

**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 №2.16

Дисциплина: «Программирование на Python»

Тема: «Работа с данными формата JSON в языке Python»

Выполнил: студент 2 курса

группы ИВТ-б-о-21-1

Хашиев Адам Мухарбекович

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.

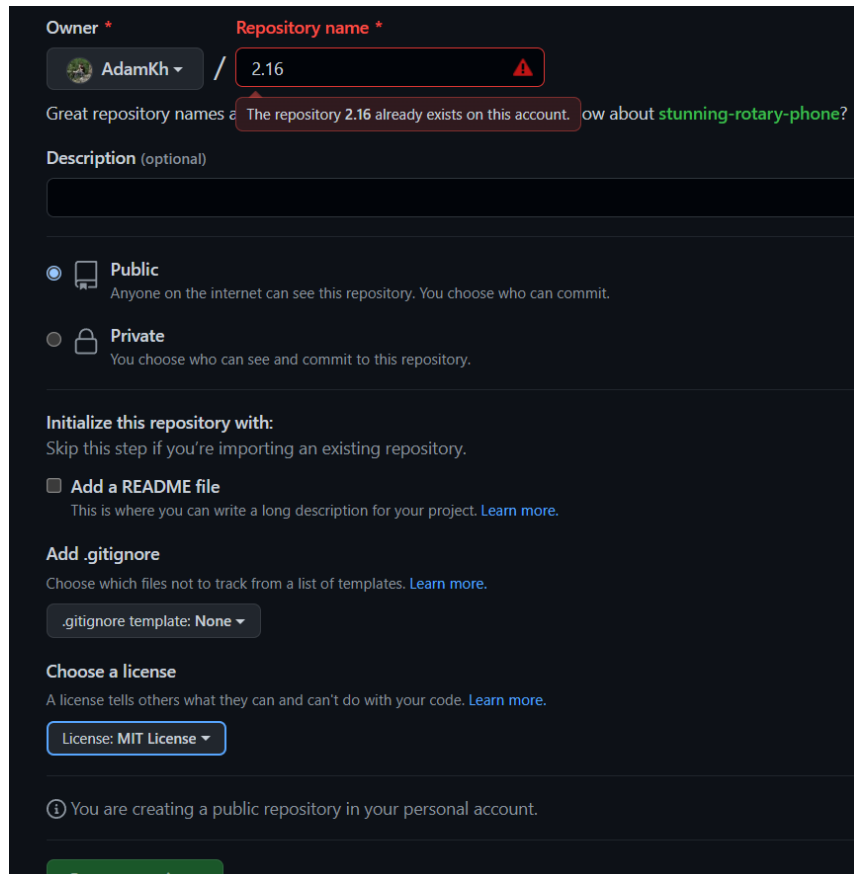


Рисунок 1.1 Создание репозитория

```
C:\Users\adamkh\Desktop\3sem\Python\2.16>git clone https://github.com/AdamKh/2.16.git
Cloning into '2.16'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Рисунок 1.2 Клонирование репозитория

```

C:\Users\adamkh\Desktop\3sem\Python\2.16\2.16>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/adamkh/Desktop/3sem/Python/2.16/2.16/.git]

