

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение
высшего образования**

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра информационных систем и технологий

Отчет по лабораторной работе №9.

Дисциплина: «Основы программной инженерии»

Выполнил:

Студент группы ПИЖ-б-о-22-1,

направление подготовки: 09.03.04

«Программная инженерия»

ФИО: Джараян Арег Александрович

Проверил:

Воронкин Р. А.

Ставрополь 2022

Тема: Лабораторная работа 2.6 Работа со словарями в языке Python.

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

Выполнение работы:

1. Изучил теоретический материал работы.
2. Создал репозиторий на git.hub.

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 * aregdz / **Repository name *** lab9

✔ lab9 is available.

Great repository names are short and memorable. Need inspiration? How about **bookish-palm-tree** ?

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.](#)

Choose a license

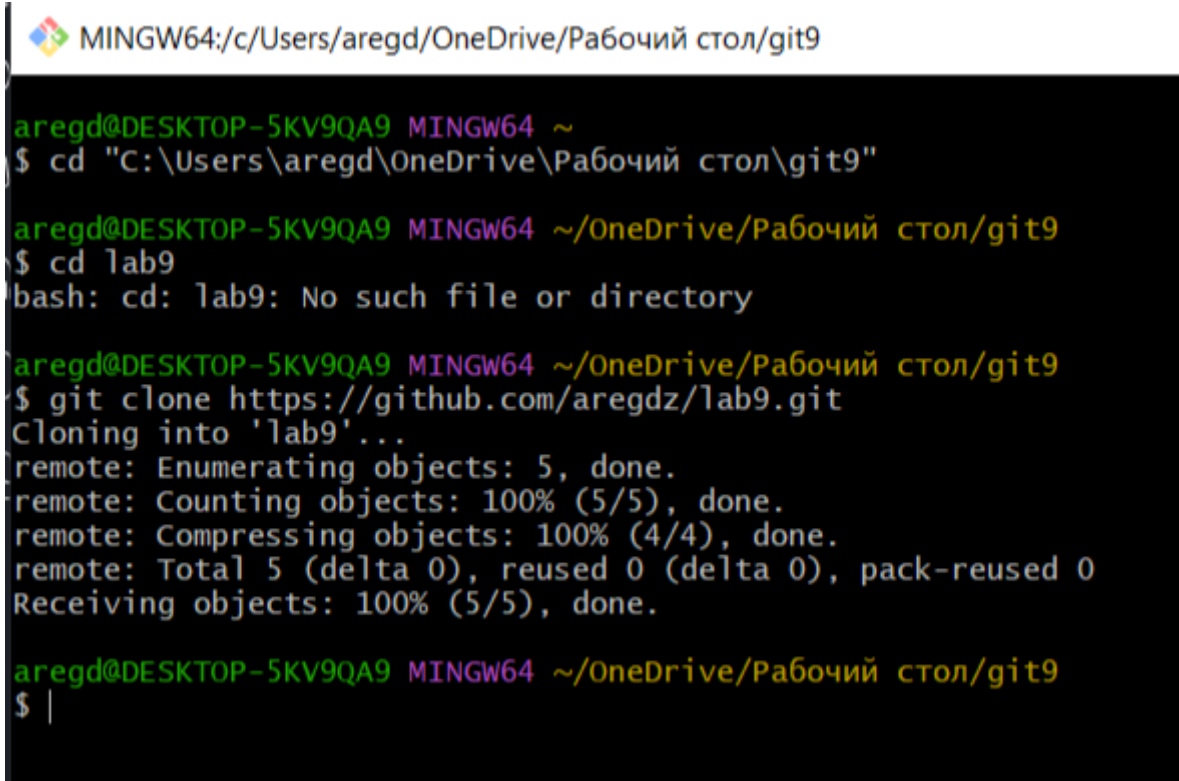
License: **MIT License**

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

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

Рисунок 1 – создание репозитория

3. Клонировал репозиторий.



```
MINGW64:/c/Users/aregd/OneDrive/Рабочий стол/git9

aregd@DESKTOP-5KV9QA9 MINGW64 ~
$ cd "C:\Users\aregd\OneDrive\Рабочий стол\git9"

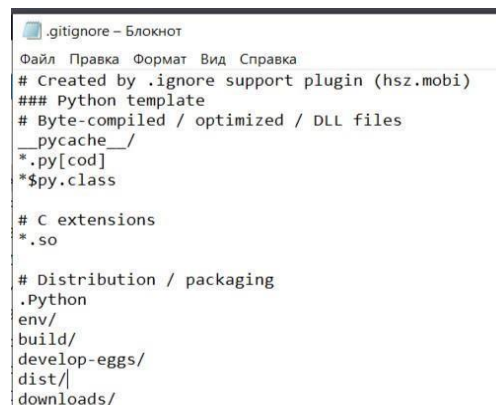
aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9
$ cd lab9
bash: cd: lab9: No such file or directory

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9
$ git clone https://github.com/aregdz/lab9.git
Cloning into 'lab9'...
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.

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9
$ |
```

Рисунок 2 – клонирование репозитория 4.

Дополнить файл gitignore необходимыми правилами.



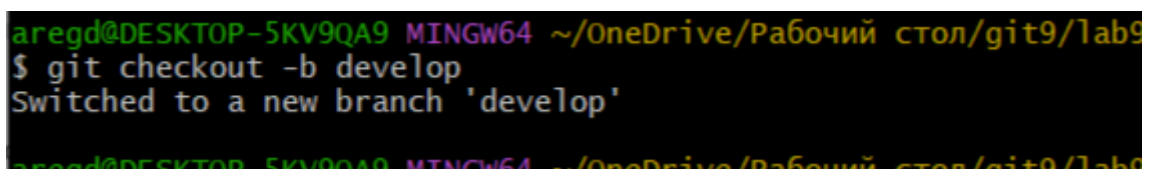
```
.gitignore – Блокнот
Файл Правка Формат Вид Справка
# Created by .ignore support plugin (hsz.mobi)
### Python template
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
env/
build/
develop-eggs/
dist/
downloads/
```

Рисунок 3 – .gitignore для IDE PyCharm

5. Организовать свой репозиторий в соответствии с моделью ветвления git-flow.



```
aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9/lab9
$ git checkout -b develop
