Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИУК «Информатика и управление»**

КАФЕДРА **ИУК4 «Программное обеспечение ЭВМ, информационные технологии»**

ДОМАШНЯЯ РАБОТА №2

«Работа с файлами»

ДИСЦИПЛИНА: «Перспективные языки программирования»

Выполнил: студент гр. ИУК4-31Б _____ (Суриков Н. С.)
(подпись) (Ф.И.О.)

Проверил: _____ (Осипова О. В.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2024

Цель: приобретение практических навыков работы с файлами и файловой системой на языке программирования Python.

Задачи:

1. Ознакомиться со способами работы с файлами и файловой системой в python;
2. Изучить способы работы с файлами формата csv, json, xml;
3. Закрепить полученные в ходе выполнения домашней работы навыки.

Вариант 5

Задача 1

Выберите любую папку на своем компьютере, имеющуювложенные директории.

Выведите на печать в терминал и сохраните в файл txt названия всех файлов начинающихся на букву а.

Заархивируйте данную папку средствами python.

Листинг программы:

```
import os
import zipfile


files_starting_with_a = []
test_dir = "test_folder"
zip_filename = "test_folder.zip"

for root, dirs, files in os.walk(test_dir):
    for file in files:
        if file.startswith("a"):
            files_starting_with_a.append(os.path.join(root, file))

with open("files_starting_with_a.txt", "w") as f:
    for file in files_starting_with_a:
        f.write(file + "\n")

with zipfile.ZipFile(zip_filename, "w", zipfile.ZIP_DEFLATED) as zipf:
    for root, dirs, files in os.walk(test_dir):
        for file in files:
            file_path = os.path.join(root, file)
```

```
22              zipf.write(file_path, os.path.relpath(file_path, test_dir))
```

Результат работы программы:

Содержимое тестовой папки:

```
├── subfolder1
│   ├── apple.txt
│   └── banana.txt
└── subfolder2
    └── avocado.txt
```

Вывод программы:

```
test_folder/subfolder1/apple.txt
test_folder/subfolder2/avocado.txt
```

Задача 2

Создать файл в формате xml, содержащий сведения о автомобилях. Структура записи: гос. номер, марка, цвет, ФИО владельца.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по гос. номеру);

2 - добавить новую запись;

3 - изменить запись (по гос. номеру);

4 - получить информацию (по гос. номеру).

5 – отсортировать автомобили по:

- *Марке автомобиля*
- *Цвету*
- *ФИО владельца*

6 – сохранить в отдельный файл и вывести на экран все автомобили, которыми владеет заданный человек по ФИО.

