|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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: |

**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:**

1 import os

2 import zipfile

3

4

5 files\_starting\_with\_a = []

6 test\_dir = "test\_folder"

7 zip\_filename = "test\_folder.zip"

8

9 for root, dirs, files in os.walk(test\_dir):

10 for file in files:

11 if file.startswith("a"):

12 files\_starting\_with\_a.append(os.path.join(root, file))

13

14 with open("files\_starting\_with\_a.txt", "w") as f:

15 for file in files\_starting\_with\_a:

16 f.write(file + "\n")

17

18 with zipfile.ZipFile(zip\_filename, "w", zipfile.ZIP\_DEFLATED) as zipf:

19 for root, dirs, files in os.walk(test\_dir):

20 for file in files:

21 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

60 def update\_car(self, license\_plate, brand=None, color=None, owner=None):

61 """Изменяет запись о автомобиле по гос. номеру."""

62 tree = ET.parse(self.xml\_file)

63 root = tree.getroot()

64

65 for car in root.findall("car"):

66 if car.find("license\_plate").text == license\_plate:

67 if brand:

68 car.find("brand").text = brand

69 if color:

70 car.find("color").text = color

71 if owner:

72 car.find("owner").text = owner

73 break

74

75 tree.write(self.xml\_file)

76

77 def get\_car\_info(self, license\_plate):

78 """Получает информацию о автомобиле по гос. номеру."""

79 tree = ET.parse(self.xml\_file)

80 root = tree.getroot()

81

82 for car in root.findall("car"):

83 if car.find("license\_plate").text == license\_plate:

84 return {

85 "license\_plate": car.find("license\_plate").text,

86 "brand": car.find("brand").text,

87 "color": car.find("color").text,

88 "owner": car.find("owner").text,

89 }

90 return None

91

92 def sort\_cars(self, by):

93 """Сортирует автомобили по указанному критерию."""

94 tree = ET.parse(self.xml\_file)

95 root = tree.getroot()

96 cars = [car for car in root.findall("car")]

97

98 if by == "brand":

99 cars.sort(key=lambda x: x.find("brand").text)

100 elif by == "color":

101 cars.sort(key=lambda x: x.find("color").text)

102 elif by == "owner":

103 cars.sort(key=lambda x: x.find("owner").text)

104

105 return cars

106

107 def get\_cars\_by\_owner(self, owner):

108 """Получает все автомобили, принадлежащие заданному владельцу."""

109 tree = ET.parse(self.xml\_file)

110 root = tree.getroot()

111

112 owner\_cars = []

113 for car in root.findall("car"):

114 if car.find("owner").text == owner:

115 owner\_cars.append(

116 {

117 "license\_plate": car.find("license\_plate").text,

118 "brand": car.find("brand").text,

119 "color": car.find("color").text,

120 "owner": car.find("owner").text,

121 }

122 )

123

124 return owner\_cars

125

126 def display\_all\_cars(self):

127 """Выводит информацию о всех автомобилях."""

128 tree = ET.parse(self.xml\_file)

129 root = tree.getroot()

130

131 for car in root.findall("car"):

132 print(

133 f"{car.find('license\_plate').text}, {car.find('brand').text}, "

134 f"{car.find('color').text}, {car.find('owner').text}"

135 )

136

137

138 class MenuItem:

139 def \_\_init\_\_(self, title, action):

140 self.title = title

141 self.action = action

142

143

144 class Menu:

145 def \_\_init\_\_(self):

146 self.items = []

147

148 def add\_item(self, title, action):

149 self.items.append(MenuItem(title, action))

150

151 def display(self):

152 print("\nМеню:")

153 for index, item in enumerate(self.items):

154 print(f"{index + 1}. {item.title}")

155 print("0. Выход")

156

157 def execute(self, choice):

158 if 0 < choice <= len(self.items):

159 self.items[choice - 1].action()

160 elif choice == 0:

161 print("Выход из программы.")

162 return True

163 else:

164 print("Неверный выбор, попробуйте снова.")

165 return False

166

167

168 def main():

169 car\_manager = CarManager("cars.xml")

170 menu = Menu()

171

172 *# Добавляем пункты меню*

173 menu.add\_item("Добавить новую запись", lambda: add\_new\_car(car\_manager))

174 menu.add\_item("Удалить запись", lambda: remove\_car(car\_manager))

175 menu.add\_item("Изменить запись", lambda: update\_car(car\_manager))

176 menu.add\_item("Получить информацию о автомобиле", lambda: get\_car\_info(car\_manager))

177 menu.add\_item("Отсортировать автомобили", lambda: sort\_cars(car\_manager))

178 menu.add\_item(

179 "Получить автомобили по владельцу", lambda: get\_cars\_by\_owner(car\_manager)

180 )

181 menu.add\_item("Вывести все автомобили", lambda: car\_manager.display\_all\_cars())

182

183 while True:

184 menu.display()

185 choice = int(input("Выберите действие: "))

186 if menu.execute(choice):

187 break

188

189

190 def add\_new\_car(car\_manager):

191 license\_plate = input("Введите гос. номер: ")

192 brand = input("Введите марку: ")

193 color = input("Введите цвет: ")

194 owner = input("Введите ФИО владельца: ")

195 car\_manager.add\_car(license\_plate, brand, color, owner)

196 print("Запись добавлена.")

197

198

199 def remove\_car(car\_manager):

200 license\_plate = input("Введите гос. номер для удаления: ")

201 car\_manager.remove\_car(license\_plate)

202 print("Запись удалена.")

203

204

205 def update\_car(car\_manager):

206 license\_plate = input("Введите гос. номер для изменения: ")

207 brand = input("Введите новую марку (или оставьте пустым для пропуска): ")

208 color = input("Введите новый цвет (или оставьте пустым для пропуска): ")

209 owner = input("Введите новое ФИО владельца (или оставьте пустым для пропуска): ")

210 car\_manager.update\_car(

211 license\_plate,

212 brand if brand else None,

213 color if color else None,

214 owner if owner else None,

215 )

216 print("Запись изменена.")

217

218

219 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 c.— 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 c.— Режим доступа: http://www.iprbookshop.ru/66183.html