

Лабораторная работа 2.16

Тема: Работа с данными формата JSON в языке Python.

Цель работы: приобретение навыков по работе с данными формата JSON с помощью языка программирования Python версии 3.x.

Порядок выполнения работы


1. Создаём аккаунт в GitHub. Затем создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

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 *

 RomanGorchakov ▾

Repository name *

/ Test

✓ Test is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-octo-chainsaw](#) ?

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.](#)

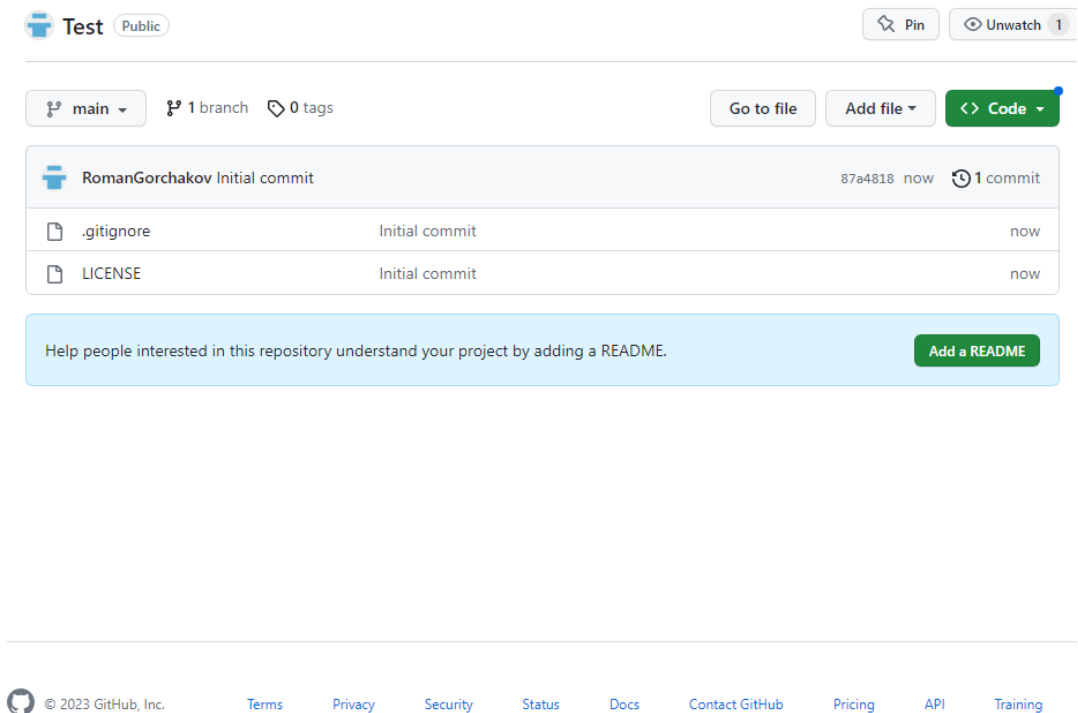
Choose a license

License: MIT License ▾


















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

 You are creating a public repository in your personal account.

Create repository



2. Теперь необходимо дополнить файл `.gitignore` с необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «<https://github.com/github/gitignore>» и скачиваем оттуда файл «Python.gitignore».

 Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
 PlayFramework.gitignore	Added ./project/project to PlayFramework.gitignore	7 years ago
 Plone.gitignore	Covered by global vim template	10 years ago
 Prestashop.gitignore	Update for Prestashop 1.7 (#3261)	3 years ago
 Processing.gitignore	Ignore transpiled .java and .class files (#3016)	4 years ago
 PureScript.gitignore	Update PureScript adding .spago (#3278)	3 years ago
 Python.gitignore	Update Python.gitignore	last year
 Qooxdoo.gitignore	Add gitignore for qooxdoo apps	13 years ago
 Qt.gitignore	Remove trailing whitespace	2 years ago
 R.gitignore	Merge pull request #3792 from jl5000/patch-1	2 years ago
 README.md	Merge pull request #3854 from AnilSeervi/patch-1	2 years ago
 ROS.gitignore	Added ignore for files created by <code>catkin_make_isolated</code>	6 years ago
 Racket.gitignore	Update Racket.gitignore	2 years ago
 Rails.gitignore	Ignore Rails .env according recommendations	2 years ago
 Raku.gitignore	Changes the name of Perl 6 to Raku (#3312)	3 years ago
 RhodesRhomobile.gitignore	Add Rhodes mobile application framework gitignore	13 years ago
 Ruby.gitignore	Ruby: ignore RuboCop remote inherited config files (#3197)	4 years ago

```
1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28
```

3. Теперь создаём файл «README.md», где вносим информацию о своей группе и ФИО. Сохраняем набранный текст через кнопку «Commit changes».



