

Лабораторная работа №11

Выполнил Эсеналиев Арсен

ИБТ-б-о-21-1

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


1. Создал общедоступный репозиторий на GitHub с MIT

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *


Repository name *

 Arsen445 ▾


/ LB2.16 ✓

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

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:
Skip this step if you're importing an existing repository.


☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)


Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

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

License: MIT License ▾

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

 You are creating a public repository in your personal account.

Create repository

2. Выполнил клонирование созданного репозитория.

```
D:\>cd REP11

D:\REP11>git clone https://github.com/Arsen445/LB2.16.git
Cloning into 'LB2.16'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

D:\REP11>_
```

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

```
# 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

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files
.idea/**/dataSources/
```

4. Организовал репозиторий в соответствии с моделью ветвления git-flow.

```
D:\REP11>git flow init
Already initialized for gitflow.
To force reinitialization, use: git flow init -f

D:\REP11>cd LB2.16

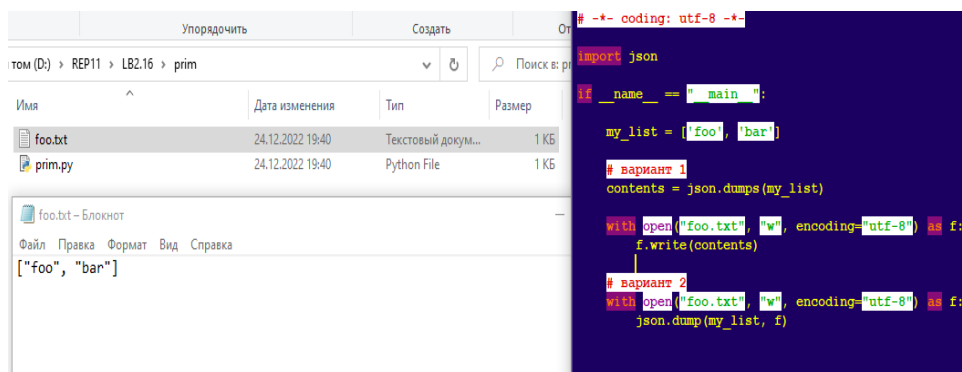
D:\REP11\LB2.16>git flow init

which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

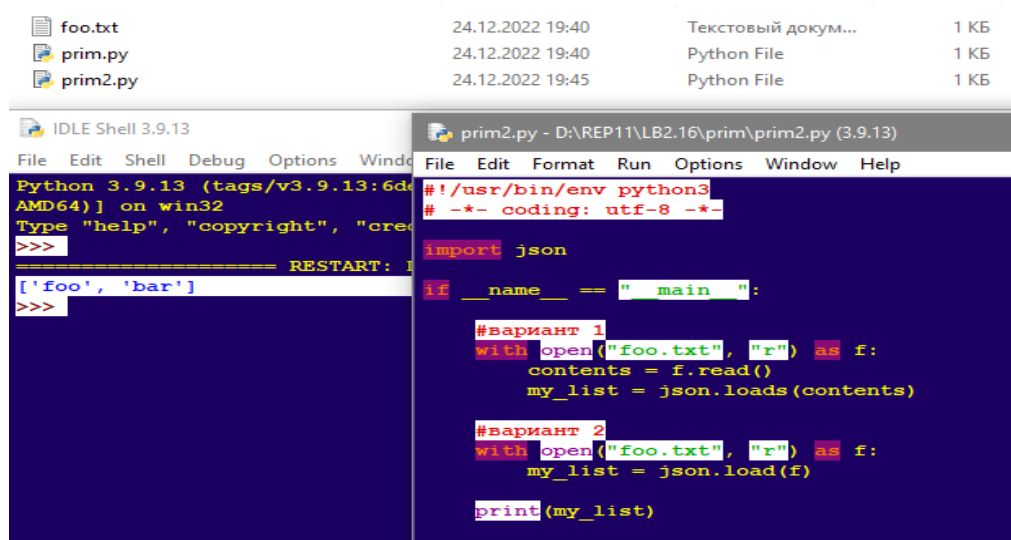
How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/REP11/LB2.16/.git/hooks]
```

5. Проработал примеры лабораторной работы.

