МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 1

на тему: «Основные типы данных. Управляющие конструкции»

по дисциплине: «Программирование на языке Python»

Вариант № 18

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверили: Захарова О.В., Раков В.И.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2019 г.

**Задание:**

1. Написать программу, вычисляющую выражение: . Значения всех переменных задавать с клавиатуры. При задании неподходящих данных программа должна показать сообщение об ошибке. Использовать модуль math и управляющие конструкции (не использовать lambda). x n nx x c b a sin 5  

2. Создать список c элементами разных типов. Функционал программы:

1) показать значения списка на экране;

2) добавление нового элементов в начало списка (добавлять элементы разных типов);

3) удаление первого элемента списка;

4) сформировать кортеж, состоящий из вещественных элементов списка больших 100; вывести содержимое кортежа на экран;

5) найти сумму всех целочисленных отрицательных элементов списка;

6) сформировать строку из значений элементов списка и посчитать количество цифр в строке;

7) задать с клавиатуры множество M1, сформировать множество M2 из списка; вывести на экран разницу множеств M2 и M1;

8) получить из списка словарь, ключом каждого элемента сделать позицию элемента в словаре; построчно отобразить на экране элементы словаря с четными значениями ключа.

Программа должна обеспечить возможность выполнять пункты 1-8 до тех пор, пока пользователь не решит из неё выйти (использовать оператор цикла).

3. Написать программу, вычисляющую площадь фигур. Меню программы имеет следующие пункты: «E» (площадь эллипса), «T» (площадь треугольника), «R» (площадь прямоугольника), «Q» (выход из программы). В случае ввода неверных данных выдать сообщение об ошибке. Для проверки условий использовать условный оператор. Программа должна обеспечить возможность вычислять площадь фигур до тех пор, пока пользователь не решит из неё выйти (использовать оператор цикла). Не использовать lambda.

**Код:**

**«main.py»**

import task1import task2import task3

while True:

print("Choose task:")

print("1 - 1") print("2 - 2") print("3 - 3") print("0 - exit")

task\_number = input("Your choose: ") print()

if task\_number == '1': task1.TaskOneMain()

elif task\_number == '2': task2.TaskTwoMain()

elif task\_number == '3': task3.TaskThreeMain()

elif task\_number == '0': break

else: print(task\_number + "\_Undefined command")

**«task1.py»**

import math

def calculateExpression(a, b, c, n, x):

res = 0.0

try:

res = ((5 \* math.pow(a, n \* x))

/ math.fabs(b + c)) \

- math.sqrt(math.fabs(math.sin(math.pow(x, n \* x))))

except ZeroDivisionError:

print("Incorrect values! Division by zero!")

return res

def TaskOneMain():

while True:

while True:

a = input("a = ")

try:

a = float(a)

except ValueError:

print("Its not float")

continue

break

while True:

b = input("b = ")

try:

b = float(b)

except ValueError:

print("Its not float")

continue

break

while True:

c = input("c = ")

try:

c = float(c)

except ValueError:

print("Its not float")

continue

break

while True:

n = input("n = ")

try:

n = float(n)

except ValueError:

print("Its not float")

continue

break

while True:

x = input("x = ")

try:

x = float(x)

except ValueError:

print("Its not float")

continue

break

break

print("Result: " + str(calculateExpression(a, b, c, n, x)))

input()

**«task2.py»**

def PrintList(l: list):

for v in l:

print(v)

def AddElement(l: list):

newValue = input("New value: ")

try:

l.insert(0, int(newValue))

return True

except ValueError:

pass

try:

l.insert(0, float(newValue))

return True

except ValueError:

pass

l.insert(0, newValue)

return True

def DeleteFirst(l: list):

l.pop(0)

def Task4(l: list):

t = []

for v in l:

if type(v) == float and v > 100.0:

t.append(v)

myTuple = tuple(t)

print(myTuple)

def SumIntNegative(l: list):

sum = 0

for v in l:

if type(v) == int and v < 0:

sum += v

print("Sum: " + str(sum))

def NumbersCount(l: list):

s = str(l)

count = [int(i) for i in s if i.isdigit()]

