

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ**  
**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**«Основы ветвления Git»**

**Отчет по лабораторной работе № 2.6**

**по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Образцова М. Д. .«2» декабря 2022г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

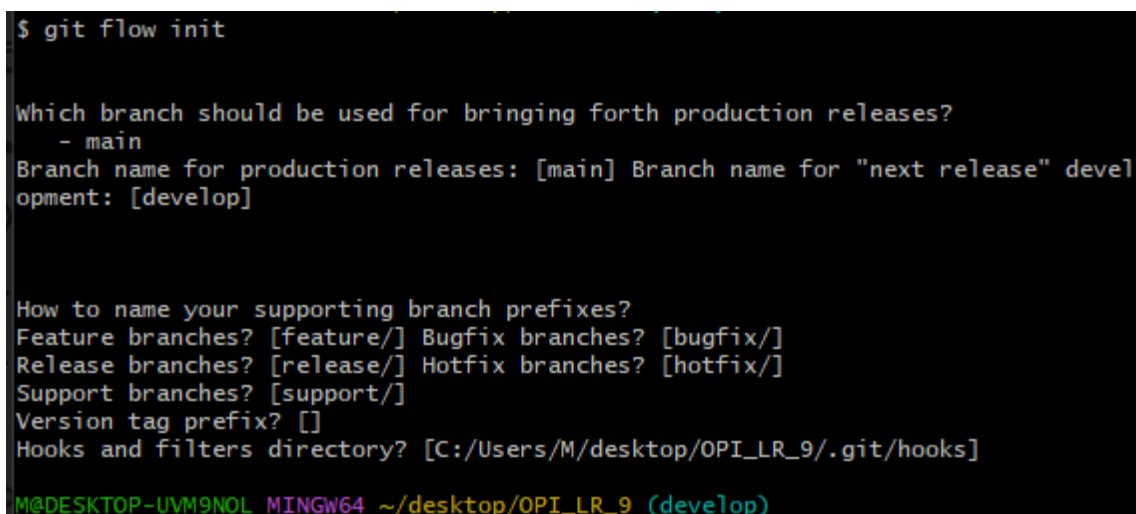
Проверил Воронкин Р.А. \_\_\_\_\_

(подпись)

Ставрополь 2022

## МЕТОДИКА И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создала репозиторий в GitHub «OPI\_LR\_9» в который добавила .gitignore для работы с IDE PyCharm с ЯП Python, выбрала лицензию MIT.



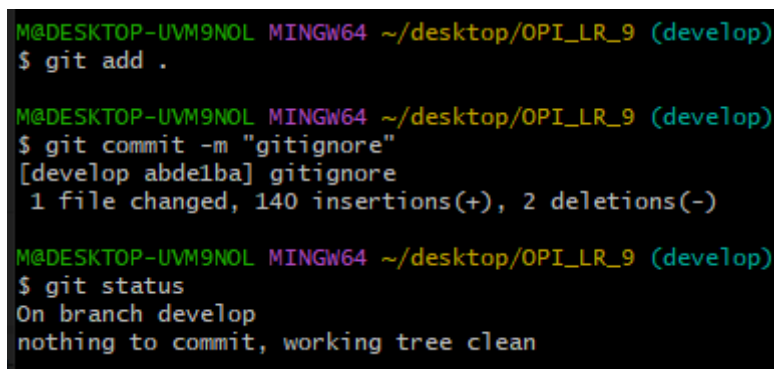
```
$ git flow init

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/M/desktop/OPI_LR_9/.git/hooks]

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_9 (develop)
```

Рисунок 1— Организация репозитория в соответствии с моделью ветвления git-flow



```
M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_9 (develop)
$ git add .

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_9 (develop)
$ git commit -m "gitignore"
[develop abdelba] gitignore
1 file changed, 140 insertions(+), 2 deletions(-)

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_9 (develop)
$ git status
On branch develop
nothing to commit, working tree clean
```

Рисунок 2 – Изменение .gitignore

2. Проработаны примеры лабораторной работы. Для каждого примера создан отдельный модуль языка Python. Зафиксированы изменения в репозитории.

```
1.py x
1 x
C:\Users\M\Desktop\OPI_LR_9\PyCharm\primer\venv\Scripts\python.exe C:\Users\M\
>>> 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? kot a s
Должность? povar
Год поступления? 2001
>>> ad
>>> Неизвестная команда ad
add
Фамилия и инициалы? kring
Должность? musik
Год поступления? 2018
>>> select 10
1: kot a s
>>> list
+-----+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+-----+
|  1 | kot a s                  | povar               | 2001          |
|  2 | kring                    | musik               | 2018          |
+-----+-----+-----+-----+-----+
>>> |
```