Switched to a new branch 'develop'

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9/lab9
```

Рисунок 4 – создание ветки develop

6. Проработал примеры из методички.

```

primer1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import sys
4  from datetime import date
5  if __name__ == '__main__':
6      # Список работников.
7      workers = []
8      # Организовать бесконечный цикл запроса команд.
9      while True:
10         # Запросить команду из терминала.
11         command = input(">>> ").lower()
12         # Выполнить действие в соответствие с командой.
13         if command == 'exit':
14             break
15         elif command == 'add':
16             # Запросить данные о работнике.
17             name = input("Фамилия и инициалы? ")
18             post = input("Должность? ")
19             year = int(input("Год поступления? "))
20             # Создать словарь.
21             worker = {
22                 'name': name,
23                 'post': post,
24                 'year': year,
25             }
26             # Добавить словарь в список.
27             workers.append(worker)
28             # Отсортировать список в случае необходимости.
29             if len(workers) > 1:
30                 workers.sort(key=lambda item: item.get('name', ''))
31
32 elif command == 'list':
33     # Заголовок таблицы.
34     line = '+-{}-+-{}-+-{}-+-{}-+'.format(
35         *args: '-' * 4,
36         '-' * 30,
37         '-' * 20,
38         '-' * 8
39     )
40     print(line)
41     # Вывести данные о всех сотрудниках.
42     for idx, worker in enumerate(workers, 1):
43         print(
44             '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
45                 *args: idx,
46                 worker.get('name', ''),
47                 worker.get('post', ''),
48                 worker.get('year', 0)
49             )
50         )
51     print(line)
52 elif command.startswith('select '):
53     # Получить текущую дату.
54     today = date.today()
55     # Разбить команду на части для выделения номера года.
56     parts = command.split( sep: ' ', maxsplit=1)
57     # Получить требуемый стаж.
58     period = int(parts[1])
59     # Инициализировать счетчик.
60     count = 0
61     # Проверить сведения работников из списка.

