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 №9**

**дисциплины «****Программирование на python»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Кожуховский Виктор Андреевич  2 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем  », очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Руководитель практики:  Воронкин Роман Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** 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.

# Методика и порядок выполнения работы

1. Изучил теоретический материал работы.

2.Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

3. Выполнил клонирование созданного репозитория.

4. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

5. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

6. Создал проект в папке репозитория.

7. Проработал примеры лабораторной работы. Создал для каждого примера отдельный модуль языка Python. Зафиксировал изменения в репозитории.

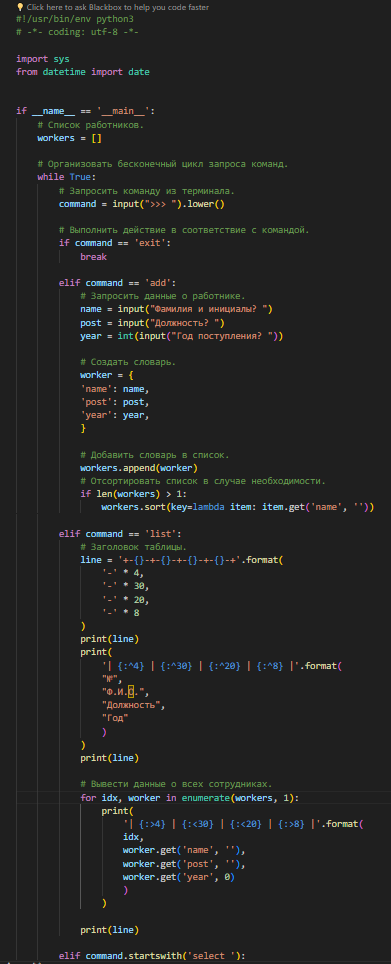


Рисунок 1. Код примера

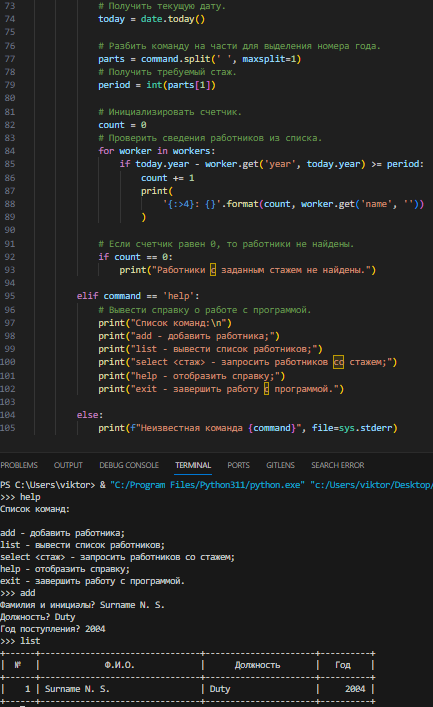


Рисунок 2. Вторая часть кода примера и вывод

9. 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.

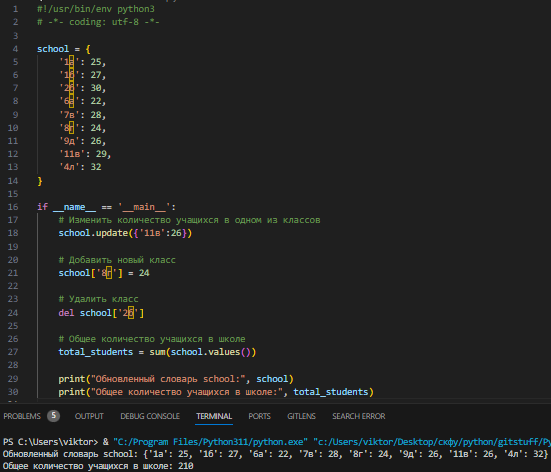


Рисунок 3. Код для решения задачи и его вывод

10. Зафиксировал сделанные изменения в репозитории.

11. Решите задачу: создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод items(), c помощью полученного объекта dict\_items создайте новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.