C:\Users\adamkh\Desktop\3sem\Python\2.16\2.16>
C:\Users\adamkh\Desktop\3sem\Python\2.16\2.16>
C:\Users\adamkh\Desktop\3sem\Python\2.16\2.16>
C:\Users\adamkh\Desktop\3sem\Python\2.16\2.16>
C:\Users\adamkh\Desktop\3sem\Python\2.16\2.16>
C:\Users\adamkh\Desktop\3sem\Python\2.16\2.16>

```

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.

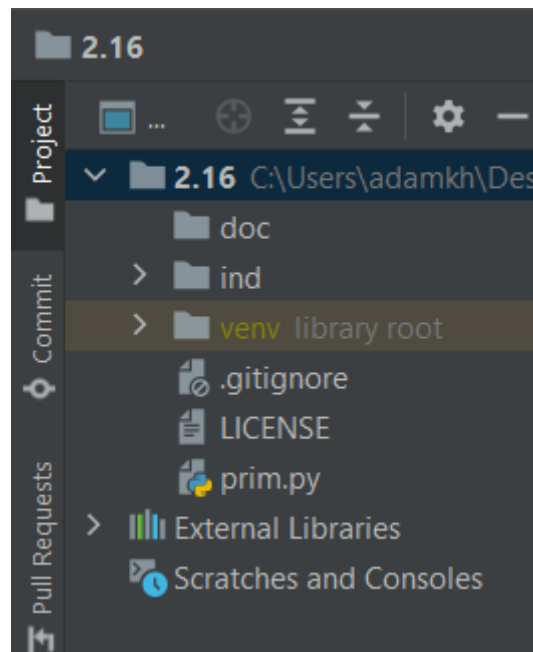


Рисунок 2.1 Создание проекта в PyCharm

```
add
Фамилия и инициалы? asd
Должность? qwe
Год поступления? 2004
>>> list
```

No	Ф.И.О.	Должность	Год
1	asd	qwe	2004

```
>>> save asd.json
>>> |
```

Рисунок 2.2 Рез-т выполнения программы

3. Выполнил индивидуальные задания.

```

>>> add
Фамилия: asd
Имя: qwe
Знак Зодиака: ert
Введите дату выпуска (dd/mm/yyyy)
11/11/1111
>>> list
+-----+-----+-----+-----+
| № |          Фамилия и имя          |      Знак Зодиака      |   Дата рождения   |
+-----+-----+-----+-----+
|  1 | asd          qwe          |      ert      | 11/11/1111      |
+-----+-----+-----+-----+
>>> save asd.json
>>>

```

Рисунок 3.1 Ввод программы задания

```

{} asd.json X
C: > Users > adamkh > Desktop > 3sem > Python >
1  [
2      {
3          "surname": "asd",
4          "name": "qwe",
5          "zodiak": "ert",
6          "date": "11/11/1111"
7      }
8  ]

```

Рисунок 3.2 Вывод программы 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
```