```

```

        for worker in workers:
            if today.year - worker.get('year', today.year) >= period:
                count += 1
        print(
            '{:>4}: {}'.format(*args: count, worker.get('name', ''))
        )
        # Если счетчик равен 0, то работники не найдены.
        if count == 0:
            print("Работники с заданным стажем не найдены.")
    elif command == 'help':
        # Вывести справку о работе с программой.
        print("Список команд:\n")
        print("add - добавить работника;")
        print("list - вывести список работников;")
        print("select <стаж> - запросить работников со стажем;")
        print("help - отобразить справку;")
        print("exit - завершить работу с программой.")
    else:
        print(f"Неизвестная команда {command}", file=sys.stderr)

```

Рисунок 5 – пример 1

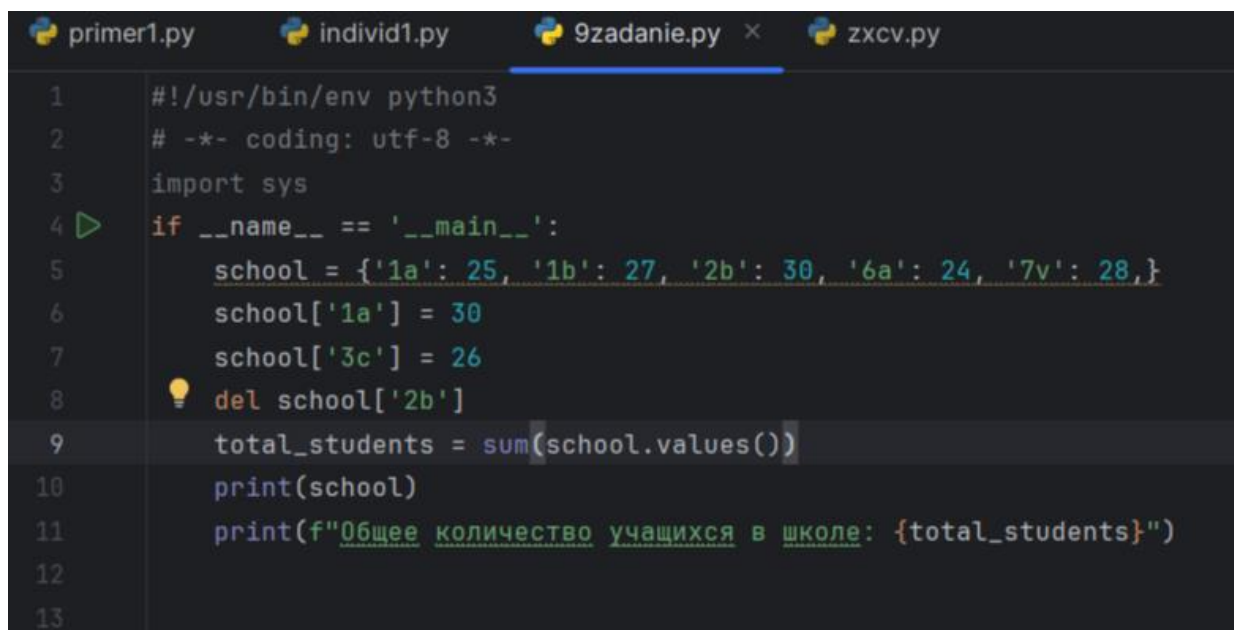
```

>>> add
Фамилия и инициалы? Джараян А.А.
Должность? Самый главный
Год поступления? 2006
>>> dшыe
>>> Неизвестная команда dшыe
list
+-----+-----+-----+-----+
| 1 | Джараян А.А. | Самый главный | 2006 |
+-----+-----+-----+-----+
>>> add
Фамилия и инициалы? Петров
Должность? рабочий
Год поступления? 1999
>>> add
>>> Неизвестная команда add
add
Фамилия и инициалы? Артёмов
Должность? специалист по возвратной логистике
Год поступления? 2002
>>> list
+-----+-----+-----+-----+
| 1 | Артёмов | специалист по возвратной логистике | 2002 |
| 2 | Джараян А.А. | Самый главный | 2006 |
| 3 | Петров | рабочий | 1999 |
+-----+-----+-----+-----+
>>> select 10
3: Петров
>>>

