

Лабораторная работа 2.15

Тема: Работа с файлами в языке Python.

Цель работы: приобретение навыков по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, изучение основных методов модуля os для работы с файловой системой, получение аргументов командной строки.

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.](#)

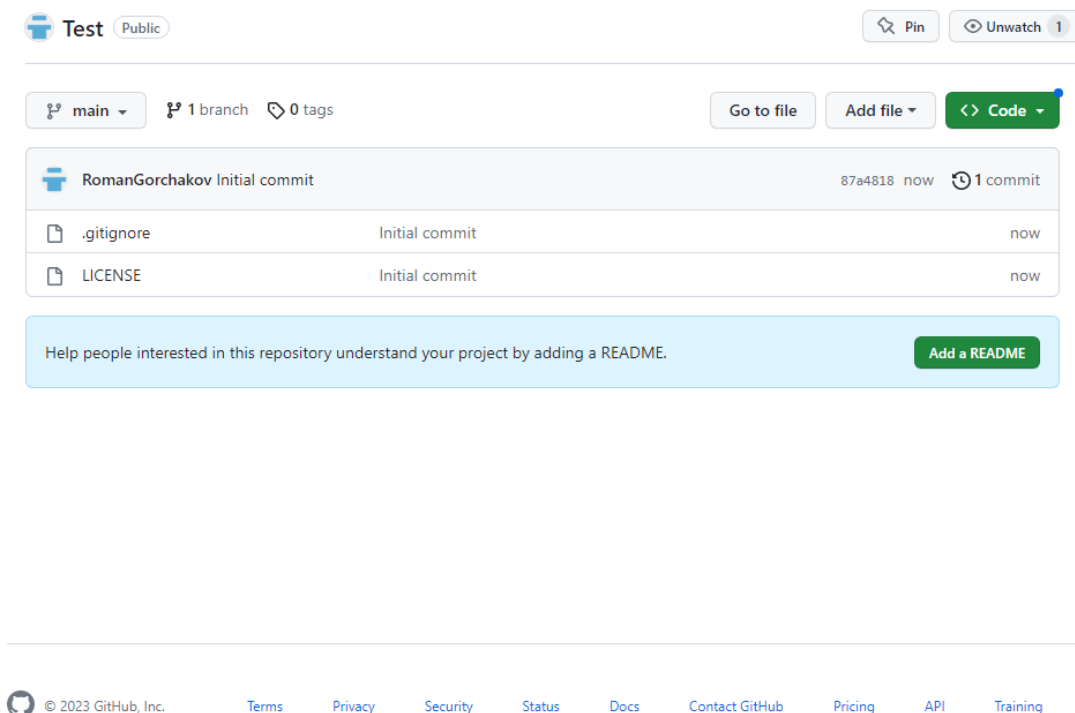
Choose a license

License: MIT License ▾


















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

③ 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» и запускаем терминал. 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/Py15 (main) $ git branch -d develop
Deleted branch develop (was 698e6c4).
● @RomanGorchakov →/workspaces/Py15 (main) $ git checkout -b develop
Switched to a new branch 'develop'
● @RomanGorchakov →/workspaces/Py15 (develop) $ git branch feature_branch
● @RomanGorchakov →/workspaces/Py15 (develop) $ git branch release/1.0.0
● @RomanGorchakov →/workspaces/Py15 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● @RomanGorchakov →/workspaces/Py15 (main) $ git branch hotfix
● @RomanGorchakov →/workspaces/Py15 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov →/workspaces/Py15 (develop) $
```

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

```
Python is the modern day language. It makes things so simple.  
It is the fastest growing programming language. Python has an easy syntax  
and user-friendly interaction.
```

```
Python is the modern day language. It makes things so simple.  
It is the fastest-growing programing language. Python has an easy syntax and user-friendly interaction.  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

```
['Python is the modern day language. It makes things so simple.\n', 'It is the fastest-growing programing language. Python has an easy syntax and user-friendly interaction.']  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

```
<_io.TextIOWrapper name='newfile.txt' mode='x' encoding='UTF-8'>  
File created successfully  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

```
UTF-8 is a variable-width character encoding used for electronic communication.  
UTF-8 is capable of encoding all 1,112,064 valid character code points.  
In Unicode using one to four one-byte (8-bit) code units.
```

```
UTF-8 is capable of encoding all 1,112,064 valid character code points.  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

```
The filepointer is at byte : 0  
After reading, the filepointer is at: 10  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

```
/home  
  
...Program finished with exit code 0  
Press ENTER to exit console.□
```

