

Лабораторная работа №10

Выполнил Урусов Максим

ИВТ-б-о-21-1

Цель: приобретение навыков по работе с функциями при написании программ 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 ^{*}

MaksimUrusov / Laba--2.14

Great repository names are short and memorable. Need inspiration? How about **super-lamp**?

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: Python ▾

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 REP10

D:\REP10>git clone https://github.com/Arsen445/2.15.git
Cloning into '2.15'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

D:\REP10>
```

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:\REP10\2.15>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:/REP10/2.15/.git/hooks]
```

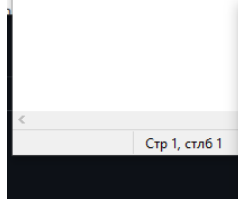
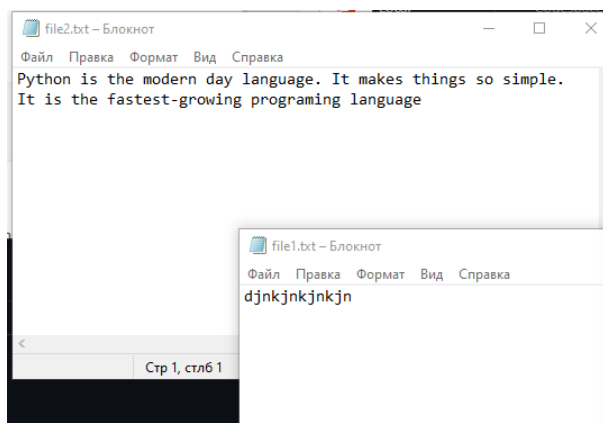
5. Создал проект пайчарм

Новый том (D:) > REP6 > LB2.11 >

Имя	Дата изменения	Тип	Размер
.idea	18.10.2022 1:22	Папка с файлами	
doc	03.11.2022 4:19	Папка с файлами	
.gitignore	18.10.2022 1:22	Файл "GITIGNORE"	6 КБ
LICENSE	03.11.2022 4:01	Файл	2 КБ
README.md	03.11.2022 4:01	Файл "MD"	1 КБ

6. Проработал примеры лабораторной работы.

Запись в файл



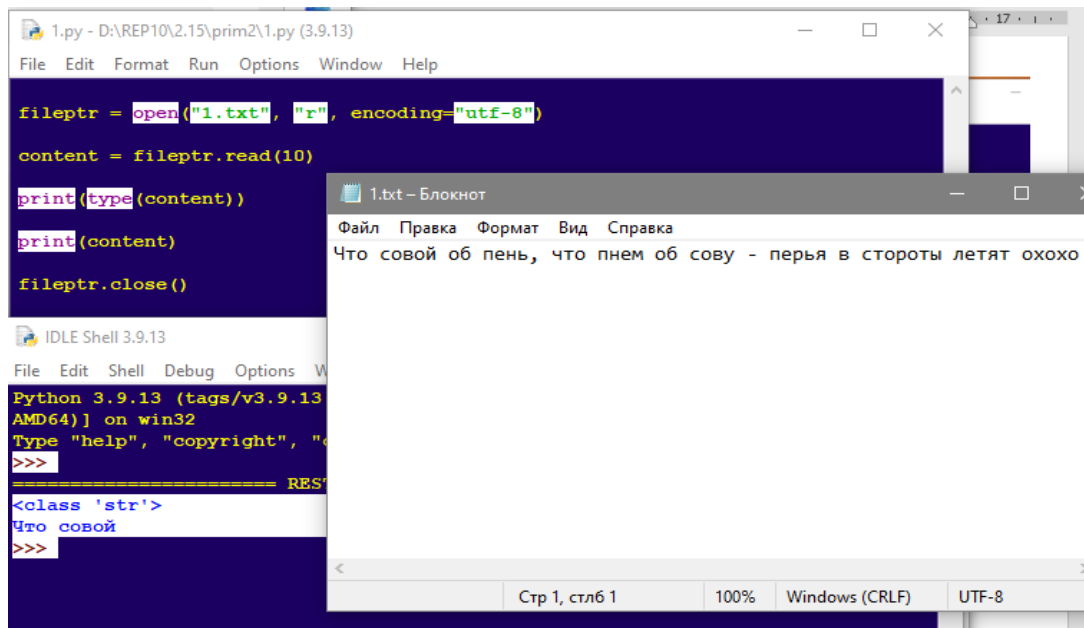
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def add_two(a):
    x = 2
    return a + x