```

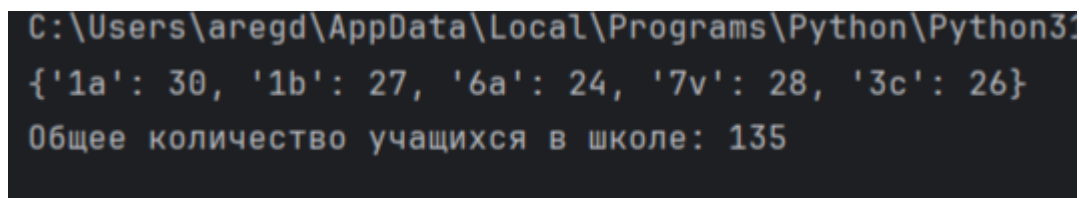
Рисунок 6 – пример выполнения примера 1

7. 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.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 import sys
4 if __name__ == '__main__':
5     school = {'1a': 25, '1b': 27, '2b': 30, '6a': 24, '7v': 28,}
6     school['1a'] = 30
7     school['3c'] = 26
8     del school['2b']
9     total_students = sum(school.values())
10    print(school)
11    print(f"Общее количество учащихся в школе: {total_students}")
12
13
```

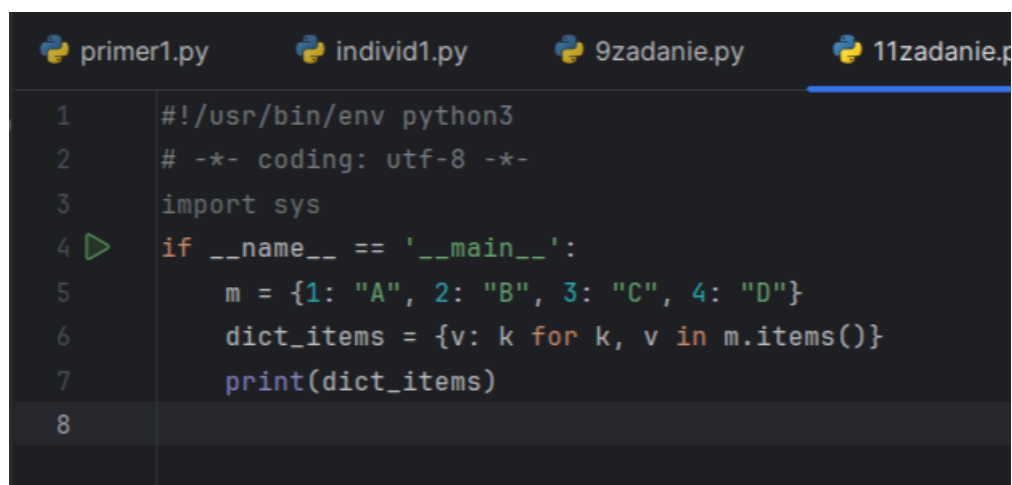
Рисунок 7 – задание 9



```
C:\Users\aregd\AppData\Local\Programs\Python\Python31
{'1a': 30, '1b': 27, '6a': 24, '7v': 28, '3c': 26}
Общее количество учащихся в школе: 135
```

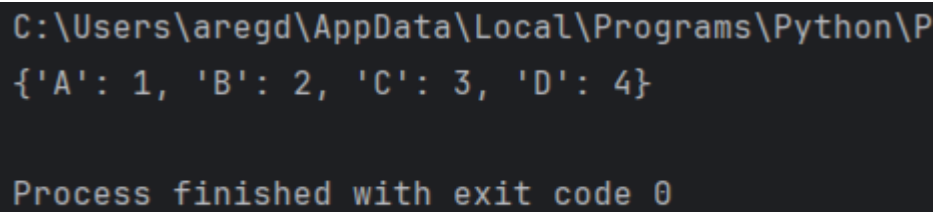
Рисунок 8 – пример выполнения 9 задания

8. 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.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 import sys
4 if __name__ == '__main__':
5     m = {1: "A", 2: "B", 3: "C", 4: "D"}
6     dict_items = {v: k for k, v in m.items()}
7     print(dict_items)
8
```

Рисунок 9 – выполнение задания 11



```
C:\Users\aregd\AppData\Local\Programs\Python\Python39\python.exe  
{'A': 1, 'B': 2, 'C': 3, 'D': 4}  
  
Process finished with exit code 0
```