print("Count: " + str(len(count)))

def CreateM(l: list):

print("Enter set elements:")

custom\_set = set(map(str, input().split()))

print(str(custom\_set))

list\_set = set(map(str, l))

print(list\_set.difference(custom\_set))

def CreateDictionary(l: list):

myDict = {}

for i, v in enumerate(l):

myDict[i] = v

for i, v in myDict.items():

print(v) if i % 2 == 0 else 0

def TaskTwoMain():

custom\_list = [-1, 153.5, "Python", -14, 17, -5, 147.4, "Hello"]

while True:

print("1 - Show list")

print("2 - Add new element")

print("3 - Delete first element")

print("4 - Show element > 100")

print("5 - Sum all int negative element")

print("6 - Count of numbers in string of list")

print("7 - Create M1")

print("8 - Show dictionary")

print("0 - Exit")

command = input("Command:")

if command == '1':

PrintList(custom\_list)

elif command == '2':

AddElement(custom\_list)

elif command == '3':

DeleteFirst(custom\_list)

elif command == '4':

Task4(custom\_list)

elif command == '5':

SumIntNegative(custom\_list)

elif command == '6':

NumbersCount(custom\_list)

elif command == '7':

CreateM(custom\_list)

elif command == "8":

CreateDictionary(custom\_list)

elif command == '0':

return

else:

print("Incorrect command")

**«task3.py»**

import math

def Ellipse():

res = 0.0

while True:

a = input("Big semiaxis = ")

try:

a = float(a)

except ValueError:

print("Its not float")

continue

break

while True:

b = input("Little semiaxis = ")

try:

b = float(b)

except ValueError:

print("Its not float")

continue

break

res = math.pi \* a \* b

print("Res: " + str(res))

input()

pass

def Triangle():

res = 0.0

while True:

a = input("Basis of triangle = ")

try:

a = float(a)

except ValueError:

print("Its not float")

continue

break

while True:

h = input("Height = ")

try:

h = float(h)

except ValueError:

print("Its not float")

continue

break

res = 1/2 \* a \* h

print("Res: " + str(res))

input()

def Rectangle():

res = 0.0

while True:

a = input("a = ")

try:

a = float(a)

except ValueError:

print("Its not float")

continue

break

while True:

b = input("b = ")

try:

b = float(b)

except ValueError:

print("Its not float")

continue

break

res = a \* b

print("Area: " + str(res))

input()

def TaskThreeMain():

while True:

print("Ellipse - e")

print("Triangle - t")

print("Rectangle - r")

print("Quit - q")

command = input("Command:")

if command == 'e':

Ellipse()

elif command == 't':

Triangle()

elif command == 'r':

Rectangle()

elif command == 'q':

break

else:

print("Undefined command...")

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 2

на тему: «Функциональное программирование»

по дисциплине: «Программирование на языке Python»

Вариант № 18

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверили: Захарова О.В., Раков В.И.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2019 г.

**Задание:**

1. Написать программу, вычисляющую выражение (лабораторная работа № 1, вариант по списку студентов, задание 1). Значения всех переменных задавать с клавиатуры. При задании неверных данных выдать сообщение об ошибке. Использовать модуль math. При разработке программы не использовать управляющие конструкции. Использовать функции первого класса и высшего порядка.

2. Написать программу, вычисляющую площадь фигур. Функционал программы разработать в соответствии с первой лабораторной работой, вариантом по списку студентов, заданием 3 (например для первого варианта: вычисление площади прямоугольника («R»); вычисление площади прямоугольного треугольника («T»); вычисление площади многоугольника («M»), выход из программы («E»); в случае ввода неверных данных выдать сообщение об ошибке).

Входные данные задать в виде одного списка. Программа должна обеспечить возможность вычислять площадь фигур до тех пор, пока в списке есть входные данные. При разработке программы не использовать управляющие конструкции. Использовать функции первого класса и высшего порядка.

