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 №4.6**

**дисциплины «Объектно-ориентированное программирование»**

**Вариант 9**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Иващенко Олег Андреевич  3 курс, группа ИВТ-б-о-22-1,  09.03.02 «Информационные и вычислительные машины», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Руководитель практики:  Воронкин Роман Александрович, доцент департамента цифровых, робототехнических систем и электроники  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

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

Ставрополь, 2024 г.

**Тема**: «Классы данных»

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

Порядок выполнения работы:

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

Таблица 1 – Листинг программы

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  """  Выполнить индивидуальное задание лабораторной работы 4.5, использовав  классы данных, а также загрузку и сохранение данных в формат XML.  """  import os  import argparse  import xml.etree.ElementTree as ET  class FileNode:  """Класс для представления узла дерева (файла или папки)."""  def \_\_init\_\_(self, name, is\_dir):  """Инициализация узла."""  self.name = name  self.is\_dir = is\_dir  self.children = []  def build\_tree(path, level, max\_level, show\_hidden):  """  Рекурсивно строит дерево каталогов.  :param path: Путь к текущему каталогу.  :param level: Текущий уровень вложенности.  :param max\_level: Максимальный уровень вложенности.  :param show\_hidden: Показывать ли скрытые файлы.  :return: Узел дерева.  """  if level > max\_level:  return None  name = os.path.basename(path)  is\_dir = os.path.isdir(path)  node = FileNode(name, is\_dir)  if is\_dir:  for item in os.listdir(path):  if not show\_hidden and item.startswith('.'):  continue  child\_path = os.path.join(path, item)  child\_node = build\_tree(child\_path, level + 1, max\_level, show\_hidden)  if child\_node:  node.children.append(child\_node)  return node  def print\_tree(node, level=0):  """  Выводит дерево каталогов на экран.  :param node: Корневой узел дерева.  :param level: Текущий уровень вложенности.  """  if not node:  return  indent = " " \* level  print(f"{indent}/{node.name}" if node.is\_dir else f"{indent}{node.name}")  for child in node.children:  print\_tree(child, level + 1)  def save\_to\_xml(node, parent\_element=None):  """  Сохраняет дерево каталогов в XML.  :param node: Корневой узел дерева.  :param parent\_element: Родительский XML-элемент.  :return: XML-элемент.  """  if parent\_element is None:  element = ET.Element("directory", {"name": node.name})  else:  element = ET.SubElement(  parent\_element,  "directory" if node.is\_dir else "file",  {"name": node.name}  )  for child in node.children:  save\_to\_xml(child, element)  return element  def load\_from\_xml(element):  """  Загружает дерево каталогов из XML.  :param element: XML-элемент.  :return: Узел дерева.  """  name = element.attrib["name"]  is\_dir = element.tag == "directory"  node = FileNode(name, is\_dir)  for child\_element in element:  child\_node = load\_from\_xml(child\_element)  node.children.append(child\_node)  return node  def main():  """Основная функция программы."""  parser = argparse.ArgumentParser(description="Утилита для отображения дерева каталогов.")  parser.add\_argument("directory", nargs="?", default=".", help="Путь к каталогу")  parser.add\_argument("-l", "--level", type=int, default=100,  help="Максимальная глубина отображения")  parser.add\_argument("-a", "--all", action="store\_true",  help="Показывать скрытые файлы")  parser.add\_argument("--save", type=str, help="Сохранить дерево в XML файл")  parser.add\_argument("--load", type=str, help="Загрузить дерево из XML файла")  args = parser.parse\_args()  if args.load:  tree\_xml = ET.parse(args.load)  root = load\_from\_xml(tree\_xml.getroot())  print("Дерево загружено из XML:")  print\_tree(root)  return  path = os.path.abspath(args.directory)  if not os.path.exists(path):  print("Ошибка: каталог не существует.")  return  if not os.path.isdir(path):  print("Ошибка: указанный путь не является каталогом.")  return  root = build\_tree(path, 0, args.level, args.all)  print(f"Дерево каталогов для {path}:")  print\_tree(root)  if args.save:  root\_xml = save\_to\_xml(root)  tree\_xml = ET.ElementTree(root\_xml)  tree\_xml.write(args.save, encoding="utf-8", xml\_declaration=True)  print(f"Дерево сохранено в {args.save}")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