if __name__ == '__main__':
    print(add_two(3))
    print(x)
```

```
5
Traceback (most recent call last):
  File "D:\REP6\LB2.11\prim1.py", line 11, in <module>
    print(x)
NameError: name 'x' is not defined
>>>
```

Чтение из файла



The screenshot shows the Python IDLE environment. The main editor window contains a Python script that opens a file named '1.txt' in read mode ('r') with UTF-8 encoding. It reads the first 10 characters of the file, prints the type of the content (which is a string), prints the content itself, and then closes the file. The output window shows the result of the print statements: the type is '<class 'str'>' and the content is 'Что совой'. A separate Notepad window titled '1.txt - Блокнот' shows the full text of the file: 'Что совой об пень, что пнем об сову - перья в стороты летят оохо'.

```
1.py - D:\REP10\2.15\prim2\1.py (3.9.13)
File Edit Format Run Options Window Help

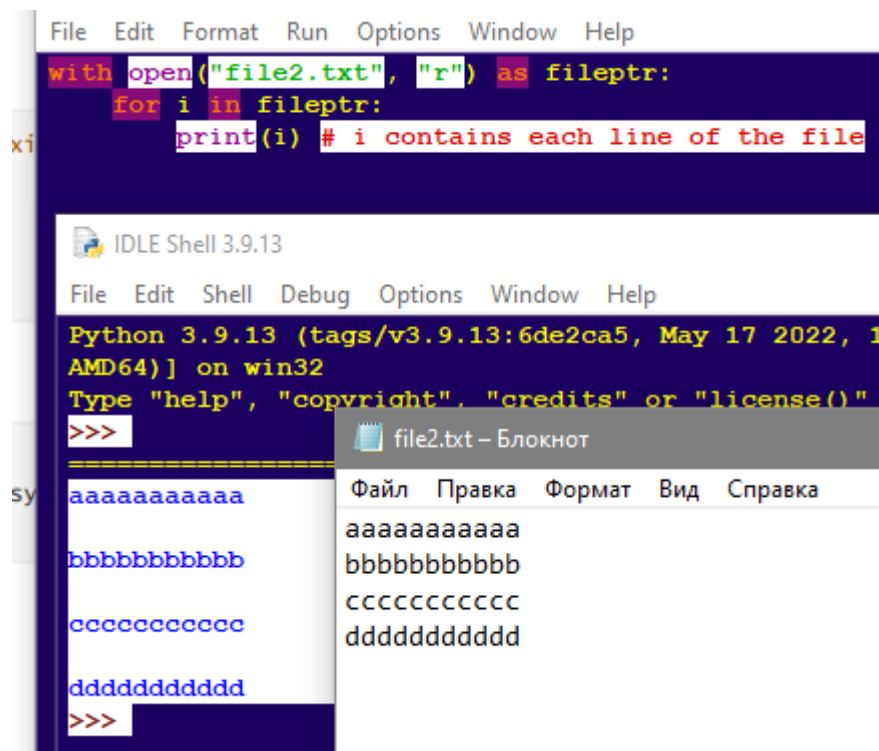
fileptr = open("1.txt", "r", encoding="utf-8")
content = fileptr.read(10)
print(type(content))
print(content)
fileptr.close()

IDLE Shell 3.9.13
File Edit Shell Debug Options Window Help
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 1
AMD64) on win32
Type "help", "copyright", "credits" or "license()"
>>>
===== RES
<class 'str'>
Что совой
>>>
```

1.txt - Блокнот
Файл Правка Формат Вид Справка
Что совой об пень, что пнем об сову - перья в стороты летят оохо

Стр 1, стлб 1 100% Windows (CRLF) UTF-8

Построчное чтение содержимого файла в цикле



The screenshot shows the Python IDLE environment. The main editor window contains a Python script that opens a file named 'file2.txt' in read mode ('r') and iterates over each line of the file using a for loop. For each line, it prints the line content. The output window shows the result of the print statements: the type is '<class 'str'>' and the content is 'aaaaa'. A separate Notepad window titled 'file2.txt - Блокнот' shows the full text of the file: 'aaaaa', 'bbbbb', 'ccccc', 'ddddd'.