Пример списка с входными данными:

L = [ ['R', 'r', 'M', 'T', 'T', 'E'], [1, 2, 3, 4, 5, 6], <входные данные > ]

**Код:**

**«main.py»**

import task1

import task2

while True:

print("Choose task:")

print("1 - 1")

print("2 - 2")

print("0 - exit")

task\_number = input("Your choose: ")

print()

if task\_number == '1':

task1.TaskOneMain()

elif task\_number == '2':

task2.TaskTwoMain()

elif task\_number == '0':

break

else:

print(task\_number + "\_Undefined command")

**«task1.py»**

import math

def TaskOneMain():

while True:

while True:

a = input("a = ")

try:

a = float(a)

except ValueError:

print("Its not float")

continue

break

while True:

b = input("b = ")

try:

b = float(b)

except ValueError:

print("Its not float")

continue

break

while True:

c = input("c = ")

try:

c = float(c)

except ValueError:

print("Its not float")

continue

break

while True:

n = input("n = ")

try:

n = float(n)

except ValueError:

print("Its not float")

continue

break

while True:

x = input("x = ")

try:

x = float(x)

except ValueError:

print("Its not float")

continue

break

break

calculate = lambda a, b, c, n, x: ((5 \* math.pow(a, n \* x))

/ math.fabs(b + c)) \

- math.sqrt(math.fabs(math.sin(math.pow(x, n \* x))))

print("Result: " + str(calculate(a, b, c, n, x)))

input("Press any key...")

**«task2.py»**

import math

import re

def TaskTwoMain():

while True:

print("Ellipse - e (enter two values for ellipse)")

print("Triangle - t (enter two values for triangle)")

print("Rectangle - r ( enter two values for rectangle)")

print("Quit - q")

commands = list(filter(

lambda command: command.lower() in ['e', 't', 'r', 'q'],

[i for i in input().split()]

))

#

#

print("Enter a, b:")

values = list(filter(

lambda value: re.match("[0-9.0-9]", value),

[j for j in input().split()]

))

results = list\

(

map

(

lambda command:

(

command.lower() == 'e' and

(

float(values[0]) \* float(values[1]) \* math.pi

)

or command.lower() == 't' and

(

1/2 \* float(values[0]) \* float(values[1])

)

or command.lower() == 'r' and (float(values[0]) \* float(values[1]))

or command.lower() == 'q' and "exit"

),

commands

)

)

list(map

(

lambda command, result: print(f'{command}: {result}'),

commands,

results

)

)

list(map(

lambda result:

(

result == "exit" and (exit(0))

),

results

))

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 3

на тему: «Объектно-ориентированное программирование»

по дисциплине: «Программирование на языке Python»

Вариант № 18

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверили: Захарова О.В., Раков В.И.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2019 г.

**Задание:**

В программе разработать класс «Процессор» с обязательными атрибутами: артикул, фирма-производитель, название, количество ядер, тактовая частота, стоимость. Вводимую информацию о процессорах хранить в списке объектов класса «Процессор».

Меню программы: 1) добавление информации в список (разработать конструктор с произвольным количеством параметром); 2) удаление информации о выбранном объекте списка; 3) отображение информации обо всех объектах списка в удобном виде; 4) поиск информации о процессорах указанную пользователем фирмы-производителя.

Программа должна работать до тех пор (выполнять выбранные пользователем пункты меню программы), пока пользователь не решит из неё выйти.

В лабораторной работе не разрабатывать GUI.

**Код:**

**«main.py»**

from processor import Processor

def main():

processors = list()

while True:

print("1 - Add processor")

print("2 - Remove processor")

print("3 - Print all processors")

print("4 - Find processors by firm-producer")

print("0 - Exit")

command = input("Command: ")

if command == '1':

proc = Processor(article=input("Article: "),

producer=input("Producer: "),

name=input("Name: "),

cores=input("Cores count: "),

frequency=input("Frequency: "),

price=input("Price: ")

)

processors.append(proc)

elif command == '2':

article = input("Article to delete: ")

for proc in processors:

if proc.data.get('article') == article:

processors.remove(proc)

elif command == '3':

for v in processors:

print(v)

elif command == '4':

producer = input("Producer: ")

for proc in processors:

if proc.data.get('producer') == producer:

print(proc)

elif command == '0':

return

else:

print("Undefined command...Try again")

if \_\_name\_\_ == '\_\_main\_\_':

main()