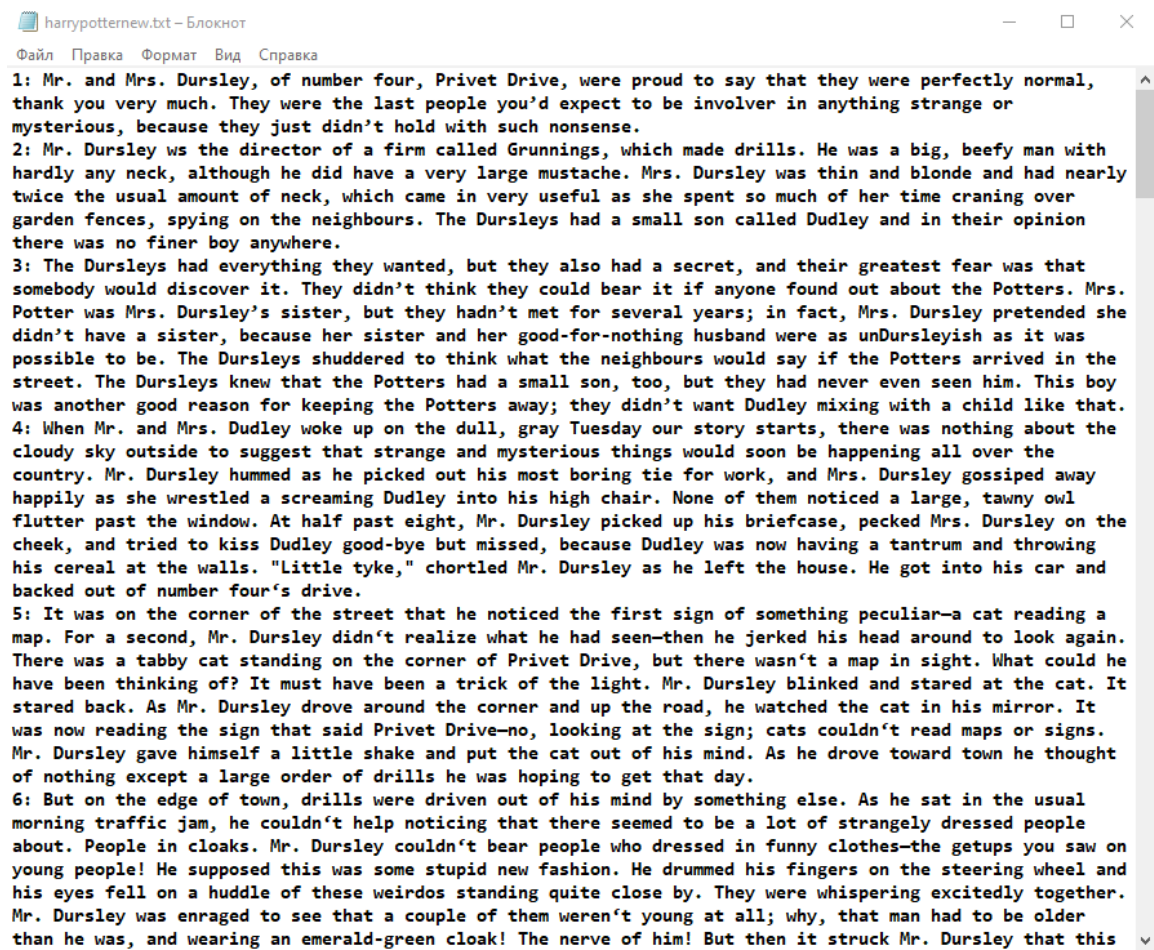
```
Number of arguments: 1 arguments  
Argument List: ['main.py']  
  
...Program finished with exit code 0  
Press ENTER to exit console.□
```

```
Argument #0 is main.py  
No. of arguments passed is 1  
  
...Program finished with exit code 0  
Press ENTER to exit console.□
```

6. Создаём файл «individual1.py», в котором пользователю нужно составить программу, считывающую английский текст с файла и выводящую на экран слова, начинающиеся с гласных букв.

```
you  
expect  
astonishing  
over  
inside  
up.  
on  
and  
on,  
in  
a  
as  
opened  
out  
and  
at  
in  
all  
over  
up  
and  
in  
  
...Program finished with exit code 0  
Press ENTER to exit console.□
```

7. Создаём файл «individual2.py», в котором пользователю нужно составить программу, которая будет считывать содержимое файла, добавлять к считанным строкам порядковый номер и сохранять их в таком виде в новом файле. Имя исходного файла необходимо запросить у пользователя, так же, как и имя целевого файла. Каждая строка в созданном файле должна начинаться с ее номера, двоеточия и пробела, после чего должен идти текст строки из исходного файла.




