```
File Edit Format Run Options Window Help

with open("file2.txt", "r") as fileptr:
    for i in fileptr:
        print(i) # i contains each line of the file

IDLE Shell 3.9.13
File Edit Shell Debug Options Window Help
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 1
AMD64) on win32
Type "help", "copyright", "credits" or "license()"
>>>
===== RES
<class 'str'>
aaaaa
>>>
```

file2.txt - Блокнот
Файл Правка Формат Вид Справка
aaaaa
bbbbb
ccccc
ddddd

Построчное чтение содержимого файла с помощью методов файлового объекта

Readline()

The screenshot shows the Python IDLE environment. The main editor window contains a Python script that opens a file named 'file2.txt' in read mode ('r') and uses the `readline()` method to read the file line by line. The first two lines are read and stored in `content1` and `content2`, which are then printed. Below the editor is the IDLE Shell, which shows the execution of the script. The output displays the first two lines of the file: 'wuehiuehoeijve' and 'khsviueuigrve'. The third line, 'khviue', is not yet printed because the script has not reached the `print(content2)` statement.

```
File Edit Format Run Options Window Help
with open("file2.txt", "r") as fileptr:
    content1 = fileptr.readline()
    content2 = fileptr.readline()

    print(content1)
    print(content2)
```

IDLE Shell 3.9.13

```
File Edit Shell Debug Options Window Help
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:44) on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
===== RESTART: D:\REP10\2.15\prim4\2.py
wuehiuehoeijve
khsviueuigrve
>>>
```

file2.txt – Блокнот

```
Файл Правка Форм
wuehiuehoeijve
khsviueuigrve
khviue
```

Readlines()

The screenshot shows the Python IDLE environment. The main editor window contains a Python script that opens a file named 'file2.txt' in read mode ('r') and uses the `readlines()` method to read the entire file at once. The resulting list of lines is stored in `content` and then printed. Below the editor is the IDLE Shell, which shows the execution of the script. The output displays the entire content of the file as a list of strings, each ending with a newline character: ['wuehiuehoeijve\n', 'khsviueuigrve\n', 'khviue\n']. The third line, 'khviue', is not yet printed because the script has not reached the `print(content)` statement.

```
File Edit Format Run Options Window Help
with open("file2.txt", "r") as fileptr:
    content = fileptr.readlines()