Листинг программы:

```
1  import xml.etree.ElementTree as ET
2  import os
3
4
5  class CarManager:
6      def __init__(self, xml_file):
7          self.xml_file = xml_file
8          self.create_xml_file()
```

```
 9
10      def create_xml_file(self):
11          """Создает XML-файл, если он не существует."""
12          if not os.path.exists(self.xml_file):
13              root = ET.Element("cars")
14              self.add_test_data(root)
15              tree = ET.ElementTree(root)
16              tree.write(self.xml_file)
17
18      def add_test_data(self, root):
19          """Добавляет тестовые данные о автомобилях."""
20          test_cars = [
21                  {"license_plate": "A123BC", "brand": "Toyota", "color": "Red",
"owner": "Ivanov Ivan"},
22                  {"license_plate": "B456CD", "brand": "BMW", "color": "Black",
"owner": "Petrov Petr"},
23                  {"license_plate": "C789EF", "brand": "Audi", "color": "Blue",
"owner": "Sidorov Sidor"},
24          ]
25
26          for car in test_cars:
27              car_element = ET.Element("car")
28                          ET.SubElement(car_element,  "license_plate").text  =
car["license_plate"]
29              ET.SubElement(car_element, "brand").text = car["brand"]
30              ET.SubElement(car_element, "color").text = car["color"]
31              ET.SubElement(car_element, "owner").text = car["owner"]
32              root.append(car_element)
33
34      def add_car(self, license_plate, brand, color, owner):
35          """Добавляет новую запись о автомобиле."""
36          tree = ET.parse(self.xml_file)
37          root = tree.getroot()
38
39          car = ET.Element("car")
40          ET.SubElement(car, "license_plate").text = license_plate
41          ET.SubElement(car, "brand").text = brand
42          ET.SubElement(car, "color").text = color
43          ET.SubElement(car, "owner").text = owner
44
45          root.append(car)
46          tree.write(self.xml_file)
47
48      def remove_car(self, license_plate):
49          """Удаляет запись о автомобиле по гос. номеру."""
50          tree = ET.parse(self.xml_file)
51          root = tree.getroot()
52
53          for car in root.findall("car"):
54              if car.find("license_plate").text == license_plate:
55                  root.remove(car)
56                  break
57
58          tree.write(self.xml_file)
59
```

```
    def update_car(self, license_plate, brand=None, color=None, owner=None):
        """Изменяет запись о автомобиле по гос. номеру."""
        tree = ET.parse(self.xml_file)
        root = tree.getroot()

        for car in root.findall("car"):
            if car.find("license_plate").text == license_plate:
                if brand:
                    car.find("brand").text = brand
                if color:
                    car.find("color").text = color
                if owner:
                    car.find("owner").text = owner
                break

        tree.write(self.xml_file)

    def get_car_info(self, license_plate):
        """Получает информацию о автомобиле по гос. номеру."""
        tree = ET.parse(self.xml_file)
        root = tree.getroot()

        for car in root.findall("car"):
            if car.find("license_plate").text == license_plate:
                return {
                    "license_plate": car.find("license_plate").text,
                    "brand": car.find("brand").text,
                    "color": car.find("color").text,
                    "owner": car.find("owner").text,
                }
        return None

    def sort_cars(self, by):
        """Сортирует автомобили по указанному критерию."""
        tree = ET.parse(self.xml_file)
        root = tree.getroot()
        cars = [car for car in root.findall("car")]

        if by == "brand":
            cars.sort(key=lambda x: x.find("brand").text)
        elif by == "color":
            cars.sort(key=lambda x: x.find("color").text)
        elif by == "owner":
            cars.sort(key=lambda x: x.find("owner").text)

        return cars

    def get_cars_by_owner(self, owner):
        """Получает все автомобили, принадлежащие заданному владельцу."""
        tree = ET.parse(self.xml_file)
        root = tree.getroot()

        owner_cars = []
        for car in root.findall("car"):
            if car.find("owner").text == owner:
```

```
                owner_cars.append(
                    {
                        "license_plate": car.find("license_plate").text,
                        "brand": car.find("brand").text,
                        "color": car.find("color").text,
                        "owner": car.find("owner").text,
                    }
                )

        return owner_cars

    def display_all_cars(self):
        """Выводит информацию о всех автомобилях."""
        tree = ET.parse(self.xml_file)
        root = tree.getroot()

        for car in root.findall("car"):
