МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по курсовой работе

по дисциплине «Программирование»

Тема: Генерация отчетов

Студент гр. 7382 _____ Глазунов С.А.

Преподаватель _____ Кринкин К.В.

Санкт-Петербург
2018

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Глазунов С.А.

Группа 7382

Тема работы : Генерация отчетов

Исходные данные: В качестве основы для курсовой работы используется код лабораторной работы No4.

Содержание пояснительной записки:«Введение», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 5 страниц.

Дата выдачи задания: 28.11.2017

Дата сдачи реферата: 23.12.2017

Дата защиты реферата: 23.12.2017

Студент _____ Глазунов С.А.

Преподаватель _____ Кринкин К.В.

Аннотация.

Необходимо, имея код лабораторной работы No4, реализовать алгоритм,который делает следующие преобразования со списком:

Отсортировать список по невозрастанию по полю year в этом списке; Менять местами элементы не затрагивая поля, кроме тех, что указывают наследующие и предыдущие элементы; Пишется две функции, которые производят все эти преобразования и возвращают головной элемент списка.

Приложение

main.py

```
#!./venv/bin/python3.6
import argparse
import sys
import os
import shutil
from mygithub import Gengit
from word import Dword


TIME_REPORT = "ready_project.docx"
READY_WORD = "generated_doc.docx"
FROM_CONSOLE = "cmd"
PDF = "PDF"
PDF_EXTENSION = "{}.pdf"
ERROR_MESSAGE = "Неверные входные данныk!"
INPUT_URL_MESSAGE = "url of repo(ssh): "
INPUT_WIKI_URL_MESSAGE = "wiki repo(http): "
INPUT_BRANCH_MESSAGE = "branch: "
LEN_WORD_EXTENSION = 5
DELETE_WORD = False
DELETED_PICTURE = "picture"
FLAG_ARG = "-f"


def create_parser():
    parser = argparse.ArgumentParser()
    parser.add_argument(FLAG_ARG)
    return parser


def input_cmd():
    url = input(INPUT_URL_MESSAGE)
    wiki_url = input(INPUT_WIKI_URL_MESSAGE)
    branch = input(INPUT_BRANCH_MESSAGE)
    return url, wiki_url, branch


def delete_directories_and_files(git, git_wiki):
    if os.path.exists(git.local_repo):
        shutil.rmtree(git.local_repo)
    if os.path.exists(git.local_wiki):
        shutil.rmtree(git_wiki.local_wiki)
    if os.path.exists(READY_WORD) and DELETE_WORD:
        os.remove(READY_WORD)
    if os.path.exists(DELETED_PICTURE):
        os.remove(DELETED_PICTURE)


def input_file(name):
    with open(name) as file:
        content = file.readlines()
    content = [element.strip() for element in content]
    return content[:3]


def main(type_of_input):
    if type_of_input is FROM_CONSOLE:
        url, wiki_url, branch = input_cmd()
    else:
        url, wiki_url, branch = input_file(type_of_input)
    git = Gengit(url, branch)
```

```
    git_wiki = Gengit(wiki_url)

    if git.download_git() is False or git_wiki.download_git_wiki() is False:
        delete_directories_and_files(git, git_wiki)
        print(ERROR_MESSAGE)

    word = Dword()
    path_doc = os.path.join(git.local_repo, TIME_REPORT)
    word.save(path_doc)
    if word.js_content[PDF]:
        word.convert_to_pdf(docname=path_doc)
        git.add(PDF_EXTENSION.format(TIME_REPORT[:-LEN_WORD_EXTENSION]))
    else:
        git.add(TIME_REPORT)
    git.push()
    delete_directories_and_files(git, git_wiki)


if __name__ == "__main__":
    parser = create_parser()
    namespace = parser.parse_args()

    if not namespace.f:
        main(FROM_CONSOLE)
    else:
        main(namespace.f)
```

main.py

```
print('main file')
```

func.py

```
print('func file')
```