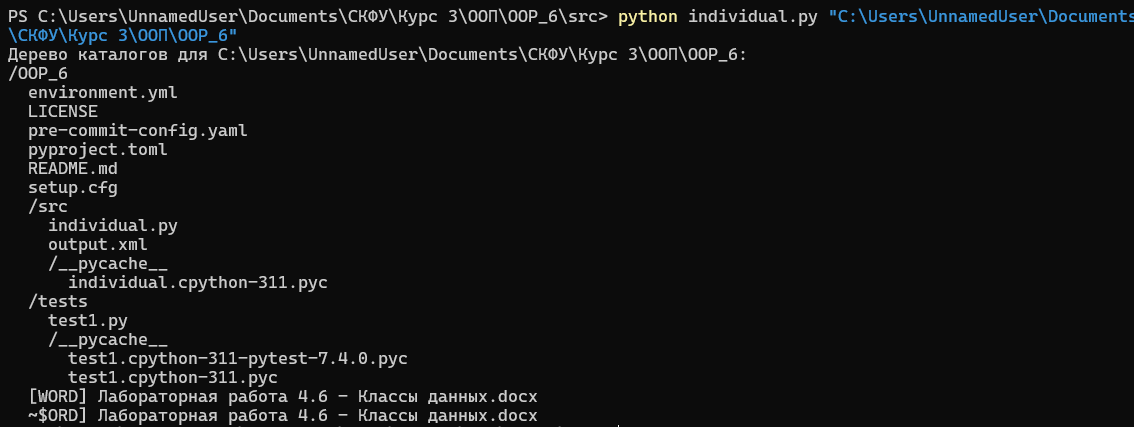


Рисунок 1 – Вывод программы

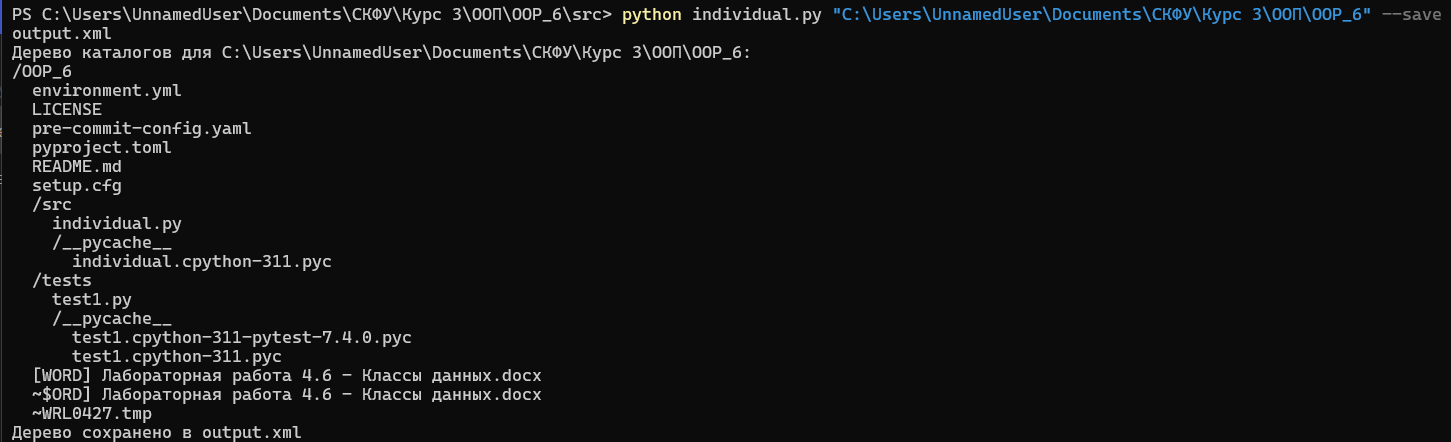


Рисунок 2 – Сохранение дерева каталогов в файл

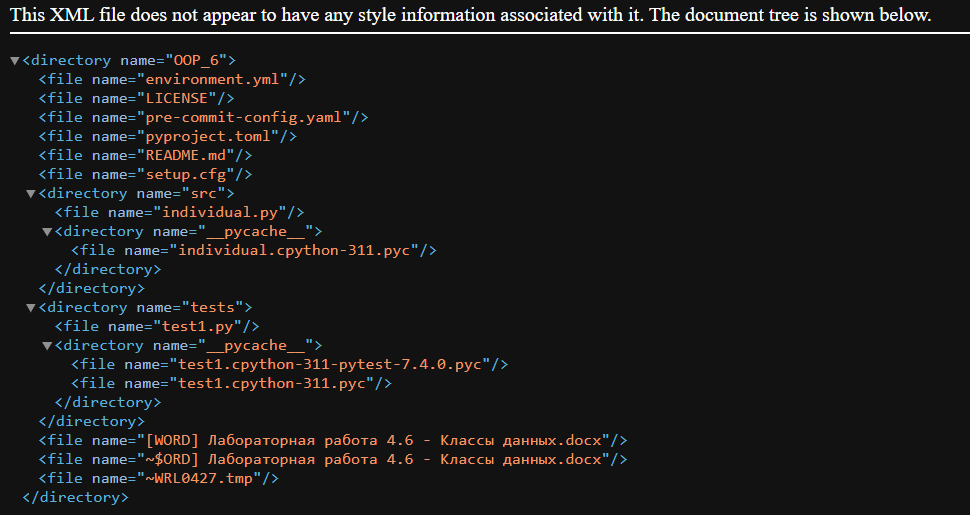


Рисунок 3 – Содержание сохранённого файла

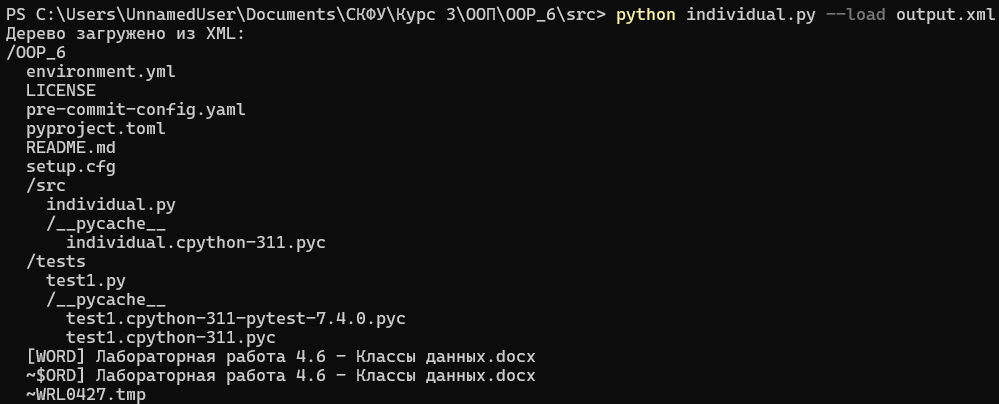


Рисунок 4 – Вывод сохранённого файла в консоль

Таблица 2 – Листинг теста

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import os  import xml.etree.ElementTree as ET  from src.individual import FileNode, build\_tree, print\_tree, save\_to\_xml, load\_from\_xml  def test\_build\_tree():  test\_dir = "test\_dir"  os.makedirs(test\_dir, exist\_ok=True)  with open(os.path.join(test\_dir, "file1.txt"), "w") as f:  f.write("test")  os.makedirs(os.path.join(test\_dir, "subdir"), exist\_ok=True)  root = build\_tree(test\_dir, 0, 10, False)  assert root.name == "test\_dir"  assert root.is\_dir is True  assert len(root.children) == 2  assert root.children[0].name == "file1.txt"  assert root.children[0].is\_dir is False  assert root.children[1].name == "subdir"  assert root.children[1].is\_dir is True  os.remove(os.path.join(test\_dir, "file1.txt"))  os.rmdir(os.path.join(test\_dir, "subdir"))  os.rmdir(test\_dir)  def test\_save\_and\_load\_xml():  