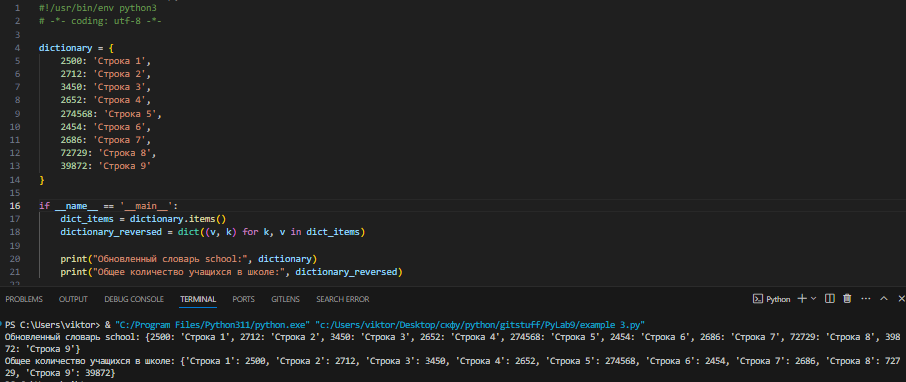


Рисунок 4. Код для решения задачи и его вывод

12. Зафиксировал сделанные изменения в репозитории.

13. Приведите в отчете скриншоты работы программ и UML-диаграммы деятельности решения индивидуального задания.

Использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата рождения (массив из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по знакам Зодиака; вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение.