Рисунок 10 – результат выполнения задания 11

9. 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.

```
primer1.py individ1.py xzcv.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import sys
4  from datetime import date
5  if __name__ == '__main__':
6      print("Список команд:\n")
7      print("add - добавить рейс:")
8      print("list - вывести список рейсов:")
9      print("select <тип> - вывод на экран пунктов назначения и номеров рейсов для данного типа самолёта")
10     print("help - отобразить справку:")
11     print("exit - завершить работу с программой.")
12     # Список работников.
13     aircrafts = []
14     # Организовать бесконечный цикл запроса команд.
15     while True:
16         # Запросить команду из терминала.
17         command = input(">>> ").lower()
18         # Выполнить действие в соответствие с командой.
19         if command == 'exit':
20             break
21         elif command == 'add':
22             # Запросить данные о работнике.
23             name = input("Название пункта назначения рейса? ")
24             number = int(input("Номер рейса? "))
25             tip = input("Тип самолёта? ")
26             # Создать словарь.
27             i = {
28                 'name': name,
29                 'number': number,
30                 'tip': tip,
31             }
32             # Добавить словарь в список.
33             aircrafts.append(i)
34             # Отсортировать список в случае необходимости.
35             if len(aircrafts) > 1:
36                 aircrafts.sort(key=lambda item: item.get('name', ''))
37         elif command == 'list':
38             # Заголовок таблицы.
39             line = '+-{}-+-{}-+-{}-+-{}-+'.format(
40                 *args: '-' * 4,
41                 '-' * 30,
42                 '-' * 20,
43                 '-' * 8
44             )
45             print(line)
46             # Вывести данные о всех сотрудниках.
47             for idx, i in enumerate(aircrafts, 1):
48                 print(
49                     '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
50                         *args: idx,
51                         i.get('name', ''),
52                         i.get('number', ''),
53                         i.get('tip', '')
54                     )
55                 )
56             print(line)
57         elif command.startswith('select '):
58             # Разбить команду на части для выделения номера года.
59             parts = command.split(' ', maxsplit=1)
60             # Проверить сведения работников из списка.
61             count = 0
62             for i in aircrafts:
63                 for k, v in i.items():
64                     if v == parts[1]:
65                         print("Пункт назначения - ", i["name"])
66                         print("Номер рейса - ", i["number"])
67                         count += 1
68             # Если счетчик равен 0, то работники не найдены.
69             if count == 0:
70                 print("Рейс с заданным типом самолёта не найден.")
71         elif command == 'help':
72             # Вывести справку о работе с программой.
73             print("Список команд:\n")
74             print("add - добавить рейс:")
75             print("list - вывести список рейсов:")
76             print("select <тип> - вывод на экран пунктов назначения и номеров рейсов для данного типа самолёта")
77             print("help - отобразить справку:")
78             print("exit - завершить работу с программой.")
79         else:
80             print(f"Неизвестная команда {command}")
```

Рисунок 11 – выполнение индивидуального задания


```

Список команд:

add - добавить рейс;
list - вывести список рейсов;
select <тип> - вывод на экран пунктов назначения и номеров рейсов для данного типа самолёта
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Название пункта назначения рейса? madrid
Номер рейса? 12
Тип самолета? boeing
>>> add
Название пункта назначения рейса? paris
Номер рейса? 34
Тип самолета? airbus
>>> add
Название пункта назначения рейса? Athens
Номер рейса? 234
Тип самолета? embraer
>>> list
+-----+-----+-----+
| 1 | Athens | 234 | embraer |
| 2 | madrid | 12 | boeing |
| 3 | paris | 34 | airbus |
+-----+-----+-----+
>>> select boeing
Пункт назначения - madrid
Номер рейса - 12
>>> |

```

Рисунок 12 – результат выполнения индивидуального задания

9.Зафиксировал все изменения в github в ветке develop.

```

MINGW64:/c:/Users/aregd/OneDrive/Рабочий стол/git9/lab9
aregd@DESKTOP-5KV9QA9 MINGW64 ~
$ cd "C:\Users\aregd\OneDrive\Рабочий стол\git9\lab9"

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9/lab9 (develop)
$ git add .

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9/lab9 (develop)
$ git commit -m"s"
[develop 0d18f4c] s
4 files changed, 177 insertions(+)
create mode 100644 PyCharm/11zadanie.py
create mode 100644 PyCharm/9zadanie.py
create mode 100644 PyCharm/individl.py
create mode 100644 PyCharm/primer1.py

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9/lab9 (develop)
$ |