root = FileNode("test\_dir", True)  file\_node = FileNode("file1.txt", False)  subdir\_node = FileNode("subdir", True)  root.children.append(file\_node)  root.children.append(subdir\_node)  xml\_file = "test.xml"  root\_xml = save\_to\_xml(root)  tree\_xml = ET.ElementTree(root\_xml)  tree\_xml.write(xml\_file, encoding="utf-8", xml\_declaration=True)  tree\_xml = ET.parse(xml\_file)  loaded\_root = load\_from\_xml(tree\_xml.getroot())  assert loaded\_root.name == "test\_dir"  assert loaded\_root.is\_dir is True  assert len(loaded\_root.children) == 2  assert loaded\_root.children[0].name == "file1.txt"  assert loaded\_root.children[0].is\_dir is False  assert loaded\_root.children[1].name == "subdir"  assert loaded\_root.children[1].is\_dir is True  os.remove(xml\_file)  def test\_print\_tree(capsys):  root = FileNode("test\_dir", True)  file\_node = FileNode("file1.txt", False)  subdir\_node = FileNode("subdir", True)  root.children.append(file\_node)  root.children.append(subdir\_node)  print\_tree(root)  captured = capsys.readouterr()  expected\_output = "/test\_dir\n file1.txt\n /subdir\n"  assert captured.out == expected\_output  if \_\_name\_\_ == "\_\_main\_\_":  test\_build\_tree()  test\_save\_and\_load\_xml()  test\_print\_tree() |