    print(content)
```

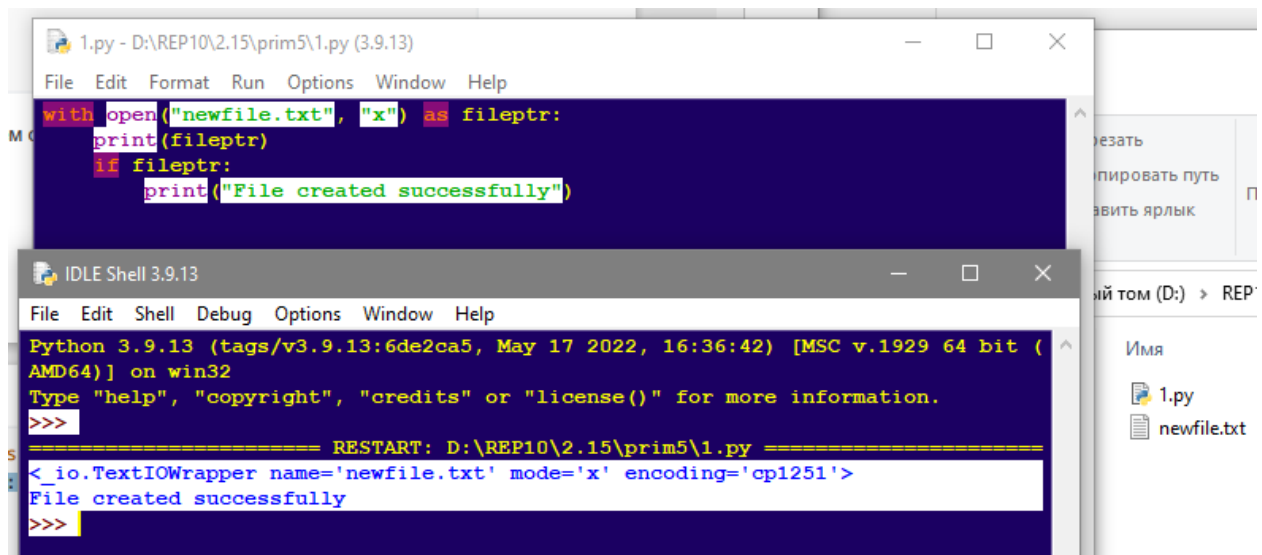
IDLE Shell 3.9.13

```
File Edit Shell Debug Options Window Help
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:44) on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
===== RESTART: D:\REP10\2.15\prim4\2.py
['wuehiuehoeijve\n', 'khsviueuigrve\n', 'khviue\n']
>>>
```

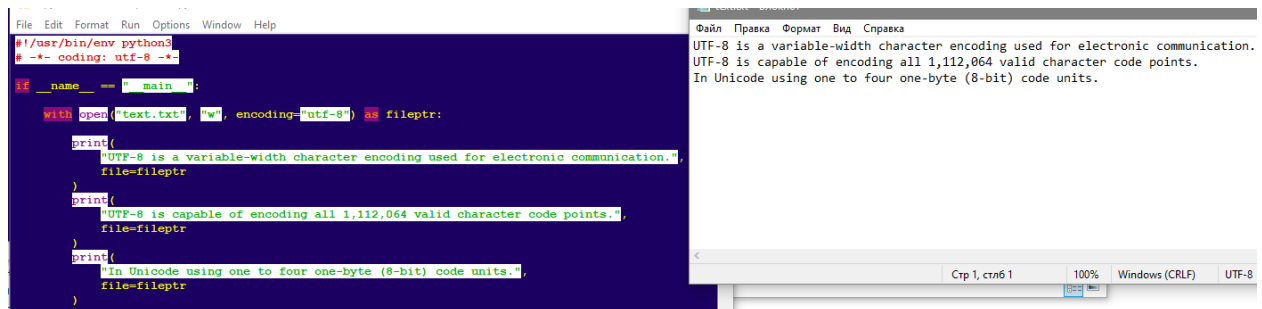
file2.txt – Блокнот

