.2 Клонирование репозитория

```
C:\Users\den-n\Laba2.6>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/den-n/Laba2.6/.git/hooks]

C:\Users\den-n\Laba2.6>
```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления

### git-flow



```
.gitignore - Блокнот
Файл Правка Формат Вид Справка
.idea/
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio,
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml
```

Рисунок 1.4 Изменение .gitignore

2. Создал проект PyCharm в папке репозитория, проработал примеры ЛР.

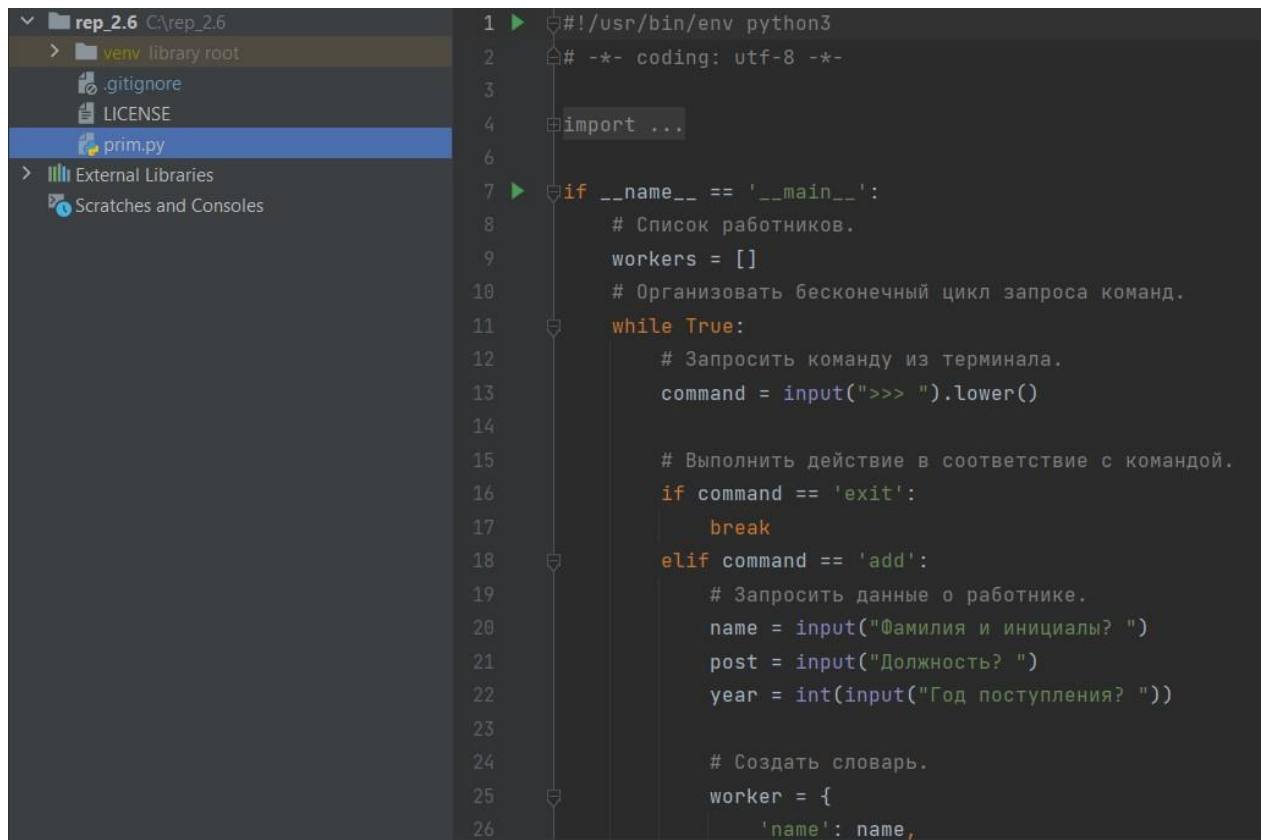


Рисунок 2.1 Создание проекта в PyCharm

```
C:\rep_2.6\venv\Scripts\python.exe C:/rep_2.6/prim.py
>>> 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? Zolo I S
Должность? Owner
Год поступления? 2004
>>> list
+-----+-----+-----+-----+
| № |           Ф.И.О.           | Должность | Год |
+-----+-----+-----+-----+
|  1 | Zolo I S                   | Owner     | 2004 |
+-----+-----+-----+-----+
>>> 200
>>> Неизвестная команда 200
select 200
Работники с заданным стажем не найдены.
>>> select 10
1: Zolo I S
>>>
```

Рисунок 2.2 Рез-т выполнения 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.

