

**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

Назаров Никита Юрьевич

Выполнение работы.

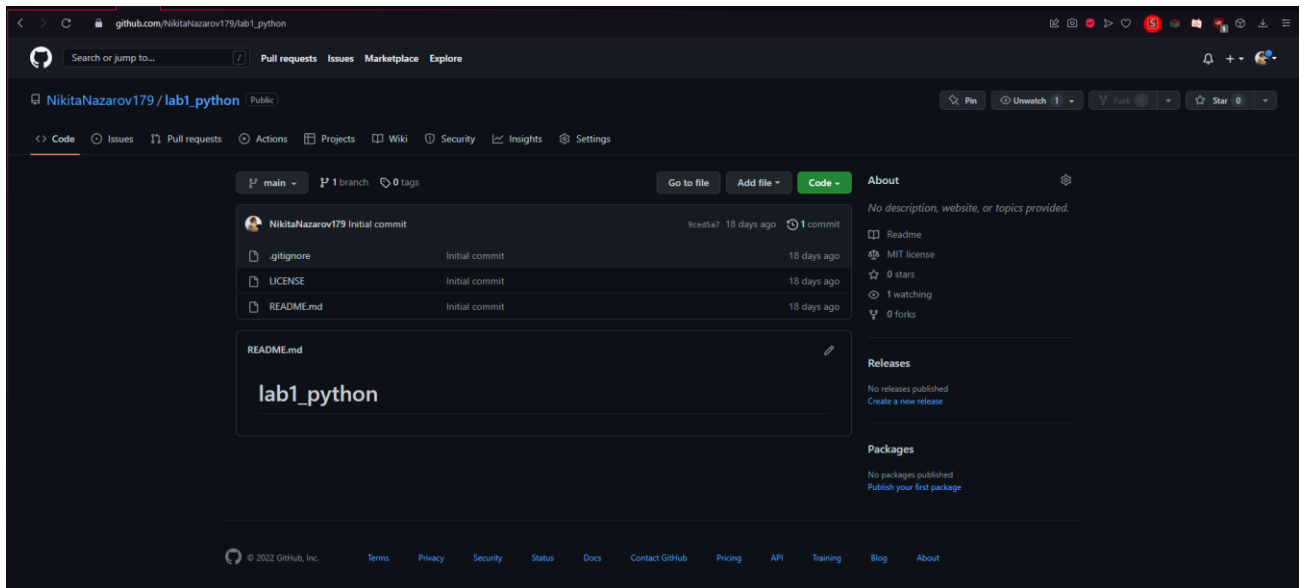


Рисунок 1 – создал новый репозиторий в github

```
H:\cross\git\nazarov>git clone https://github.com/NikitaNazarov179/lab1_python.git
Cloning into 'lab1_python'...
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.
H:\cross\git\nazarov>
```

Рисунок 2 – клонировал репозиторий

```
H:\cross\git\nazarov\lab1_python>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? [H:/cross/git/nazarov/lab1_python/.git/hooks]
```

Рисунок 3 – организовал репозиторий в соответствии с моделью ветвления git-flow



```
*.gitignore - Блокнот
Файл  Правка  Формат  Вид  Справка
# .idea/*.iml
# .idea/modules
# *.iml
# *.ipr

# CMake
cmake-build-*/

# Mongo Explorer plugin
.idea/**/mongoSettings.xml

# File-based project format
*.iws

# IntelliJ
out/

# mpeltonen/sbt-idea plugin
.idea_modules/

# JIRA plugin
atlassian-ide-plugin.xml

# Cursive Clojure plugin
.idea/replstate.xml

# SonarLint plugin
.idea/sonarlint/
```

Рисунок 4 – редактировал файл .gitignore для roucharm

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
from datetime import date

if __name__ == '__main__':
    # Список работников.
    workers = []
    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()
        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break
        elif command == 'add':
            # Запросить данные о работнике.
            name = input("Фамилия и инициалы? ")
            post = input("Должность? ")
            year = int(input("Год поступления? "))
            # Создать словарь.
            worker = {
                'name': name,
                'post': post,
                'year': year,
            }
            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))
        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 8,
            )
            print(line)

```

Рисунок 5 – листинг 1 примера