```
Файл Правка Форм
wuehiuehoeijve
khsviueuigrve
khviue
```

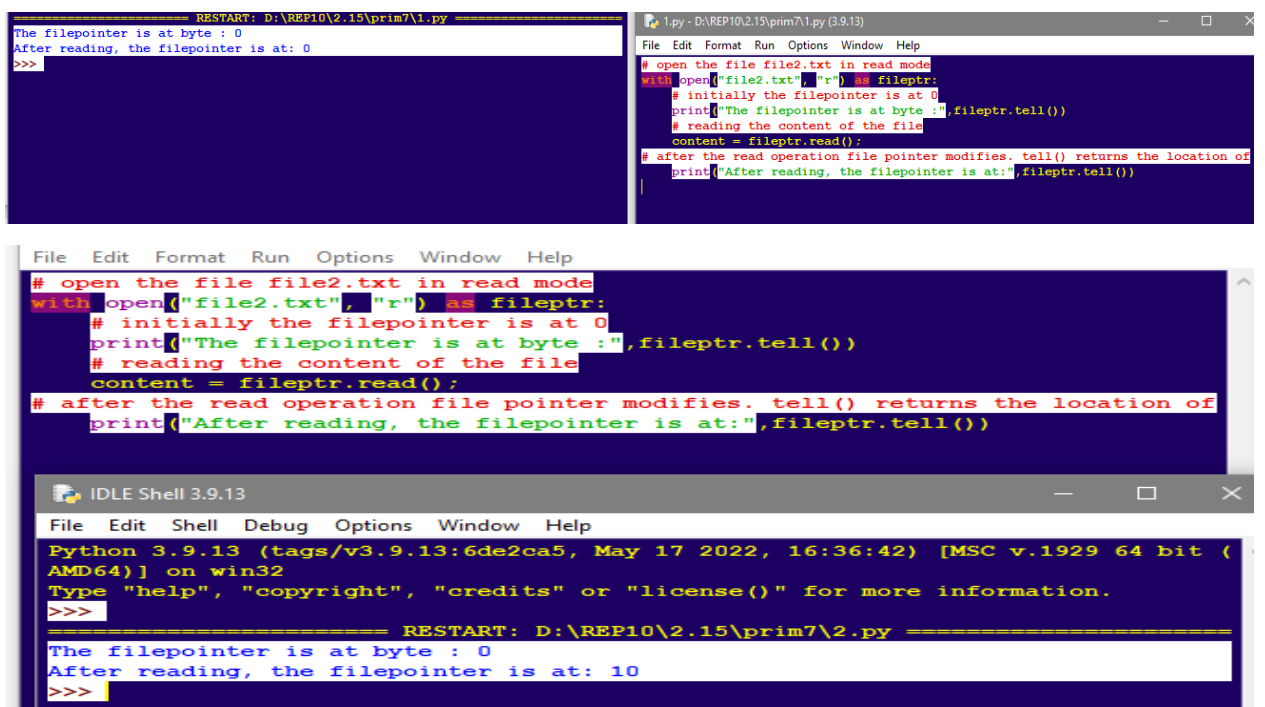
Создание нового файла



Изменение кодировки файла

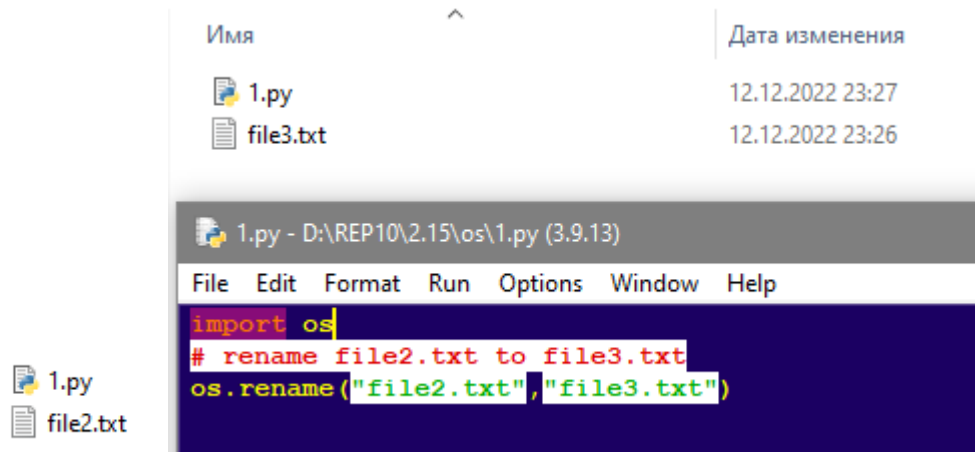


Позиция указателя файла

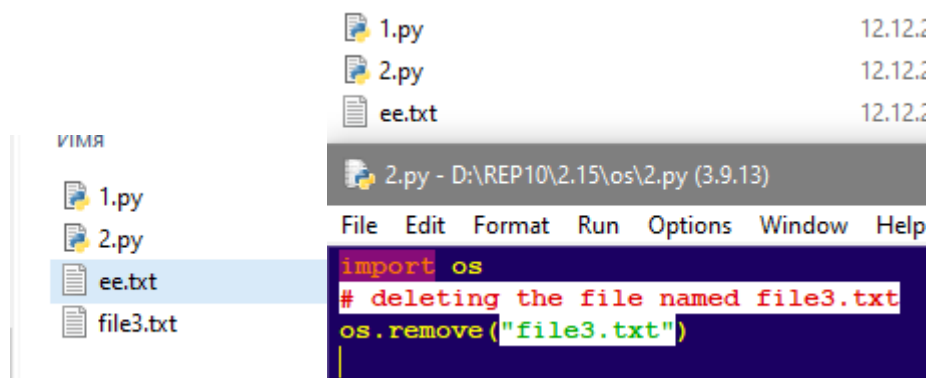


Модуль os

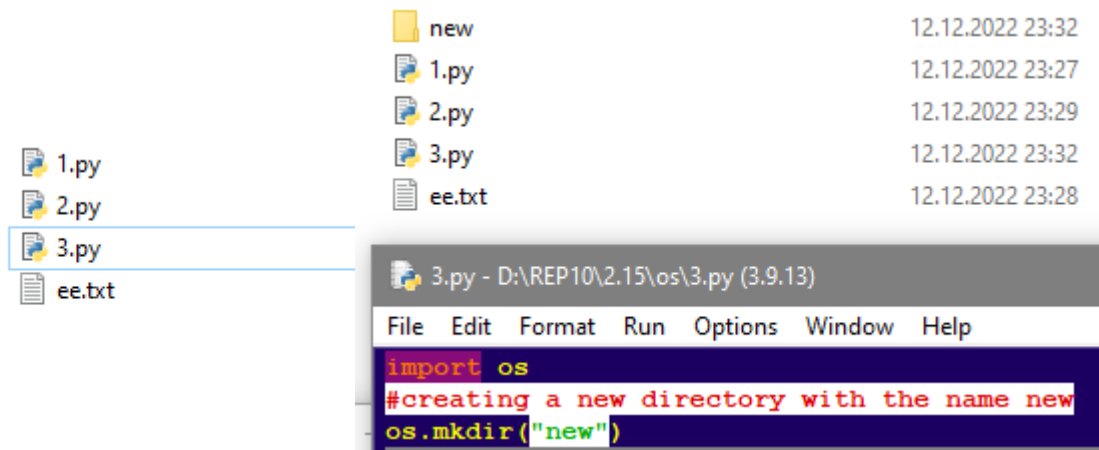
Модуль Python os обеспечивает взаимодействие с операционной системой. Модуль os предоставляет функции, которые участвуют в операциях обработки файлов, таких как переименование, удаление и т. д. Он предоставляет нам функцию `rename()` для переименования указанного файла в новое имя. Синтаксис для использования функции `rename()` приведен ниже.



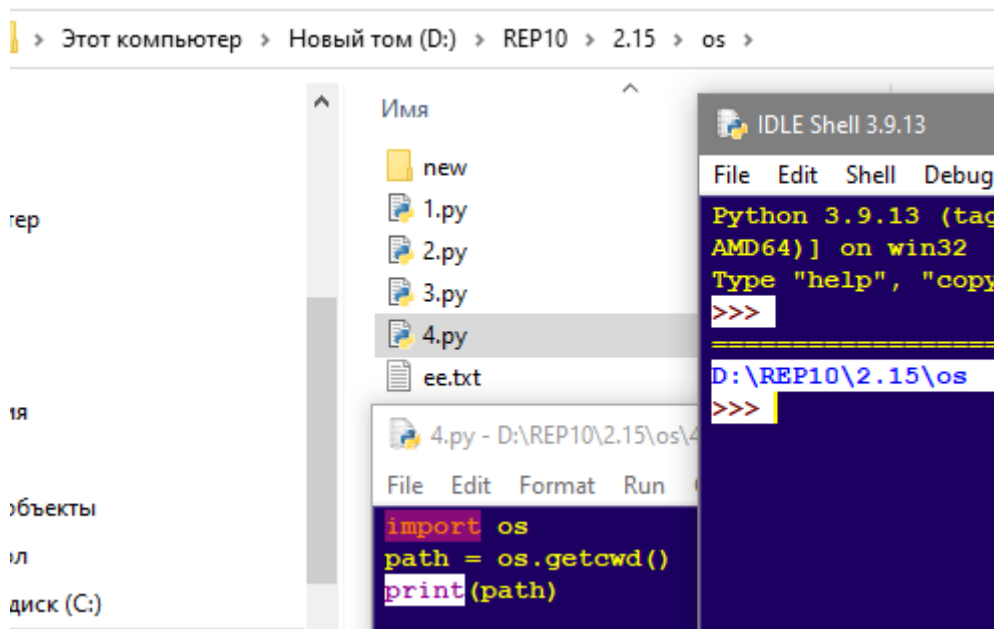
Удаление файла



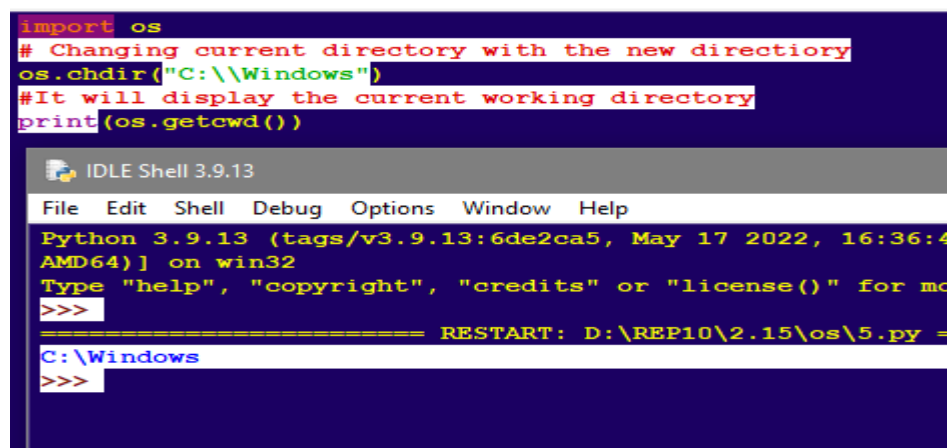
Создание нового каталога



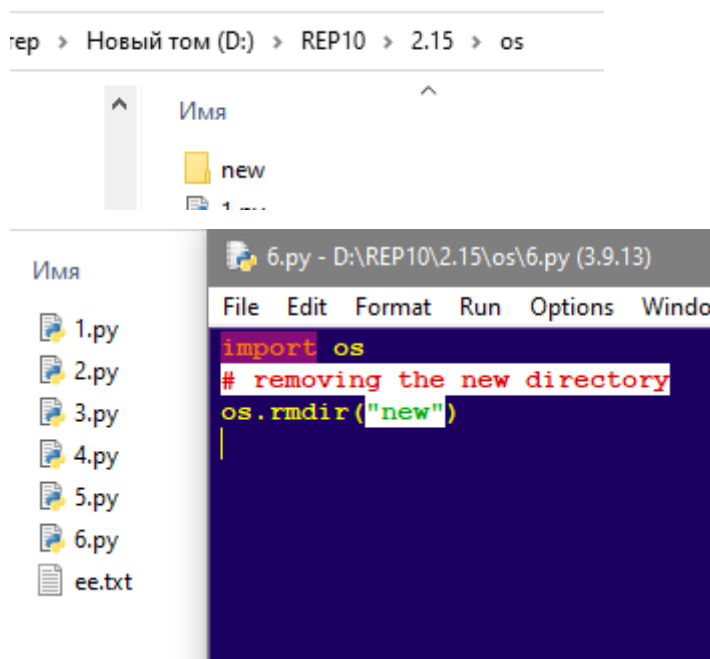
Получение текущего рабочего каталога



Изменение текущего рабочего каталога



Удаление каталога

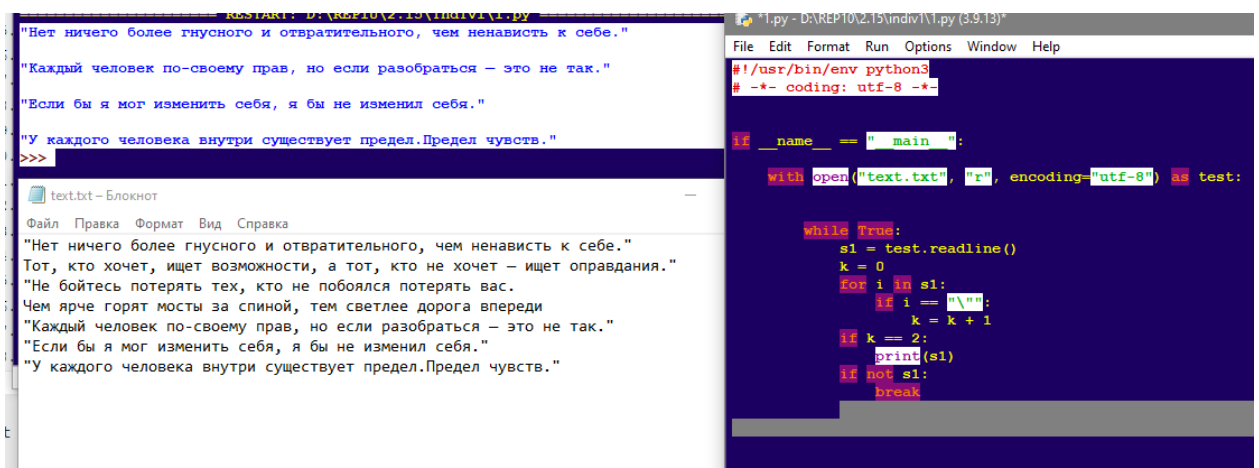