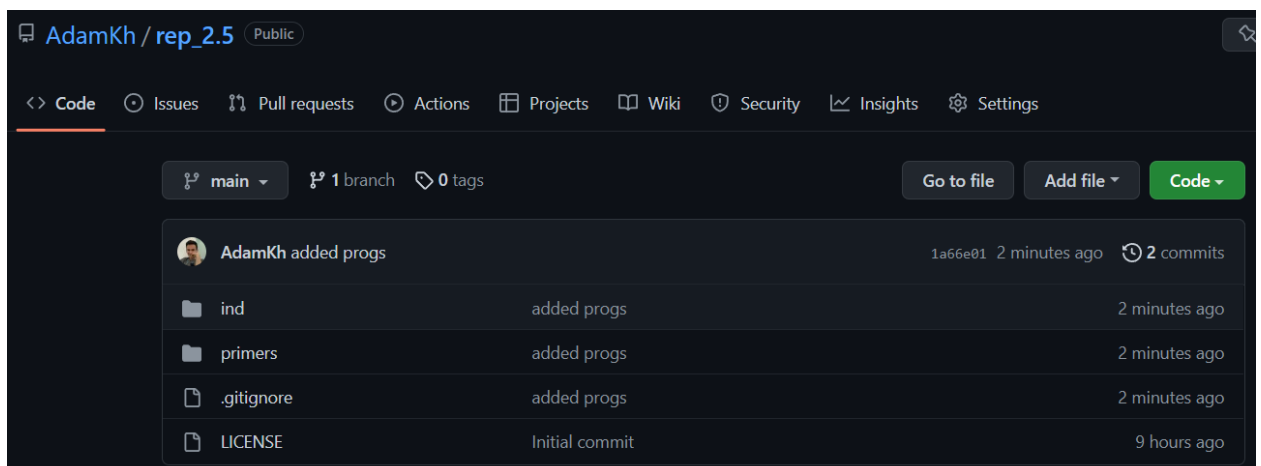
Рисунок 4.1 коммит и пуш изменений и переход на ветку main

```
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



AdamKh / rep_2.5 Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

AdamKh added progs 1a66e01 2 minutes ago 2 commits

ind	added progs	2 minutes ago
primers	added progs	2 minutes ago
.gitignore	added progs	2 minutes ago
LICENSE	Initial commit	9 hours ago

Рисунок 4.4 Изменения на удаленном сервере

Контр. вопросы и ответы на них:

1. Для чего используется JSON?

JSON (англ. JavaScript Object Notation, обычно произносится как JAYsən) – текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

2. Какие типы значений используются в JSON?

Набор пар ключ: значение. Упорядоченный набор значений.

3. Как организована работа со сложными данными в JSON?

JSON может содержать другие вложенные объекты в JSON, в дополнение к вложенным массивам. Такие объекты и массивы будут передаваться, как значения назначенные ключам и будут представлять собой связку ключ-значение.

4. Самостоятельно ознакомьтесь с форматом данных JSON5? В чем отличие этого формата от формата данных JSON?

Формат обмена данными JSON5 (JSON5) - это надмножество JSON, целью которого является смягчение некоторых ограничений JSON путем расширения его синтаксиса для включения некоторых продуктов из ECMAScript 5.1.

Эта библиотека JavaScript является официальной эталонной реализацией библиотек синтаксического анализа и сериализации JSON5.

Краткое описание возможностей. Следующие функции ECMAScript 5.1, которые не поддерживаются в JSON, были расширены до JSON5. Объекты.

Ключи объекта могут быть идентификатором ECMAScript 5.1.

5. Какие средства языка программирования Python могут быть использованы для работы с данными в формате JSON5?

Реализация Python формата данных JSON5.

JSON5 расширяет формат обмена данными JSON, чтобы сделать его более удобным для использования в качестве языка конфигурации:

Комментарии в стиле JavaScript (как однострочные, так и многострочные) разрешены.

Ключи объектов могут быть без кавычек, если они являются допустимыми идентификаторами ECMAScript.

Объекты и массивы могут заканчиваться запятыми.

Строки могут заключаться в одинарные кавычки, допускаются многострочные строковые литералы.

Есть еще несколько более мелких расширений JSON; см. полную информацию на странице выше.

Этот проект реализует реализацию чтения и записи для Python; где возможно, он отражает стандартный пакет Python JSON API для простоты использования.

Есть одно заметное отличие от JSON api: методы `load()` и `loads()` поддерживают опциональную проверку (и отклонение) повторяющихся ключей объекта; `pass allow_duplicate_keys = False` для этого (по умолчанию разрешены дубликаты).

Это ранний выпуск. Это было достаточно хорошо протестировано, но это МЕДЛЕННО. Он может быть в 1000-6000 раз медленнее, чем модуль JSON, оптимизированный для C, и в 200 раз (или более) медленнее, чем модуль JSON на чистом Python.

6. Какие средства предоставляет язык Python для сериализации данных в формате JSON?

`json.dump()` # конвертировать python объект в json и записать в файл

`json.dumps()` # тоже самое, но в строку.

7. В чем отличие функций `json.dump()` и `json.dumps()`?

Dumps записывает в строку, а **dump** в файл.

8. Какие средства предоставляет язык Python для десериализации данных из формата JSON?

`json.load()` # прочитать json из файла и конвертировать в python объект

`json.loads()` # тоже самое, но из строки с json (s на конце от string/строка)

9. Какие средства необходимо использовать для работы с данными формата JSON, содержащими кириллицу?

```
import codecs  
  
json.load(codecs.open('sample.json', 'r', 'utf-8-sig'))
```

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema?

Что такое схема данных?

Схема JSON - это словарь, который позволяет аннотировать и проверять документы JSON.

Преимущества:

- Описывает ваш существующий формат (ы) данных.
- Предоставляет понятную документацию, читаемую человеком и машиной.
- Проверяет данные, которые полезны для:
- Автоматизированное тестирование.
- Обеспечение качества предоставленных клиентом данных.