**«processor.py»**

from typing import List

class Processor:

data = dict

def \_\_init\_\_(self, \*\*kwargs):

self.data = kwargs

def \_\_str\_\_(self):

return f"\nArticle: {self.data.get('article')}\n" \

f"Producer: {self.data.get('producer')}\n" \

f"Name: {self.data.get('name')}\n" \

f"Cores count: {self.data.get('cores')}\n" \

f"Frequency: {self.data.get('Frequency')}\n"\

f"Price: {self.data.get('price')}\n"

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 4

на тему: «Разработка графического интерфейса»

по дисциплине: «Программирование на языке Python»

Вариант № 18

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверили: Захарова О.В., Раков В.И.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2019 г.

**Задание:**

Разработать и реализовать графический интерфейс для задания лабораторной работы № 3 (вариант в соответствии со списком студентов).

Обязательные элементы графического интерфейса: надписи, кнопки, текстовые поля, выпадающий список, чекбоксы и/или радиокнопки, всплывающее окно с сообщением.

**Код:**

**«main.py»**

from PyQt5 import QtWidgets

from form import MainWindow

import sys

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication(sys.argv)

main\_window = MainWindow()

main\_window.show()

sys.exit(app.exec())

**«processor.py»**

class Processor:

data = dict

def \_\_init\_\_(self, \*\*kwargs):

self.data = kwargs

def \_\_str\_\_(self):

return f"\nArticle: {self.data.get('article')}\n" \

f"Producer: {self.data.get('producer')}\n" \

f"Name: {self.data.get('name')}\n" \

f"Cores count: {self.data.get('cores')}\n" \

f"Frequency: {self.data.get('Frequency')}\n"\

f"Price: {self.data.get('price')}\n"

@staticmethod

def attributes():

return ['article', 'producer', 'name',

'cores', 'frequency', 'price']

**«form.py»**

from typing import List

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QMessageBox

import lab4 as view

import processor as proc

global processors

processors = list()

class MainWindow(QtWidgets.QMainWindow, view.Ui\_MainWindow):

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

super().\_\_init\_\_()

self.setupUi(self)

self.init\_actions()

self.init\_list()

self.init\_values()

self.update\_table\_content()

def init\_actions(self):

# set actions on buttons

self.buttonAdd.clicked.connect(self.add\_processor)

self.buttonDelete.clicked.connect(self.delete\_processor)

self.buttonSearch.clicked.connect(self.search\_processor)

self.buttonPrintAll.clicked.connect(self.print\_all)

self.buttonPrintByCheckBox.clicked.connect(self.print\_by\_combo\_box)

def init\_values(self):

self.comboBoxCores.clear()

self.comboBoxCores.addItems(['None', '2', '4', '6'])

processors.append(proc.Processor(

article=1,

producer=1,

name=1,

cores=1,

frequency=1,

price=1

))

processors.append(proc.Processor(

article=2,

producer=2,

name=2,

cores=2,

frequency=2,

price=1

))

processors.append(proc.Processor(

article=3,

producer=3,

name=3,

cores=3,

frequency=3,

price=333

))

processors.append(proc.Processor(

article=4,

producer=4,

name=4,

cores=4,

frequency=4,

price=444

))

processors.append(proc.Processor(

article=5,

producer=5,

name=5,

cores=5,

frequency=5,

price=555

))

processors.append(proc.Processor(

article=6,

producer=6,

name=6,

cores=6,

frequency=6,

price=6

))

def add\_processor(self):

processor = proc.Processor(

article=self.editArticle.text(),

producer=self.editProducer.text(),

name=self.editName.text(),

cores=self.editCores.text(),

frequency=self.editFrequency.text(),

price=self.editPrice.text()