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

```

file2.txt      | 2 ++
file3.txt      | Bin 0 -> 167 bytes
harrypotter.txt | 103 ++++++
harrypotternew.txt | 103 ++++++
individual1.py | 17 ++++++
individual2.py | 10 ++++++
newfile.txt    | 2 ++
text.txt       | 3 ++
25 files changed, 428 insertions(+)
create mode 100644 example1.py
create mode 100644 example10.py
create mode 100644 example11.py
create mode 100644 example12.py
create mode 100644 example13.py
create mode 100644 example14.py
create mode 100644 example15.py
create mode 100644 example16.py
create mode 100644 example17.py
create mode 100644 example2.py
create mode 100644 example3.py
create mode 100644 example4.py
create mode 100644 example5.py
create mode 100644 example6.py
create mode 100644 example7.py
create mode 100644 example8.py
create mode 100644 example9.py
create mode 100644 file2.txt
create mode 100644 file3.txt
create mode 100644 harrypotter.txt
create mode 100644 harrypotternew.txt
create mode 100644 individual1.py
create mode 100644 individual2.py
create mode 100644 newfile.txt
create mode 100644 text.txt
● @RomanGorchakov →/workspaces/Py15 (main) $ git push -u
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 2 threads
Compressing objects: 100% (26/26), done.
Writing objects: 100% (26/26), 15.04 KiB | 3.01 MiB/s, done.
Total 26 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/RomanGorchakov/Py15
  698e6c4..9d43297  main -> main
branch 'main' set up to track 'origin/main'.
○ @RomanGorchakov →/workspaces/Py15 (main) $ 

```


 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. Как открыть файл в языке Python только для чтения?
С помощью команды `open(<file-name>, «r»)`.
2. Как открыть файл в языке Python только для записи?
С помощью команды `open(<file-name>, «w»)`.
3. Как прочитать данные из файла в языке Python?
С помощью метода `read()`.
4. Как записать данные в файл в языке Python?
С помощью команды `open(<file-name>, «a»)`.
5. Как закрыть файл в языке Python?

С помощью метода `close()`.

6. Изучите самостоятельно работу конструкции `with ... as`. Каково ее назначение в языке Python? Где она может быть использована еще, помимо работы с файлами?

Конструкция `with ... as` используется для оборачивания выполнения блока инструкций менеджером контекста. Применяется данная конструкция для гарантии того, что критические функции выполнятся в любом случае. Самый распространенный пример использования этой конструкции - открытие файлов.

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

Для чтения содержимого файла можно использовать несколько методов:

- `read(size)` — считывает из файла указанное количество символов (или весь файл, если `size` не указан)
- `readline()` — считывает одну строку из файла
- `readlines()` — считывает все строки файла в список

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

`os.name` - имя операционной системы. Доступные варианты: `'posix'`, `'nt'`, `'mac'`, `'os2'`, `'ce'`, `'java'`.

`os.environ` - словарь переменных окружения. Изменяемый (можно добавлять и удалять переменные окружения).

`os.getlogin()` - имя пользователя, вошедшего в терминал (Unix).

`os.getpid()` - текущий `id` процесса.

`os.uname()` - информация об ОС. возвращает объект с атрибутами: `sysname` - имя операционной системы, `nodename` - имя машины в сети (определяется реализацией), `release` - релиз, `version` - версия, `machine` - идентификатор машины.

`os.access(path, mode, *, dir_fd=None, effective_ids=False, follow_symlinks=True)` - проверка доступа к объекту у текущего пользователя.

Флаги: `os.F_OK` - объект существует, `os.R_OK` - доступен на чтение, `os.W_OK` - доступен на запись, `os.X_OK` - доступен на исполнение.

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

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

`os.symlink(source, link_name, target_is_directory=False, *, dir_fd=None)` - создаёт символическую ссылку на объект.

`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).

`os.system(command)` - исполняет системную команду, возвращает код её завершения (в случае успеха 0).

`os.urandom(n)` - n случайных байт. Возможно использование этой функции в криптографических целях.

`os.path` - модуль, реализующий некоторые полезные функции на работы с путями.