```

Рисунок 13 – фиксация изменений в ветку develop

10.Слил ветки.

```

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9/lab9 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9/lab9 (main)
$ git merge develop
Updating 3e13fda..0d18f4c
Fast-forward
 PyCharm/11zadanie.py | 7 +++++
 PyCharm/9zadanie.py  | 12 ++++++++
 PyCharm/individ1.py  | 79 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 PyCharm/primer1.py   | 79 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 4 files changed, 177 insertions(+)
 create mode 100644 PyCharm/11zadanie.py
 create mode 100644 PyCharm/9zadanie.py
 create mode 100644 PyCharm/individ1.py
 create mode 100644 PyCharm/primer1.py

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git9/lab9 (main)
$

```

Рисунок 14 – сливание ветки develop в ветку main

Контрольные вопросы:

1. Что такое словари в языке Python?

Словарь (dict) представляет собой структуру данных (которая ещё называется ассоциативный массив), предназначенную для хранения произвольных объектов с доступом по ключу. Данные в словаре хранятся в формате ключ – значение.

2. Может ли функция len() быть использована при работе со словарями?

Да, функция len() может быть использована для работы со словарями в Python. Она возвращает количество элементов (пар ключ-значение) в словаре.

3. Какие методы обхода словарей Вам известны?

- Цикл for по ключам
- Использование метода items(), который возвращает пары ключ-значение
- Обход только ключей с использованием метода keys()
- Обход только значений с использованием метода values()

4. Какими способами можно получить значения из словаря по ключу?

```

my_dict = {'a': 1, 'b': 2, 'c': 3}
for value in my_dict.values():
    print(value)

```

5. Какими способами можно установить значение в словаре по ключу?

```

my_dict = {}
my_dict['ключ'] = 'значение'

```

6. Что такое словарь включений?

Словарь включений аналогичен списковым включениям, за исключением того, что он создаёт объект словаря вместо списка.

```

{x: x * x for x in (1, 2, 3, 4)}
{1: 1, 2: 4, 3: 9, 4: 16}

```

7. Самостоятельно изучите возможности функции zip() приведите примеры ее использования.

Функция zip() в Python используется для объединения двух или более итерируемых объектов (списков, кортежей, и т. д.) в один объект, создавая пары значений. Это может быть полезно, когда вам нужно объединить данные из нескольких источников. Вот примеры использования функции zip().

```
names = ['Анна', 'Петр', 'Мария']
scores = [85, 92, 78]
student_data = list(zip(names, scores))
print(student_data)
```

Результат:

```
[('Анна', 85), ('Петр', 92), ('Мария', 78)]
```

8. Самостоятельно изучите возможности модуля datetime. Каким функционалом по работе с датой и временем обладает этот модуль?

Модуль datetime в Python предоставляет обширный функционал для работы с датой и временем. Вот некоторые из его основных возможностей

1.Создание объектов даты и времени:

- datetime.date: Представляет дату (год, месяц, день).
- datetime.time: Представляет время (час, минута, секунда, микросекунда).
- datetime.datetime: Представляет комбинацию даты и времени.

2.Получение текущей даты и времени:

- datetime.datetime.now(): Возвращает текущую дату и время.

3.Разбор и форматирование даты и времени:

- datetime.datetime.strptime(): Разбор строки в объект datetime.
- datetime.datetime.strftime(): Преобразование объекта datetime в строку с заданным форматом.

4.Арифметика с датой и временем:

- Можно выполнять операции сложения и вычитания времени и даты, а также вычислять разницу между двумя моментами времени.

5.Работа с таймзонами:

- Модуль datetime поддерживает работу с часовыми поясами и таймзонами.

6.Извлечение информации:

- Можно получать год, месяц, день, часы, минуты, секунды и другую информацию о дате и времени.

7.Выполнение сравнений:

- Можно сравнивать даты и времена на предмет того, какой из них раньше или позже.

8.Работа с интервалами времени:

- Модуль `datetime` поддерживает интервалы времени, которые позволяют выразить продолжительность времени.