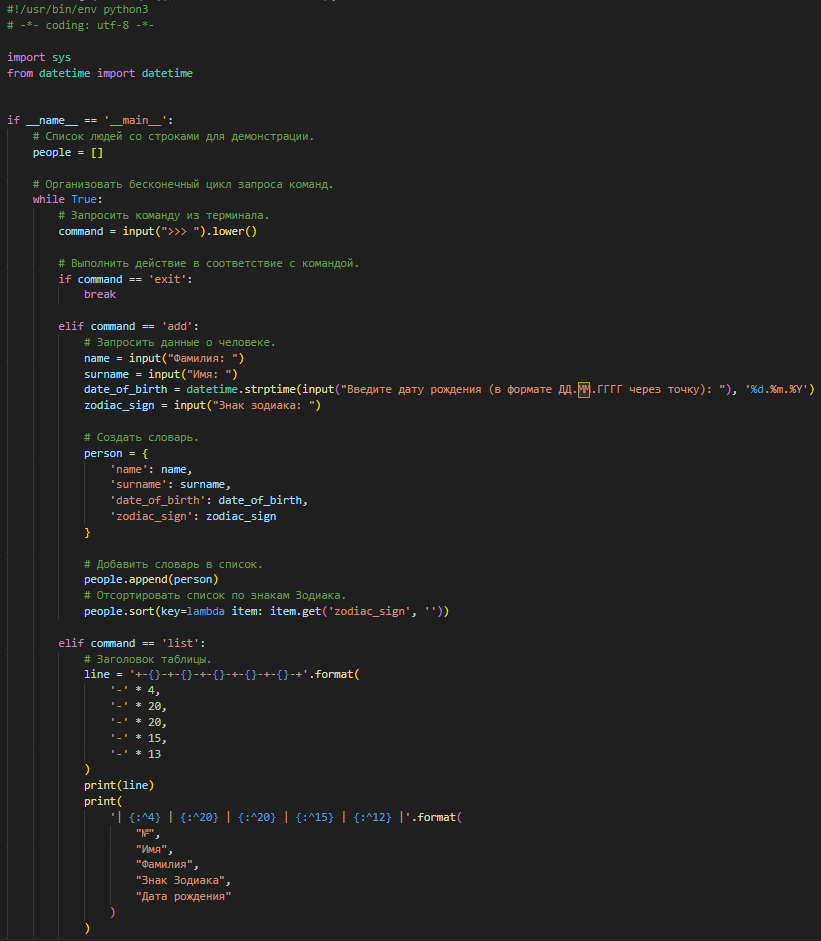


Рисунок 5. Первая часть кода индивидуального задания

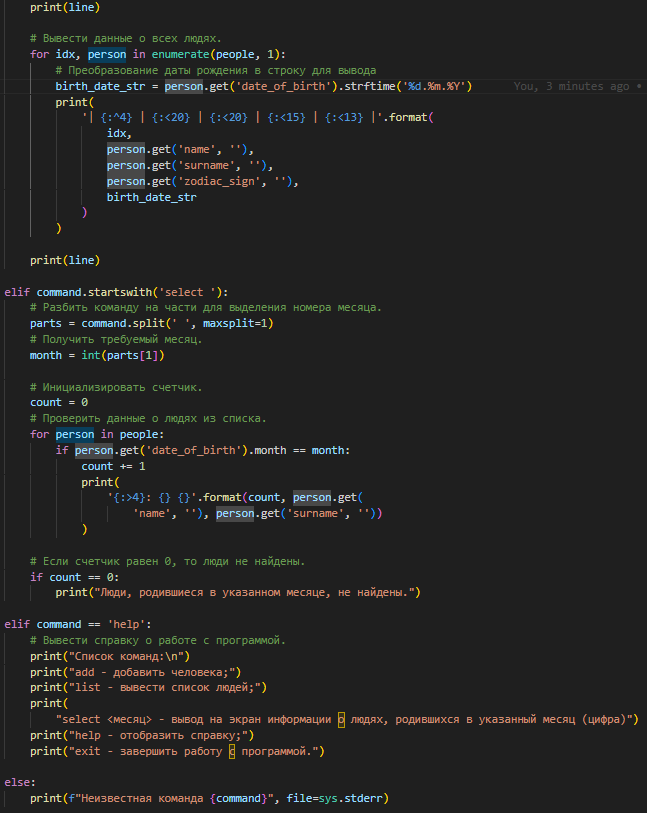


Рисунок 6. Вторая часть кода индивидуального задания

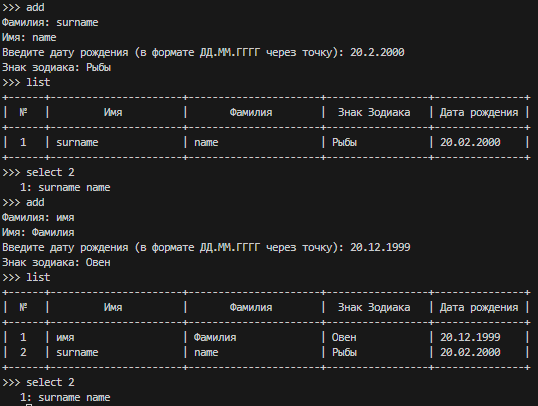


Рисунок 7. Вывод программы индивидуального задания

14. Зафиксировал сделанные изменения в репозитории.

15. Добавил отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксировал изменения.

16. Выполнил слияние ветки для разработки с веткой main / master.

17. Отправил сделанные изменения на сервер GitHub.

Вопросы для защиты работы

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.

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

Может.

3. Какие методы обхода словарей Вам известны?

Элементы словаря перебираются в цикле for также, как элементы других сложных объектов. Однако "по-умолчанию" извлекаются только ключи:

nums

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

for i in nums:

print(i)

1

2

3

Но по ключам всегда можно получить значения:

for i in nums:

print(nums[i])

one

two

three

С другой стороны у словаря как класса есть метод items(), который создает особую структуру, состоящую из кортежей. Каждый кортеж включает ключ и значение:

n = nums.items()

>>> n

dict\_items([(1, 'one'), (2, 'two'), (3, 'three')])

В цикле for можно распаковывать кортежи, таким образом сразу извлекая как ключ, так и его значение:

for key, value in nums.items():

... print(key, 'is', value)

1 is one

2 is two

3 is three

Методы словаря keys() и values() позволяют получить отдельно перечни ключей и значений. Так что если, например, надо перебрать только значения или только ключи, лучше воспользоваться одним из этих методов:

v\_nums = []

>>> for v in nums.values():

... v\_nums.append(v)

>>> v\_nums

['one', 'two', 'three']

То же самое можно сделать с помощью списковых включений:

v\_nums = [v for v in nums.values]

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

Метод get() позволяет получить элемент по его ключу:

>>> nums.get(1)

'one'

Использование квадратных скобок:

num = nums[1]

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

С помощью setdefault() можно добавить элемент в словарь:

>>> nums.setdefault(4, 'four')

'four'

>>> nums

{3: 'three', 4: 'four'}

С помощью update() можно добавить в словарь другой словарь:

>>> nums.update({6: 'six', 7: 'seven'})

>>> nums

{3: 'three', 4: 'four', 6: 'six', 7: 'seven'}

Использование квадратных скобок:

num[1] = ‘Two’

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 используется для объединения элементов из двух или более итерируемых объектов в кортежи.