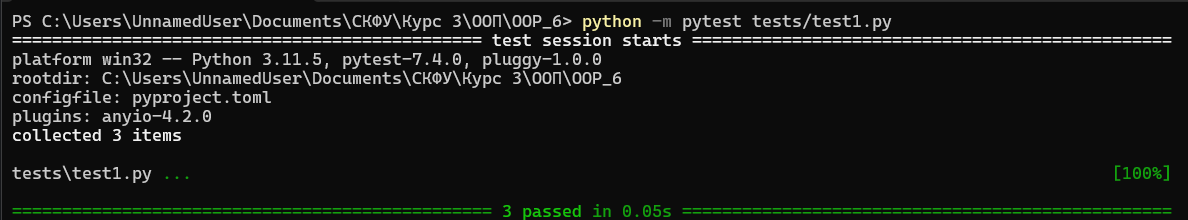


Рисунок 5 – Вывод теста программы

Ответы на контрольные вопросы:

1. Как создать класс данных в языке Python?

Класс данных создается с помощью декоратора @dataclass из модуля dataclasses. Этот декоратор автоматически добавляет стандартные методы, такие как \_\_init\_\_, \_\_repr\_\_ и \_\_eq\_\_, на основе объявленных атрибутов.

1. Какие методы по умолчанию реализует класс данных?

Класс данных автоматически реализует следующие методы:

* \_\_init\_\_ - конструктор для инициализации атрибутов.
* \_\_repr\_\_ - метод для строкового представления объекта (удобный вывод).
* \_\_eq\_\_ - метод для сравнения объектов по значениям атрибутов.

1. Как создать неизменяемый класс данных?

Чтобы сделать класс данных неизменяемым, используется параметр frozen=True в декораторе @dataclass, запрещающий изменение атрибутов после создания объекта.

Вывод: В процессе выполнения лабораторной работы были приобретены навыки по работе с классами данных при написании программ с помощью языка программирования Python версии 3.x.

Ссылка на репозиторий GitHub:

[IUnnamedUserI/OOP\_6: Объектно-ориентированное программирование. Лабораторная работа №6](https://github.com/IUnnamedUserI/OOP_6)