Запись в файл



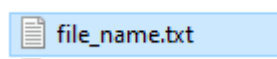
Чтение из файла



Пример1

```
>>> list
Список работников пуст.
>>> add
Фамилия и инициалы? Peh jk
Должность? rpr
Год поступления? 2000
>>> add
Фамилия и инициалы? ej ejd
Должность? edr
Год поступления? 23
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1  | Peh jk                    |      rpr            |      2000     |
|  2  | ej ejd                    |      edr            |       23      |
+-----+-----+-----+-----+
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
load - загрузить данные из файла;
save - сохранить данные в файл;
exit - завершить работу с программой.
>>> save
Неизвестная команда save
>>> save file_name.txt
>>>
```



```
file_name.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
[
  {
    "name": "Peh jk",
    "post": "rpr",
    "year": 2000
  },
  {
    "name": "ej ejd",
    "post": "edr",
    "year": 23
  }
]
```

```
>>> list
Список работников пуст.
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
load - загрузить данные из файла;
save - сохранить данные в файл;
exit - завершить работу с программой.
>>> load file_name.txt
>>> list
```

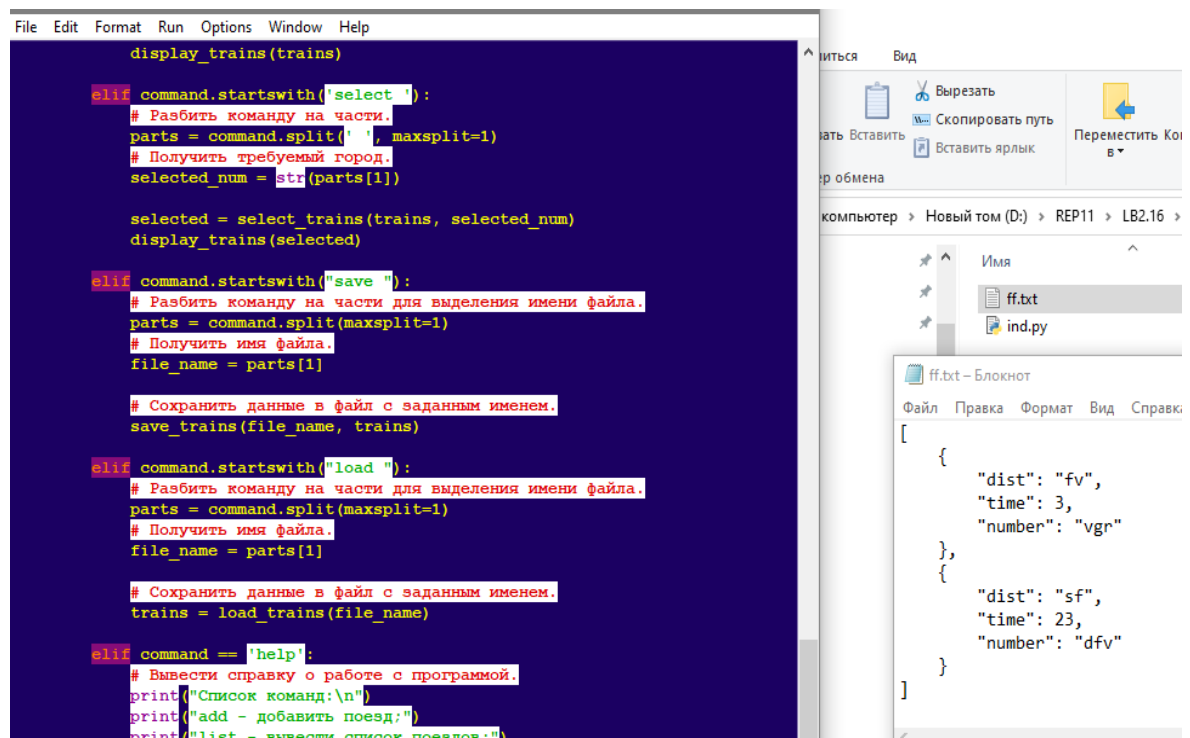
No	Ф.И.О.	Должность	Год
1	Peh jk	rpr	2000
2	ej ejd	edr	23

```
>>>
```

Индивидуальное 1

Для своего варианта лабораторной работы 2.8 необходимо дополнительно реализовать сохранение и чтение данных из файла формата JSON.