Индивидуальное 1

Составить программу с использованием списков и словарей для решения задачи. Номер варианта определяется по согласованию с преподавателем.

Исходный файл, из которого выполняется чтение, необходимо также добавить в репозиторий, каждое предложение в файле должно находиться на отдельной строке.

Написать программу, которая считывает текст из файла и выводит на экран только цитаты, то есть предложения, заключенные в кавычки.



Индивидуальное 2

Составить программу с использованием текстовых файлов. Номер варианта необходимо получить у преподавателя.

Автоматическая проверка орфографии не помешала бы многим из нас. В данном упражнении мы напишем простую программу, сверяющую слова из текстового файла со словарем. Неправильно написанными будем считать все слова, которых не нашлось в словаре. Имя файла, в котором требуется выполнить орфографическую проверку, пользователь должен передать при помощи аргумента командной строки. В случае отсутствия аргумента должна выдаваться соответствующая ошибка. Сообщение об ошибке также должно появляться, если не удастся открыть указанный пользователем файл. Также Вам следует игнорировать регистр символов при выполнении проверки.

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":

    a = {1: "sova", 2: "sich", 3: "mouse"}
    print(a)
    print(a[1])
    print(type(a[1]))

    print("Введите путь к файлу")
    f = input()

    with open(f, "r", encoding="utf-8") as test:

        b = [f for line in test for f in line.split()]
        print(b)
        print(b[0])
        print(type(b[0]))

        i = 0

        while i < len(b):
            q = b[i]
            print(b[i])
            print(type(b[i]))

            if a[1] == q.lower() or a[2] == q.lower() or a[3] == q.lower():
                print("Слово не имеет ошибок")
            else:
                print("В слове ошибка или его нет в списке")
            i = i+1