)

processors.append(processor)

self.clear\_add\_edits()

self.update\_table\_content()

def delete\_processor(self):

article = self.editDeleteArticle.text()

for processor in processors:

if str(processor.data.get('article')) == article:

processors.remove(processor)

self.editDeleteArticle.clear()

self.update\_table\_content()

def search\_processor(self):

producer = self.editSearch.text()

data = list()

for processor in processors:

if str(processor.data.get('producer')) == producer:

data.append(processor)

self.editSearch.clear()

if len(data) == 0:

msg\_box\_no = QtWidgets.QMessageBox.question(self, 'Empty', "There are no such processors", QMessageBox.Ok)

if msg\_box\_no == QMessageBox.Ok:

pass

self.update\_table\_content()

pass

else:

self.update\_table\_content(data)

def print\_all(self):

self.update\_table\_content()

def init\_list(self):

self.tableResult.setColumnCount(6)

self.tableResult.setRowCount(1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

def update\_table\_content(self, data: List[proc.Processor] = None):

if data is None:

data = processors.copy()

global msg\_box

if len(data) == 0:

msg\_box = QtWidgets.QMessageBox.question(self, 'Empty', "There are no processors", QMessageBox.Ok)

if msg\_box == QMessageBox.Ok:

pass

self.tableResult.clear()

rows = len(data)

self.tableResult.setRowCount(rows + 1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

if self.checkBoxPrice.isChecked():

check\_list = list()

for row, processor in enumerate(data):

if float(data[row].data['price']) > 100:

check\_list.append(processor)

if len(check\_list) == 0:

msg\_box = QtWidgets.QMessageBox.question(self, 'Empty', "There are no processors", QMessageBox.Ok)

if msg\_box == QMessageBox.Ok:

pass

else:

self.tableResult.clear()

rows = len(check\_list)

self.tableResult.setRowCount(rows + 1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

for row, processor in enumerate(check\_list):

for column, attribute in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

row + 1,

column,

QtWidgets.QTableWidgetItem(

str(check\_list[row].data[attribute])

)

)

else:

for row, processor in enumerate(data):

for column, attribute in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

row + 1,

column,

QtWidgets.QTableWidgetItem(

str(data[row].data[attribute])

)

)

def print\_by\_combo\_box(self):

index = self.comboBoxCores.currentIndex()

value = self.comboBoxCores.currentText()

if index == 0:

self.update\_table\_content()

elif index == 1:

self.print\_by\_cb\_index(value)

elif index == 2:

self.print\_by\_cb\_index(value)

elif index == 3:

self.print\_by\_cb\_index(value)

def print\_by\_cb\_index(self, value):

self.tableResult.clear()

cores\_list = list()

for row, processor in enumerate(processors):

if str(processors[row].data['cores']) == value:

if self.checkBoxPrice.isChecked():

if float(processors[row].data['price']) > 100:

cores\_list.append(processor)

else:

cores\_list.append(processor)

if len(cores\_list) == 0:

msg\_box\_cores = QtWidgets.QMessageBox.question(self, 'Empty', "There are no processors", QMessageBox.Ok)

if msg\_box\_cores == QMessageBox.Ok:

pass

rows = len(cores\_list)

self.tableResult.setRowCount(rows + 1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

for row, processor in enumerate(cores\_list):

for column, attribute in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

row + 1,

column,

QtWidgets.QTableWidgetItem(

str(cores\_list[row].data[attribute])

)

)

def clear\_add\_edits(self):

self.editArticle.clear()

self.editProducer.clear()

self.editName.clear()

self.editCores.clear()

self.editFrequency.clear()

self.editPrice.clear()

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 5

на тему: «Работа с файлами»

по дисциплине: «Программирование на языке Python»

Вариант № 18

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверили: Захарова О.В., Раков В.И.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2019 г.

**Задание:**

В программу лабораторной работы № 4 добавить главное меню с пунктами «Файл», «Справка».

Пункт главного меню «Файл» должен включать подпункты «Создать» (очистка формы для ввода новых данных), «Открыть» (считывание из файла информации об объектах и отображение её на форме), «Сохранить» (обновление текущего файла; если файл не создан, то действия пункта «Сохранить как…»), «Сохранить как…» (сохранение информации об объектах в файл по указанному пути), «Выход» (дружественный выход из программы).

Пункт главного меню «Справка» должен показывать информацию о разработчике программы.