```

Фамилия и инициалы? Назаров Никита
Должность? Стажер
Год поступления? 2021
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Назаров Никита          |      Стажер          |   2021  |
+-----+-----+-----+-----+
>>> |

```

Рисунок 6 – результат работы 1 примера

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

if __name__ == '__main__':
    school = {"1a": 23, "1b": 22, "2a": 21, "2b": 20, "3a": 19, "4a": 18, "4b": 24, "106": 22,} # Словарь классов с количеством учащихся
    school["2a"] = 19 # Во 2а классе изменилось количество учащихся
    school["1b"] = 15 # В школе появился 1в класс
    del school["106"] # Класс 106 был удален
    print(f"Всего учеников в школе: {sum(school.values())}") # Выводим на экран общее число учащихся в школе
```

Рисунок 7 – листинг программы для выполнения 1 задания

```
H:\Python\python.exe H:/cross/git/nazarov/lab1_python/Задания/1.py
Всего учеников в школе: 160
```

Рисунок 8 – результат работы программы для решения 1 задания

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

if __name__ == '__main__':

    numb = {1: 'n', 2: 'c', 3: 'f', 4: 'u'}
    print({i:n for n, i in numb.items()})
```

Рисунок 9 – листинг программы для выполнения 2 задания

```
H:\Python\python.exe H:/cross/git/nazarov/lab1_python/Задания/2.py
{'n': 1, 'c': 2, 'f': 3, 'u': 4}

Process finished with exit code 0
```

Рисунок 10 – результат работы программы

```

H:\Python\python.exe F:/lab1_python/Задания/ind.py
help - список всех команд
>>> add
Название продукта: сосиски
Магазин: магнит
Стоимость товара: 220
>>> add
Название продукта: колбаса
Магазин: пятерочка
Стоимость товара: 300
>>> add
Название продукта: сыр
Магазин: перекресток
Стоимость товара: 120
>>> list
+-----+-----+-----+-----+
| № | Товар | Магазин | Стоимость товара |
+-----+-----+-----+-----+
| 1 | колбаса | пятерочка | 300 |
| 2 | сосиски | магнит | 220 |
| 3 | сыр | перекресток | 120 |
+-----+-----+-----+-----+
>>> select сыр
1: перекресток 120

```

Рисунок 11 – результат работы программы для решения индивидуального задания

```

help - список всех команд
>>> add
Название продукта: asd
Магазин: asdasdsada
Стоимость товара: 456
>>> add
Название продукта: sad
Магазин: adfgd
Стоимость товара: 345
>>> add
Название продукта: aaa
Магазин: sdfhgdsfh
Стоимость товара: 345
>>> list
+-----+-----+-----+-----+
| № | Товар | Магазин | Стоимость товара |
+-----+-----+-----+-----+
| 1 | aaa | sdfhgdsfh | 345 |
| 2 | asd | asdasdsada | 456 |
| 3 | sad | adfgd | 345 |
+-----+-----+-----+-----+
>>> |

```

Рисунок 12 – сортировка по алфавиту

```

C:\ Командная строка
H:\cross\git\nazarov\lab1_python>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

H:\cross\git\nazarov\lab1_python>git merge develop
Updating 9ced5a7..0151e0c
Fast-forward
 .gitignore | 150 ++++++++-----
 .idea/.gitignore | 3 +
 .idea/.name | 1 +
 .idea/inspectionProfiles/profiles_settings.xml | 6 +
 .idea/lab1_python.iml | 8 ++
 .idea/misc.xml | 4 +
 .idea/modules.xml | 8 ++
 .idea/vcs.xml | 6 +
 .../1.py" | 9 ++
 .../2.py" | 7 +
 .../ind.py" | 103 ++++++++
 .../1.py" | 96 ++++++++
12 files changed, 354 insertions(+), 47 deletions(-)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/.name
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/lab1_python.iml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 "...\\320\\227\\320\\260\\320\\264\\320\\260\\320\\275\\320\\270\\321\\217\\1.py"
create mode 100644 "...\\320\\227\\320\\260\\320\\264\\320\\260\\320\\275\\320\\270\\321\\217\\2.py"
create mode 100644 "...\\320\\227\\320\\260\\320\\264\\320\\260\\320\\275\\320\\270\\321\\217\\ind.py"

```

Рисунок 13 – переход на ветку main и слияние с веткой develop

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
```