            print(
                 f"{car.find('license_plate').text}, {car.find('brand').text},
"
                f"{car.find('color').text}, {car.find('owner').text}"
            )


class MenuItem:
    def __init__(self, title, action):
        self.title = title
        self.action = action


class Menu:
    def __init__(self):
        self.items = []

    def add_item(self, title, action):
        self.items.append(MenuItem(title, action))

    def display(self):
        print("\nМеню:")
        for index, item in enumerate(self.items):
            print(f"{index + 1}. {item.title}")
        print("0. Выход")

    def execute(self, choice):
        if 0 < choice <= len(self.items):
            self.items[choice - 1].action()
        elif choice == 0:
            print("Выход из программы.")
            return True
        else:
            print("Неверный выбор, попробуйте снова.")
        return False


def main():
```

```
    car_manager = CarManager("cars.xml")
    menu = Menu()

    # Добавляем пункты меню
    menu.add_item("Добавить новую запись", lambda: add_new_car(car_manager))
    menu.add_item("Удалить запись", lambda: remove_car(car_manager))
    menu.add_item("Изменить запись", lambda: update_car(car_manager))
            menu.add_item("Получить информацию о автомобиле", lambda: get_car_info(car_manager))
    menu.add_item("Отсортировать автомобили", lambda: sort_cars(car_manager))
    menu.add_item(
                          "Получить автомобили по владельцу", lambda: get_cars_by_owner(car_manager)
    )
                      menu.add_item("Вывести все автомобили", lambda: car_manager.display_all_cars())

    while True:
        menu.display()
        choice = int(input("Выберите действие: "))
        if menu.execute(choice):
            break


def add_new_car(car_manager):
    license_plate = input("Введите гос. номер: ")
    brand = input("Введите марку: ")
    color = input("Введите цвет: ")
    owner = input("Введите ФИО владельца: ")
    car_manager.add_car(license_plate, brand, color, owner)
    print("Запись добавлена.")


def remove_car(car_manager):
    license_plate = input("Введите гос. номер для удаления: ")
    car_manager.remove_car(license_plate)
    print("Запись удалена.")


def update_car(car_manager):
    license_plate = input("Введите гос. номер для изменения: ")
    brand = input("Введите новую марку (или оставьте пустым для пропуска): ")
    color = input("Введите новый цвет (или оставьте пустым для пропуска): ")
       owner = input("Введите новое ФИО владельца (или оставьте пустым для пропуска): ")
    car_manager.update_car(
        license_plate,
        brand if brand else None,
        color if color else None,
        owner if owner else None,
    )
    print("Запись изменена.")


def get_car_info(car_manager):
```

```
220         license_plate = input("Введите гос. номер: ")
221         car_info = car_manager.get_car_info(license_plate)
222         if car_info:
223             print(car_info)
224         else:
225             print("Автомобиль не найден.")
226 
227 
228     def sort_cars(car_manager):
229         sort_by = input("Сортировать по (brand/color/owner): ")
230         sorted_cars = car_manager.sort_cars(sort_by)
231         for car in sorted_cars:
232             print(
233                 f"{car.find('license_plate').text}, {car.find('brand').text}, "
234                 f"{car.find('color').text}, {car.find('owner').text}"
235             )
236 
237 
238     def get_cars_by_owner(car_manager):
239         owner = input("Введите ФИО владельца: ")
240         owner_cars = car_manager.get_cars_by_owner(owner)
241         if owner_cars:
242             for car in owner_cars:
243                 print(car)
244         else:
245             print("Автомобили не найдены.")
246 
247 
248     if __name__ == "__main__":
249         main()
```

Результат работы программы:

```
Меню:
1. Добавить новую запись
2. Удалить запись
3. Изменить запись
4. Получить информацию о автомобиле
5. Отсортировать автомобили
6. Получить автомобили по владельцу
7. Вывести все автомобили
0. Выход
Выберите действие: 7

A123BC, Toyota, Red, Ivanov Ivan
B456CD, BMW, Black, Petrov Petr
C789EF, Audi, Blue, Sidorov Sidor
```

Вывод: в ходе работы были приобретены практические навыки работы с файлами и файловой системой средствами языка Python.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Васильев А.Н. Python на примерах [Электронный ресурс]: практический курс по программированию/ Васильев А.Н.— Электрон. текстовые данные. — СПб.: Наука и Техника, 2017.— 432 с.— Режим доступа: http://www.iprbookshop.ru/73043.html
2. Буйначев С.К. Основы программирования на языке Python [Электронный ресурс]: учебное пособие/ Буйначев С.К., Боклаг Н.Ю.— Электрон. текстовые данные.— Екатеринбург: Уральский федеральный университет, ЭБС АСВ, 2014.— 92 с.— Режим доступа: http://www.iprbookshop.ru/66183.html