**Код:**

**«main.py»**

from PyQt5 import QtWidgets

from form import MainWindow

import sys

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication(sys.argv)

main\_window = MainWindow()

main\_window.show()

sys.exit(app.exec())

**«processor.py»**

class Processor:

data = dict

def \_\_init\_\_(self, \*\*kwargs):

self.data = kwargs

def \_\_str\_\_(self):

return f"\nArticle: {self.data.get('article')}\n" \

f"Producer: {self.data.get('producer')}\n" \

f"Name: {self.data.get('name')}\n" \

f"Cores count: {self.data.get('cores')}\n" \

f"Frequency: {self.data.get('frequency')}\n"\

f"Price: {self.data.get('price')}\n"

@staticmethod

def attributes():

return ['article', 'producer', 'name',

'cores', 'frequency', 'price']

def \_\_dict\_\_(self):

return {

'article': self.data.get('article'),

'producer': self.data.get('producer'),

'name': self.data.get('name'),

'cores': self.data.get('cores'),

'frequency': self.data.get('frequency'),

'price': self.data.get('price'),

}

**«form.py»**

import pickle

import json

import os

from typing import List

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QMessageBox

import lab5 as view

import processor as proc

global processors

processors = list()

default\_path = "C:/Users/vscho/PycharmProjects/Python\_Lab5"

user\_path\_to\_save = "C:/Users/vscho/PycharmProject/Python\_Lab5"

class MainWindow(QtWidgets.QMainWindow, view.Ui\_MainWindow):

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

super().\_\_init\_\_()

self.setupUi(self)

self.init\_actions()

self.init\_table()

self.init\_values()

self.init\_menu()

self.update\_table\_content()

def init\_actions(self):

# set actions on buttons

self.buttonAdd.clicked.connect(self.add\_processor)

self.buttonDelete.clicked.connect(self.delete\_processor)

self.buttonSearch.clicked.connect(self.search\_processor)

self.buttonPrintAll.clicked.connect(self.print\_all)

self.buttonPrintByCheckBox.clicked.connect(self.print\_by\_combo\_box)

def init\_values(self):

self.comboBoxCores.clear()

self.comboBoxCores.addItems(['None', '2', '4', '6'])

processors.append(proc.Processor(

article=1,

producer=1,

name=1,

cores=1,

frequency=1,

price=1

))

processors.append(proc.Processor(

article=2,

producer=2,

name=2,

cores=2,

frequency=2,

price=1

))

processors.append(proc.Processor(

article=3,

producer=3,

name=3,

cores=3,

frequency=3,

price=333

))

processors.append(proc.Processor(

article=4,

producer=4,

name=4,

cores=4,

frequency=4,

price=444

))

processors.append(proc.Processor(

article=5,

producer=5,

name=5,

cores=5,

frequency=5,

price=555

))

processors.append(proc.Processor(

article=6,

producer=6,

name=6,

cores=6,

frequency=6,

price=6

))

def add\_processor(self):

processor = proc.Processor(

article=self.editArticle.text(),

producer=self.editProducer.text(),

name=self.editName.text(),

cores=self.editCores.text(),

frequency=self.editFrequency.text(),

price=self.editPrice.text()

)

processors.append(processor)

self.clear\_add\_edits()

self.update\_table\_content()

def delete\_processor(self):

article = self.editDeleteArticle.text()

for processor in processors:

if str(processor.data.get('article')) == str(article):

processors.remove(processor)

self.editDeleteArticle.clear()

self.update\_table\_content()

def search\_processor(self):

producer = self.editSearch.text()

data = list()

for processor in processors:

if str(processor.data.get('producer')) == producer:

data.append(processor)

self.editSearch.clear()

if len(data) == 0:

msg\_box\_no = QtWidgets.QMessageBox.question(self, 'Empty', "There are no such processors", QMessageBox.Ok)

if msg\_box\_no == QMessageBox.Ok:

pass

self.update\_table\_content()

pass

else:

self.update\_table\_content(data)

def print\_all(self):

self.update\_table\_content()

def init\_table(self):

self.tableResult.setColumnCount(6)