```

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

```

D:\REP6\LB2.11>git push --all
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (16/16), 92.88 KiB | 7.14 MiB/s, done.
Total 16 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/Arsen445/LB2.11.git
30b5e5e..c836547 develop -> develop

D:\REP6\LB2.11>git status
On branch develop
nothing to commit, working tree clean

D:\REP6\LB2.11>_

```

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

```
D:\REP6\LB2.11>git checkout main
Unlink of file 'doc/лб6.docx' failed. Should I try again? (y/n) y
Unlink of file 'doc/лб6.docx' failed. Should I try again? (y/n) n
warning: unable to unlink 'doc/лб6.docx': invalid argument
Updating files: 100% (13/13), done.
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

D:\REP6\LB2.11>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    doc/

nothing added to commit but untracked files present (use "git add" to track)

D:\REP6\LB2.11>git add --all
D:\REP6\LB2.11>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   "doc/\320\233\320\2216.docx"

D:\REP6\LB2.11>git merge develop
Updating 71773ee..f888dbd
Fast-forward
 .idea/.name | 1 +
 .idea/LB2.8.iml | 8 ++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++++
 .idea/modules.xml | 8 ++++++
 .idea/vcs.xml | 6 +++++
 "doc/-$\320\233\320\2216.docx" | Bin 0 -> 162 bytes
 "doc/\320\233\320\2216.docx" | Bin 0 -> 566932 bytes
 ind.py | 31 +++++++++++++++++++++++++++++++++++++
 prim1.py | 15 ++++++++
 prim2.py | 13 ++++++++
 prim3.py | 9 ++++++
```

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).