Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

**дисциплины «Объектно-ориентированное программирование»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Кожуховский Виктор Андреевич  3 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем  », очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Проверил:  Воронкин Роман Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024 г.

Тема: Классы данных в Python

Цель: приобретение навыков по работе с классами данных при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

1. Изучил теоретический материал работы.

2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

3. Выполнил клонирование созданного репозитория.

4. Дополнил файл .gitignore необходимыми правилами для работы с IDE.

5. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

6. Создал проект в папке репозитория.

7. Проработал примеры лабораторной работы.

8. Выполнил индивидуальное задание для варианта 14.

Задание 1

Выполнить индивидуальное задание лабораторной работы 4.5, использовав классы данных, а также загрузку и сохранение данных в формат XML.

import os

import argparse

import stat

import xml.etree.ElementTree as ET

from dataclasses import dataclass, field

from typing import List, Union

@dataclass

class File:

name: str

size: int

@dataclass

class Directory:

path: str

files: List[File] = field(default\_factory=list)

subdirs: List["Directory"] = field(default\_factory=list)

def is\_hidden(filepath: str) -> bool:

if os.path.basename(filepath).startswith("."):

return True

try:

return bool(os.stat(filepath).st\_file\_attributes & stat.FILE\_ATTRIBUTE\_HIDDEN)

except AttributeError:

return False

def list\_files(

startpath: str, depth: int = -1, level: int = 0, show\_hidden: bool = False

) -> Directory:

if depth == 0:

return Directory(path=startpath)

directory = Directory(path=startpath)

for element in os.listdir(startpath):

path = os.path.join(startpath, element)

if (element.startswith(".") or is\_hidden(path)) and not show\_hidden:

continue

if os.path.isdir(path):

sub\_directory = list\_files(path, depth - 1, level + 1, show\_hidden)

directory.subdirs.append(sub\_directory)

else:

size = os.path.getsize(path)

directory.files.append(File(name=element, size=size))

return directory

def save\_to\_xml(directory: Directory, xml\_file: Union[str, ET.Element]) -> ET.Element:

if isinstance(xml\_file, str):

root = ET.Element("directory", path=directory.path)

else:

root = xml\_file

root.set("path", directory.path)

for file in directory.files:

ET.SubElement(root, "file", name=file.name, size=str(file.size))

for subdir in directory.subdirs:

subdir\_element = ET.SubElement(root, "subdirectory")

save\_to\_xml(subdir, subdir\_element)

if isinstance(xml\_file, str):

tree = ET.ElementTree(root)

tree.write(xml\_file, encoding="utf-8", xml\_declaration=True)

return root

def load\_from\_xml(xml\_source: Union[str, ET.Element]) -> Directory:

if isinstance(xml\_source, str):

tree = ET.parse(xml\_source)

root = tree.getroot()

else:

root = xml\_source

directory = Directory(path=root.get("path", ""))

for file\_element in root.findall("file"):

name = file\_element.get("name", "")

size = int(file\_element.get("size", 0))

directory.files.append(File(name=name, size=size))

for subdir\_element in root.findall("subdirectory"):

subdir = load\_from\_xml(subdir\_element)

directory.subdirs.append(subdir)

return directory

def print\_directory(directory: Directory, level: int = 0) -> None:

indent = " " \* 4 \* level

for file in directory.files:

print(f"{indent}└── {file.name} ({file.size} bytes)")

for subdir in directory.subdirs:

print(f"{indent}└── {os.path.basename(subdir.path)}/")

print\_directory(subdir, level + 1)

def main() -> None:

parser = argparse.ArgumentParser(description="Display directory tree")

parser.add\_argument("path", nargs="?", default=".", help="Directory path")

parser.add\_argument("-d", "--depth", type=int, default=-1, help="Depth of the tree")

parser.add\_argument("--hidden", action="store\_true", help="Show hidden files")

parser.add\_argument("--save", type=str, help="Save directory structure to XML file")

parser.add\_argument(

"--load", type=str, help="Load directory structure from XML file"

)

args = parser.parse\_args()

if args.load:

directory = load\_from\_xml(args.load)

print(f"Loaded directory from {args.load}: {directory.path}")

print\_directory(directory)

else:

directory = list\_files(args.path, args.depth, show\_hidden=args.hidden)

if args.save:

save\_to\_xml(directory, args.save)

print(f"Saved directory structure to {args.save}")

else:

print\_directory(directory)

if \_\_name\_\_ == "\_\_main\_\_":

main()

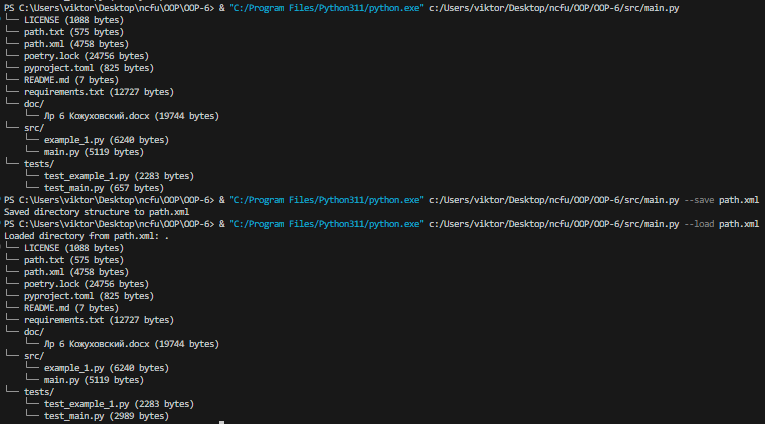


Рисунок 1. Код решения индивидуального задания 1 и его выполнение

9. Зафиксировал сделанные изменения в репозитории.

10. Выполнил слияние ветки для разработки с веткой master/main.

11. Отправил сделанные изменения на сервер GitHub.

Ссылка: https://github.com/Viktorkozh/OOP-6

Контрольные вопросы:

1. Как создать класс данных в языке Python?

from dataclasses import dataclass

@dataclass

class DataClassCard:

rank: str

suit: str

2. Какие методы по умолчанию реализует класс данных?

По умолчанию, классы данных реализует метод .\_\_repr\_\_() , чтобы предоставить хорошее строковое представление, а также метод .\_\_eq\_\_() , который в состоянии выполнять базовые сравнения объектов.

3. Как создать неизменяемый класс данных?

Чтобы сделать класс данных неизменяемым, установите frozen=True при создании.

Вывод: приобрел навыки по работе с классами данных при написании программ с помощью языка программирования Python версии 3.x.