self.tableResult.setRowCount(1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

def update\_table\_content(self, data: List[proc.Processor] = None):

if data is None:

data = processors.copy()

global msg\_box

if len(data) == 0:

msg\_box = QtWidgets.QMessageBox.question(self, 'Empty', "There are no processors", QMessageBox.Ok)

if msg\_box == QMessageBox.Ok:

pass

self.tableResult.clear()

rows = len(data)

self.tableResult.setRowCount(rows + 1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

if self.checkBoxPrice.isChecked():

check\_list = list()

for row, processor in enumerate(data):

if float(data[row].data['price'] > 100):

check\_list.append(processor)

if len(check\_list) == 0:

msg\_box = QtWidgets.QMessageBox.question(self, 'Empty', "There are no processors", QMessageBox.Ok)

if msg\_box == QMessageBox.Ok:

pass

else:

self.tableResult.clear()

rows = len(check\_list)

self.tableResult.setRowCount(rows + 1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

for row, processor in enumerate(check\_list):

for column, attribute in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

row + 1,

column,

QtWidgets.QTableWidgetItem(

str(check\_list[row].data[attribute])

)

)

else:

for row, processor in enumerate(data):

for column, attribute in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

row + 1,

column,

QtWidgets.QTableWidgetItem(

str(data[row].data[attribute])

)

)

def print\_by\_combo\_box(self):

index = self.comboBoxCores.currentIndex()

value = self.comboBoxCores.currentText()

if index == 0:

self.update\_table\_content()

elif index == 1:

self.print\_by\_cb\_index(value)

elif index == 2:

self.print\_by\_cb\_index(value)

elif index == 3:

self.print\_by\_cb\_index(value)

def print\_by\_cb\_index(self, value):

self.tableResult.clear()

cores\_list = list()

for row, processor in enumerate(processors):

if str(processors[row].data['cores']) == value:

if self.checkBoxPrice.isChecked():

if float(processors[row].data['price']) > 100:

cores\_list.append(processor)

else:

cores\_list.append(processor)

if len(cores\_list) == 0:

msg\_box\_cores = QtWidgets.QMessageBox.question(self, 'Empty', "There are no processors", QMessageBox.Ok)

if msg\_box\_cores == QMessageBox.Ok:

pass

rows = len(cores\_list)

self.tableResult.setRowCount(rows + 1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

for row, processor in enumerate(cores\_list):

for column, attribute in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

row + 1,

column,

QtWidgets.QTableWidgetItem(

str(cores\_list[row].data[attribute])

)

)

def clear\_add\_edits(self):

self.editArticle.clear()

self.editProducer.clear()

self.editName.clear()

self.editCores.clear()

self.editFrequency.clear()

self.editPrice.clear()

def init\_menu(self):

menu = self.menuBar()

menu.setNativeMenuBar(False)

self.actionCreate.triggered.connect(self.on\_create)

self.actionOpen.triggered.connect(self.on\_open)

self.actionSave.triggered.connect(self.on\_save)

self.actionSave\_as.triggered.connect(self.on\_save\_as)

self.actionExit.triggered.connect(self.on\_exit)

self.actionAbout\_2.triggered.connect(self.on\_about)

def on\_create(self):

self.clear\_add\_edits()

self.editDeleteArticle.clear()

self.editSearch.clear()

processors.clear()

self.update\_table\_content()

def on\_open(self):

path = QtWidgets.QFileDialog.getOpenFileName(

self,

'Open file',

default\_path,

"\*.json"

)[0]

global user\_path\_to\_save

user\_path\_to\_save = path

data = []

if os.path.isfile(path):

with open(path, 'r') as f:

data = json.load(f)

global processors

processors = list()

for elem in data:

processor = proc.Processor(

article=elem['article'],

producer=elem['producer'],

name=elem['name'],

cores=elem['cores'],

frequency=elem['frequency'],

price=elem['price']