Необходимо также проследить за тем, чтобы файлы генерируемый этой программой не попадали в репозиторий лабораторной работы.



Повышенная сложность

Очевидно, что программа в примере 1 и в индивидуальном задании никак не проверяет правильность загружаемых данных формата JSON. В следствие чего, необходимо после загрузки из файла JSON выполнять валидацию загруженных данных. Валидацию данных необходимо производить с использованием спецификации JSON Schema, описанной на сайте <https://json-schema.org/>. Одним из возможных вариантов работы с JSON Schema является использование пакета `jsonschema`, который не является частью стандартной библиотеки Python. Таким образом, необходимо реализовать валидацию загруженных данных с помощью спецификации JSON Schema.

```

def load_trains(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    schema = {
        "type": "array",
        "items": [
            {
                "type": "object",
                "trains": {
                    "dist": {
                        "type": "string"
                    },
                    "time": {
                        "time": "integer"
                    },
                    "number": {
                        "type": "string"
                    }
                },
                "required": ["dist", "time", "number"]
            }
        ]
    }
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        file = json.load(fin)
        validator = jsonschema.Draft7Validator(schema)
        try:
            if not validator.validate(file):
                print("Валидация прошла успешно.")
            except jsonschema.exceptions.ValidationError:
                print("Ошибка. Проверьте файл.", list(validator.iter_errors(file)))
                exit()
        return file

def main():
    """
    Главная функция программы.
    """

```

6. Зафиксируйте сделанные изменения в репозитории.(после создания веток не запустил, поэтому не работало)

```

D:\REP11\LB2.16>git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        doc/
        ind/
        prim/

nothing added to commit but untracked files present (use "git add" to track)
D:\REP11\LB2.16>git add --all

D:\REP11\LB2.16>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore
        new file:   "doc/~$\320\233\320\22111.docx"
        new file:   "doc/\320\233\320\22111.docx"
        new file:   ind/ind.py
        new file:   ind/povish/ewr.json
        new file:   ind/povish/ewr.txt
        new file:   ind/povish/ind.py
        new file:   ind/povish/povish.py
        new file:   ind/povish/text.txt
        new file:   prim/file_name.txt
        new file:   prim/foo.txt
        new file:   prim/prim.py
        new file:   prim/prim2.py
        new file:   prim/prim3.py

D:\REP11\LB2.16>git commit-m"j"
git: 'commit-mj' is not a git command. See 'git --help'.

D:\REP11\LB2.16>git commit -m "j"
[develop d3397c0] j
14 files changed, 919 insertions(+)
create mode 100644 .gitignore

```

7. Выполните слияние ветки для разработки с веткой main/master.


```

D:\REP11\LB2.16>git merge develop
Merge made by the 'ort' strategy.
 .gitignore | 279 ++++++
 "doc/~$\320\233\320\22111.docx" | Bin 0 -> 162 bytes
 ind/ind.py | 166 ++++++
 ind/povish/ewr.json | 5 +
 ind/povish/ewr.txt | 5 +
 ind/povish/ind.py | 193 ++++++
 ind/povish/povish.py | 46 +++++
 ind/povish/text.txt | 5 +
 prim/file_name.txt | 12 ++
 prim/foo.txt | 1 +
 prim/prim.py | 18 +++
 prim/prim2.py | 17 +++
 prim/prim3.py | 172 ++++++
 13 files changed, 919 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 "doc/~$\320\233\320\22111.docx"
 create mode 100644 ind/ind.py
 create mode 100644 ind/povish/ewr.json
 create mode 100644 ind/povish/ewr.txt
 create mode 100644 ind/povish/ind.py
 create mode 100644 ind/povish/povish.py
 create mode 100644 ind/povish/text.txt
 create mode 100644 prim/file_name.txt
 create mode 100644 prim/foo.txt
 create mode 100644 prim/prim.py
 create mode 100644 prim/prim2.py
 create mode 100644 prim/prim3.py

```

Контрольные вопросы:

1. Как открыть файл в языке Python только для чтения?