Примеры:

1. Основное использование:

names = ['Иван', 'Мария', 'Алексей']

ages = [25, 30, 22]

# Объединение итерируемых объектов в кортежи

zipped\_data = zip(names, ages)

# Преобразование результата в список

result\_list = list(zipped\_data)

print(result\_list)

# Вывод: [('Иван', 25), ('Мария', 30), ('Алексей', 22)]

2. Работа с тремя и более итерируемыми объектами:

names = ['Иван', 'Мария', 'Алексей']

ages = [25, 30, 22]

cities = ['Москва', 'Санкт-Петербург', 'Новосибирск']

# Объединение трех итерируемых объектов в кортежи

zipped\_data = zip(names, ages, cities)

# Преобразование результата в список

result\_list = list(zipped\_data)

print(result\_list)

# Вывод: [('Иван', 25, 'Москва'), ('Мария', 30, 'Санкт-Петербург'), ('Алексей', 22, 'Новосибирск')]

3. Работа с различными типами итерируемых объектов:

names = ['Иван', 'Мария', 'Алексей']

ages = (25, 30, 22)

is\_student = [False, True, True]

# Объединение различных типов итерируемых объектов в кортежи

zipped\_data = zip(names, ages, is\_student)

# Преобразование результата в список

result\_list = list(zipped\_data)

print(result\_list)

# Вывод: [('Иван', 25, False), ('Мария', 30, True), ('Алексей', 22, True)]

Функция `zip()` полезна, когда вам нужно работать с соответствующими элементами нескольких итерируемых объектов одновременно.

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

Модуль `datetime` в Python предоставляет классы для работы с датой и временем. Вот основные возможности, которыми обладает этот модуль:

1. Класс `datetime.datetime`:

- `datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0)`: Создает объект `datetime` с указанными значениями года, месяца, дня и т.д.

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

- `datetime.strptime(string, format)`: Преобразует строку в объект `datetime` согласно указанному формату.

- Методы для работы с компонентами даты и времени, такие как `year`, `month`, `day`, `hour`, `minute`, `second`, `microsecond`, и другие.

2. Класс `datetime.date`:

- `datetime.date(year, month, day)`: Создает объект `date` с указанными значениями года, месяца и дня.

- Методы для работы с компонентами даты, такие как `year`, `month`, `day`.

3. Класс `datetime.time`:

- `datetime.time(hour=0, minute=0, second=0, microsecond=0)`: Создает объект `time` с указанными значениями часов, минут, секунд и т.д.

- Методы для работы с компонентами времени, такие как `hour`, `minute`, `second`, `microsecond`.

4. Работа с интервалами и временными разницами:

- `datetime.timedelta`: Позволяет представлять разницу между двумя датами или временем с точностью до микросекунд.

-`timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)`: Создает объект `timedelta` с указанной разницей.

5. Форматирование и парсинг даты и времени:

- `strftime(format)`: Преобразует объект `datetime` в строку согласно указанному формату.

- `strptime(string, format)`: Преобразует строку в объект `datetime` согласно указанному формату.

6. Работа с часовыми поясами:

- `datetime.timezone`: Класс для представления часового пояса.

- `datetime.tzinfo`: Абстрактный класс для представления информации о временной зоне.

7. Модуль `datetime` также включает другие полезные функции, такие как:

- `datetime.today()`: Возвращает текущую дату.

- `datetime.combine(date, time)`: Комбинирует объекты `date` и `time` в объект `datetime`.

- `datetime.min` и `datetime.max`: Представляют минимальную и максимальную даты и времени, соответственно.

Пример использования модуля `datetime`:

from datetime import datetime, timedelta

# Получение текущей даты и времени

now = datetime.now()

print("Текущая дата и время:", now)

# Создание объекта datetime

custom\_date = datetime(2023, 5, 15, 10, 30)

print("Пользовательская дата и время:", custom\_date)

# Вычисление разницы между двумя датами

difference = custom\_date - now

print("Разница между датами:", difference)

# Добавление интервала времени

new\_date = now + timedelta(days=7)

print("Новая дата после добавления интервала:", new\_date)