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.

```
prim.py x zadaniya.py x
1  ▶ #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  ▶ if __name__ == '__main__':
6      # 1
7      school = {'1a': 20, '2b': 24, '11a': 15, '11b': 10}
8      school['1a'] = 59
9      school['2b'] = 127
10
11      del school['2b']
12      S = sum(school.values())
13      print(school, "\nКоличество учеников: ", S, '\n')
14
15      # 2
16      a = {1: 'one', 2: 'two', 3: 'three'}
17      inverse_a = {}
18      for k, v in a.items():
19          inverse_a[v] = k
20      print(a)
21      print(inverse_a)
```

```
C:\rep_2.6\venv\Scripts\python.exe C:/rep_2.6/zadaniya.py
{'1a': 59, '11a': 15, '11b': 10}
Количество учеников: 84

{1: 'one', 2: 'two', 3: 'three'}
{'one': 1, 'two': 2, 'three': 3}

Process finished with exit code 0
```

Рисунок 3.1 Вывод программы задания

4. (18 вариант). Выполнил индивидуальное задание.

```

import sys

if __name__ == '__main__':
    shops = []
    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break
        elif command == 'add':
            name = input("Название магазина ")
            product = input("Товар ")
            price = int(input("Цена "))
            shop = {
                'name': name,
                'product': product,
                'price': price,
            }
            shops.append(shop)
            if len(shops) > 1:
                shops.sort(key=lambda item: item.get('name', ''))
        elif command == 'list':
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 8,
                '-' * 20
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                    "No",
                    "Название.",
                    "Товар",
                    "Цена"
                )
            )
            print(line)
            for idx, shop in enumerate(shops, 1):
                print(
                    '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(

```

```

        idx,
        shop.get('name', ''),
        shop.get('product', ''),
        shop.get('price', 0)
    )
    )
    print(line)
elif command == 'select':
    sname = input("Название магазина ")
    cout = 0
    for shop in shops:
        if (shop.get('name') == sname):
            cout = 1
            print(
                ' | {:<5} | {:<5} '.format(
                    shop.get('product', ''),
                    shop.get('price', 0),
                )
            )
        elif cout == 0:
            print("Такого магазина нет")
elif command == 'help':
    print("Список команд:\n")
    print("add - добавить магазин;")
    print("select - показать товары из заданного магазина;")
    print("list - вывести список магазинов;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")
else:
    print(f"Неизвестная команда {command}", file=sys.stderr)

```

```

>>> add
Название магазина пятёрочка
Товар ass
Цена 1000000
>>> магнит
>>> Неизвестная команда магнит
add
Название магазина магнит
Товар лук
Цена 100
>>> add
Название магазина лента
Товар персики

```

```

Товар персики
Цена 100
>>> select
Название магазина лента
| персики | 100
>>> add
Название магазина лента
Товар бананы
Цена 150
>>> select
Название магазина лента
| персики | 100
| бананы | 150
>>>

```



Рисунок 4.1 Вывод программы индивидуального задания

5. Сделал коммит, выполнил слияние с веткой main, и запустил изменения в уд. репозиторий.

```
C:\rep_2.6>git add .  
  
C:\rep_2.6>git commit -m "added programs + modidied .gitignore"  
[develop 2582c62] added programs + modidied .gitignore  
4 files changed, 379 insertions(+), 3 deletions(-)  
create mode 100644 ind.py  
create mode 100644 prim.py  
create mode 100644 zadaniya.py  
  
C:\rep_2.6>git checkout main  
Switched to branch 'main'  
Your branch is up to date with 'origin/main'.  
  
C:\rep_2.6>
```

```
C:\rep_2.6>git push  
Everything up-to-date
```

Рисунок 4.1 коммит и пуш изменений и переход на ветку main

```

C:\rep_2.6>git merge develop
Updating 5d4b8d1..2582c62
Fast-forward
 .gitignore | 157 ++++++
--
 ind.py      | 105 ++++++
 prim.py     | 99  ++++++
 zadaniya.py | 21  ++++++
4 files changed, 379 insertions(+), 3 deletions(-)
create mode 100644 ind.py
create mode 100644 prim.py
create mode 100644 zadaniya.py
C:\rep_2.6>_

```

Рисунок 4.2 Слияние ветки main с develop

```

C:\rep_2.6>git push
info: please complete authentication in your browser...
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 5.16 KiB | 2.58 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/AdamKh/rep_2.6.git
 5d4b8d1..2582c62  main -> main

```

Рисунок 4.3 Пуш изменений на удаленный сервер

#### Контр. вопросы и ответы на них:

##### 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