Чтобы открыть файл для чтения, мы используем режим r. Для чтения мы воспользуемся функцией read(size), если параметр size не указан, функция вернет нам всю строку. file = open("text.txt", 'r', encoding = 'utf-8').

2. Как открыть файл в языке Python только для записи?

В Python открытие файлов выполняется с помощью функции open(), которой передается два аргумента - имя файла и режим. Файл может быть открыт в режиме чтения, записи, добавления.

3. Как прочитать данные из файла в языке Python?

Чтение данных из файла осуществляется с помощью методов read(размер) и readline(). Метод read(размер) считывает из файла определенное количество символов, переданное в качестве аргумента.

4. Как записать данные в файл в языке Python?

Запись данных в файл. Записать данные в файл можно с помощью метода `write()`.

5. Как закрыть файл в языке Python?

После того, как мы открыли файл, и выполнили все нужные операции, нам необходимо его закрыть. Для закрытия файла используется функция `close()`.

6. Изучите самостоятельно работу конструкции `with ... as`. Каково ее назначение в языке?

Конструкция `with ... as` используется для оборачивания выполнения блока инструкций менеджером контекста. Если в конструкции `with - as` было несколько выражений, то это эквивалентно нескольким вложенным конструкциям

7. Изучите самостоятельно документацию Python по работе с файлами. Какие помимо рассмотренных существуют методы записи/чтения информации из файла?

Один из самых распространенных способов вывести данные в Python – это напечатать их в консоли. Если вы находитесь на этапе изучения языка, такой способ является основным для того, чтобы быстро просмотреть результат своей работы

8. Какие существуют, помимо рассмотренных, функции модуля `os` для работы с файловой системой?

`os.chdir(path)` - смена текущей директории.

`os.chmod (path, mode, *, dir_fd=None, follow_symlinks=True)` - смена прав доступа к объекту (`mode` - восьмеричное число).

`os.chown (path, uid, gid, *, dir_fd=None, follow_symlinks=True)` - меняет id владельца и группы (Unix).

`os.getcwd()` - текущая рабочая директория.

`os.link (src, dst, *, src_dir_fd=None, dst_dir_fd=None, follow_symlinks=True)` - создаёт жёсткую ссылку.

`os.listdir (path=".")` - список файлов и директорий в папке.

`os.mkdir (path, mode=0o777, *, dir_fd=None)` - создаёт директорию.

`OSError`, если директория существует.

`os.makedirs (path, mode=0o777, exist_ok=False)` - создаёт директорию, создавая при этом промежуточные директории.

`os.remove (path, *, dir_fd=None)` - удаляет путь к файлу.

`os.rename (src, dst, *, src_dir_fd=None, dst_dir_fd=None)` - переименовывает файл или директорию из `src` в `dst`.

`os.rename (old, new)` - переименовывает `old` в `new`, создавая промежуточные директории.

`os.replace (src, dst, *, src_dir_fd=None, dst_dir_fd=None)` - переименовывает из `src` в `dst` с принудительной заменой.

`os.rmdir (path, *, dir_fd=None)` - удаляет пустую директорию.

`os.removedirs (path)` - удаляет директорию, затем пытается удалить родительские директории, и удаляет их рекурсивно, пока они пусты.

`os.sync()` - записывает все данные на диск (Unix).

`os.truncate (path, length)` - обрезает файл до длины `length`.

`os.utime (path, times=None, *, ns=None, dir_fd=None, follow_symlinks=True)` - модификация времени последнего доступа и изменения файла. Либо `times` - кортеж (время доступа в секундах, время изменения в секундах), либо `ns` - кортеж (время доступа в наносекундах, время изменения в наносекундах).

`os.walk (top, topdown=True, onerror=None, followlinks=False)` – генерация имён файлов в дереве каталогов, сверху вниз (если `topdown` равен `True`),

либо снизу вверх (если False). Для каждого каталога функция walk возвращает кортеж (путь к каталогу, список каталогов, список файлов).