4. В окне «Codespace» выбираем опцию «Create codespace on main». Откроется терминал, куда мы введём команду «git clone», чтобы клонировать свой репозиторий. После этого организуем репозиторий в соответствии с моделью ветвления Git-flow. Для этого введём в терминал команды: «git checkout –b develop» для создания ветки разработки; «git branch feature_branch» для создания ветки функций; «git branch release/1.0.0» для создания ветки релиза; «git checkout main» и «git branch hotfix» для создания веток hotfix.

```
@RomanGorchakov →/workspaces/Py16 (main) $ git clone https://github.com/RomanGorchakov/Py16.git ClonedPy2.g
Cloning into 'ClonedPy2.git'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), 4.80 KiB | 4.80 MiB/s, done.
Resolving deltas: 100% (1/1), done.
@RomanGorchakov →/workspaces/Py16 (main) $ git checkout -b develop
Switched to a new branch 'develop'
@RomanGorchakov →/workspaces/Py16 (develop) $ git branch feature_branch
@RomanGorchakov →/workspaces/Py16 (develop) $ git branch release/1.0.0
@RomanGorchakov →/workspaces/Py16 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
@RomanGorchakov →/workspaces/Py16 (main) $ git branch hotfix
@RomanGorchakov →/workspaces/Py16 (main) $ git checkout develop
Switched to branch 'develop'
@RomanGorchakov →/workspaces/Py16 (develop) $
```

5. Прорабатываем примеры, приведённые в теоретическом материале по лабораторной работе.

```
>>> add
Фамилия и инициалы? Горчаков Р.В
Должность?
Год поступления? 2004
>>> add
Фамилия и инициалы? Горчакова С.В
Должность?
Год поступления? 1977
>>> add
Фамилия и инициалы? Горчаков Д.В
Должность?
Год поступления? 1988
>>> list
```

№	Ф.И.О.	Должность	Год
1	Горчаков Д.В		1988
2	Горчаков Р.В		2004
3	Горчакова С.В		1977

```
>>> save data.json
>>> load data.json
>>>
```

```
[
  {
    "name": "Горчаков Д.В",
    "post": "",
    "year": 1988
  },
  {
    "name": "Горчаков Р.В",
    "post": "",
    "year": 2004
  },
  {
    "name": "Горчакова С.В",
    "post": "",
    "year": 1977
  }
]
```

6. Создаём файл «individual.py», в котором нужно реализовать сохранение и чтение данных из файла формата JSON. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей

заданной структуры; записи должны быть упорядочены по возрастанию номера рейса; вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры; если таких рейсов нет, выдать на дисплей соответствующее сообщение. Необходимо также проследить за тем, чтобы файлы генерируемый этой программой не попадали в репозиторий лабораторной работы.

```
>>> add
Название пункта назначения рейса: moscow
>>> add
Название пункта назначения рейса: saint-petersburg
>>> add
Название пункта назначения рейса: kazan
>>> add
Название пункта назначения рейса: kaliningrad
>>> list
```

No	Пункт назначения	Номер рейса	Тип самолёта
1	kaliningrad	4449	90
2	kazan	3780	1
3	moscow	8850	9
4	novoaleksandrovsk	3234	62
5	saint-petersburg	9864	48
6	stavropol	9120	82

```
>>> save schedule.json
>>> load schedule.json
>>> exit
```

```
[
  {
    "race": "kaliningrad",
    "number": 4449,
    "type": 90
  },
  {
    "race": "kazan",
    "number": 3780,
    "type": 1
  },
  {
    "race": "moscow",
    "number": 8850,
    "type": 9
  },
  {
    "race": "novoaleksandrovsk",
    "number": 3234,
    "type": 62
  },
  {
    "race": "saint-petersburg",
    "number": 9864,
    "type": 48
  },
  {
    "race": "stavropol",
    "number": 9120,
    "type": 82
  }
]
```

7. Создаём файл «valid.json», в котором пользователю необходимо реализовать валидацию загруженных данных с помощью спецификации JSON Schema.

```

{
  "comment": "Schema for the striped object specification file",
  "type": "object",
  "patternProperties": {
    "^[a-z0-9]+(inv)?$": {
      "type": "object",
      "properties": {
        "race": {
          "type": "string",
          "pattern": "^[a-z]$",
          "number": { "$ref": "#/definitions/race_number" },
          "type": { "$ref": "#/definitions/type_number" }
        }
      },
      "required": ["race", "number", "type"]
    }
  },
  "additionalProperties": false,
  "definitions": {
    "race_number": {
      "type": "number",
      "minimum": 1,
      "maximum": 99,
      "exclusiveMinimum": true
    },
    "type_number": {
      "type": "number",
      "minimum": 1000,
      "maximum": 9999,
      "exclusiveMinimum": true
    }
  }
}