Рисунок 3 – Результат выполнения программы для 1 примера

### 3. Выполнила 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.

```
C:\Users\M\Desktop\OPI_LR_9\PyCharm\ind\venv\Scripts\python.exe C:/Users/M/Desktop/OPI_LR_9/PyCharm/ind/venv/Scripts/python.exe C:/Users/M/Desktop/OPI_LR_9/PyCharm/ind/venv/Scripts/python.exe
первоначальный состав:
{'1а': 31, '11б': 15, '3в': 30, '5е': 24, '8г': 29, '6а': 30}

состав после изменений:
{'1а': 31, '11б': 15, '3в': 30, '5е': 17, '6а': 30, '2в': 27}

общее количество учащихся в школе= 150
```

Рисунок 4 – Результат выполнения 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.

```
2 x
C:\Users\M\Desktop\OPI_LR_9\PyCharm\ind\venv\Scripts\python.exe C:/Users/M/Desktop/OPI_LR_9/PyCharm/ind/venv/Scripts/python.exe C:/Users/M/Desktop/OPI_LR_9/PyCharm/ind/venv/Scripts/python.exe
Исходный словарь:
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}

Обратный словарь:
{'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}
```

Рисунок 5 – Результат выполнения программы

### Индивидуальное задание

(2) Использовать словарь, содержащий следующие ключи:

- фамилия и инициалы;
- номер группы;
- успеваемость (список из пяти элементов).

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в список, состоящий из словарей заданной структуры;
- записи должны быть упорядочены по возрастанию среднего балла;
- вывод на дисплей фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5; если таких студентов нет, вывести соответствующее сообщение.

```
3 x
add - добавить студента;
list - вывести список студентов;
find - вывод на фамилий и номеров групп студента с оценками 4 и 5 ;
help - отобразить справку;
exit - завершить работу с программой.
>>> aff
>>> Неизвестная команда aff
add
Ваши фамилия и инициалы: образцова
введите номер своей группы: 4
Ваша успеваемость: 4
>>> add
Ваши фамилия и инициалы: трушева
введите номер своей группы: 3
Ваша успеваемость: 5
>>> add
Ваши фамилия и инициалы: лэйзан
введите номер своей группы: 2
Ваша успеваемость: 3
>>> find
ФИО: образцова
группа: 4
успеваемость: 4

ФИО: трушева
группа: 3
успеваемость: 5

>>> list
+-----+-----+-----+-----+
| No |      ФИО.      | номер группы | успеваемость |
+-----+-----+-----+-----+
|  1 | образцова      | 4            | 4            |
|  2 | трушева        | 3            | 5            |
|  3 | лэйзан         | 2            | 3            |
+-----+-----+-----+-----+
>>> exit
```

Рисунок 6 – Результат выполнения индивидуального задания

## Вопросы для защиты работы

### 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 = [3, 9, 5, 1, 8]
employee_names = ["Дима", "Саша", "Катя", "Анна"]
zipped_values = zip(employee_names, employee_numbers)
zipped_list = list(zipped_values)
print(zipped_list)
```

Функция `zip` возвращает следующее:

```
>>> employee_numbers = [3, 9, 5, 1, 8]
>>> employee_names = ["Дима", "Саша", "Катя", "Анна"]
>>> zipped_values = zip(employee_names, employee_numbers)
>>> zipped_list = list(zipped_values)
>>> print(zipped_list)
[('Дима', 3), ('Саша', 9), ('Катя', 5), ('Анна', 1)]
>>>
```

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)
```

```
>>> import datetime  
>>> dt_now = datetime.datetime.now()  
>>> print(dt_now)  
2022-12-06 23:32:55.102400
```

Получить текущую дату:

```
from datetime import date  
current_date = date.today()  
print(current_date)
```

```
>>> from datetime import date  
>>> current_date = date.today()  
>>> print(current_date)  
2022-12-06  
>>>
```

Получить текущее время:

```
import datetime  
current_date_time = datetime.datetime.now()  
current_time = current_date_time.time()  
print(current_time)
```

```
>>> import datetime  
>>> current_date_time = datetime.datetime.now()  
>>> current_time = current_date_time.time()  
>>> print(current_time)  
23:34:31.333160  
>>>
```