```















8. Выполняем коммит файлов в репозиторий Git в ветку разработки, сливаем её с веткой main и отправляем изменения на сервер GitHub.




```

create mode 100644 valid.json
@RomanGorchakov →/workspaces/Py16 (develop) $ git checkout main
warning: unable to rmdir 'ClonedPy2.git': Directory not empty
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
@RomanGorchakov →/workspaces/Py16 (main) $ git merge develop
Updating 7a04f3a..84a5fa7
Fast-forward
 ClonedPy2.git      | 1 +
 data.json          | 17 +++++
 example.py         | 177 +++++
 individual.py       | 10 +
 packet/__init__.py | 1 +
 packet/display_plane.py | 30 +
 packet/get_plane.py | 10 +
 packet/load_plane.py | 3 +
 packet/packet.py    | Bin 0 -> 82 bytes
 packet/save_plane.py | 3 +
 packet/show_plane.py | 118 +
 schedule.json      | 32 +
 valid.json          | 33 +
13 files changed, 435 insertions(+)
create mode 100000 ClonedPy2.git
create mode 100644 data.json
create mode 100644 example.py
create mode 100644 individual.py
create mode 100644 packet/__init__.py
create mode 100644 packet/display_plane.py
create mode 100644 packet/get_plane.py
create mode 100644 packet/load_plane.py
create mode 100644 packet/packet.py
create mode 100644 packet/save_plane.py
create mode 100644 packet/show_plane.py
create mode 100644 schedule.json
create mode 100644 valid.json
@RomanGorchakov →/workspaces/Py16 (main) $ git push -u
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 2 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (15/15), 4.34 KiB | 2.17 MiB/s, done.
Total 15 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/RomanGorchakov/Py16
 7a04f3a..84a5fa7 main -> main
branch 'main' set up to track 'origin/main'.
@RomanGorchakov →/workspaces/Py16 (main) $

```

 Установить рекомендуемое Расширение «Python» от Microsoft для языка Python?
 Управлять Показать рекомендации

 example5.py	Files and Python programmes	6 minutes ago
 example6.py	Files and Python programmes	6 minutes ago
 example7.py	Files and Python programmes	6 minutes ago
 example8.py	Files and Python programmes	6 minutes ago
 example9.py	Files and Python programmes	6 minutes ago
 file2.txt	Files and Python programmes	6 minutes ago
 file3.txt	Files and Python programmes	6 minutes ago
 harrypotter.txt	Files and Python programmes	6 minutes ago
 harrypotternew.txt	Files and Python programmes	6 minutes ago
 individual1.py	Files and Python programmes	6 minutes ago
 individual2.py	Files and Python programmes	6 minutes ago
 newfile.txt	Files and Python programmes	6 minutes ago
 text.txt	Files and Python programmes	6 minutes ago
 ЛР2.15_ГорчаковРВ.pdf	Add files via upload	now

 README
  MIT license
 

Информация обо мне

Имя: Роман.

Фамилия: Горчаков.

Отчество: Владимирович.

Название группы: ПИЖ-6-о-22-1.

Контрольные вопросы

1. Для чего используется JSON?

JSON применяется в веб-приложениях как для обмена данными между браузером и сервером (AJAX), так и между серверами (программные HTTP-сопряжения).

2. Какие типы значений используются в JSON?

В JSON значение может быть одним из шести типов данных: строка; число; логический; null; объект; массив.

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 – это расширенная JSON-версия, которая призвана смягчить некоторые ограничения JSON, расширив его синтаксис и включив в него некоторые функции из ECMAScript 5.1.

Некоторые нововведения:

- поддерживаются как однострочные `//`, так и многострочные `/**/` комментарии;
- записи и списки могут иметь запятую после последнего элемента (удобно при копировании элементов);
- ключи записей могут быть без кавычек, если они являются валидными идентификаторами ECMAScript 5;
- строки могут заключаться как в одинарные, так и в двойные кавычки;
- числа могут быть в шестнадцатеричном виде, начинаться или заканчиваться десятичной точкой, включать Infinity, -Infinity, NaN и -NaN, начинаться со знака +.

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

Комментарии, поддержка многострочного текста, поддержка восьми- и шестнадцатеричных систем счисления

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

Для строки используется `json.dumps()`, а для файла `json.dump()`.

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

`json.dump()` – конвертирует python-объект в json и записывает в файл

`json.dumps()` – конвертирует python-объект в json и записывает в строку

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

Для строки используется `json.loads()`, а для файла `json.load()`.

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

`json_encode($array, JSON_UNESCAPED_UNICODE);`

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema? Что такое схема данных? Приведите схему данных для примера 1.

JSON Schema позволяет добавить необходимые метаданные, или «правила» для этой информации, и описать, какие поля должны быть заполнены и что они могут содержать. JSON Schema используется для валидации данных при обмене информацией между разными системами или при работе с данными, получаемыми от пользователей.

```
{
  "comment": "Schema for the striped object specification file",
  "type": "object",
  "patternProperties": {
    "^[a-z0-9]+(inv)?$": {
      "type": "object",
      "properties": {
        "name": {
          "type": "string",
          "pattern": "^[a-z]$",
          "post": {
            "type": "string",
            "pattern": "^[a-z]$",
            "year": { "$ref": "#/definitions/year_number" }
          }
        },
        "required": ["name", "post", "year"]
      }
    },
    "additionalProperties": false,
    "definitions": {
      "year_number": {
        "type": "number",
        "minimum": 1900,
        "exclusiveMinimum": true
      }
    }
  }
}
```