)

processors.append(processor)

self.update\_table\_content()

def on\_save(self):

print("UPTS: " + user\_path\_to\_save)

if os.path.exists(f'{user\_path\_to\_save}'):

data = []

for p in processors:

data.append(p.\_\_dict\_\_())

with open(f'{user\_path\_to\_save}', "w", encoding='utf-8') as fp:

json.dump(data, fp)

else:

self.on\_save\_as()

def on\_save\_as(self):

path = QtWidgets.QFileDialog.getSaveFileName(

self,

'Save file',

default\_path,

"\*.json"

)[0]

if path != '':

global user\_path\_to\_save

user\_path\_to\_save = path

data = []

for p in processors:

data.append(p.\_\_dict\_\_())

with open(user\_path\_to\_save, "w", encoding='utf-8') as fp:

json.dump(data, fp)

def on\_about(self):

about = QtWidgets.QMessageBox.question(self, 'Info', "Vladislav Shorin", QMessageBox.Ok)

if about == QMessageBox.Ok:

pass

def on\_exit(self):

exit = QtWidgets.QMessageBox.question(self, 'Exit', "You sure?", QMessageBox.Ok, QMessageBox.Abort)

if exit == QMessageBox.Ok:

QtWidgets.qApp.exit(0)

elif exit == QMessageBox.Abort:

pass

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 6

на тему: «Сетевое программирование»

по дисциплине: «Программирование на языке Python»

Вариант № 18

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверили: Захарова О.В., Раков В.И.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2019 г.

**Задание:**

Разработать клиент-серверное приложение (чат).

Требования к клиенту:

- отправка на сервер введенного пользователем сообщения;

- получение сообщений, отправленных другими клиентами;

- удобный графический интерфейс.

Требования к серверу:

- организация чата между клиентами (до 3 клиентов), а именно отправка всем клиентам полученных сообщений;

- полученные сообщения перед отправкой клиентам необходимо проверить на грамотность и исправить: после запятой должен стоять один пробел, далее должен идти непробельный символ.

Клиенты должны последовательно отправлять сообщения.

**Код:**

**«main.py» Client**

import tkinter

from socket import AF\_INET, socket, SOCK\_STREAM

from threading import Thread

def receive():

""" Handles receiving of messages. """

while True:

try:

msg = sock.recv(BUFSIZ).decode("utf8")

msg\_list.insert(tkinter.END, msg)

except OSError: # Possibly client has left the chat.

break

def send(event=None):

""" Handles sending of messages. """

msg = my\_msg.get()

my\_msg.set("") # Clears input field.

sock.send(bytes(msg, "utf8"))

if msg == "#quit":

sock.close()

top.quit()

def on\_closing(event=None):

""" This function is to be called when the window is closed. """

my\_msg.set("#quit")

send()

top = tkinter.Tk()

top.title("Simple Chat Client v1.0")

messages\_frame = tkinter.Frame(top)

my\_msg = tkinter.StringVar() # For the messages to be sent.

my\_msg.set("")

scrollbar = tkinter.Scrollbar(messages\_frame) # To navigate through past messages.

msg\_list = tkinter.Listbox(messages\_frame, height=15, width=70, yscrollcommand=scrollbar.set)

scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)

msg\_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)

msg\_list.pack()

messages\_frame.pack()

button\_label = tkinter.Label(top, text="Enter Message:")

button\_label.pack()

entry\_field = tkinter.Entry(top, textvariable=my\_msg, foreground="Red")

entry\_field.bind("<Return>", send)

entry\_field.pack()

send\_button = tkinter.Button(top, text="Send", command=send)

send\_button.pack()

quit\_button = tkinter.Button(top, text="Quit", command=on\_closing)

quit\_button.pack()

top.protocol("WM\_DELETE\_WINDOW", on\_closing)

HOST = "127.0.0.1"

PORT = 5000

BUFSIZ = 1024

ADDR = (HOST, PORT)

sock = socket(AF\_INET, SOCK\_STREAM)

sock.connect(ADDR)

receive\_thread = Thread(target=receive)

receive\_thread.start()

tkinter.mainloop() # Starts GUI execution.

**«processor.py» Server**

from socket import AF\_INET, socket, SOCK\_STREAM

from threading import Thread

import re

clients = {}

addresses = {}

HOST = "127.0.0.1"

PORT = 5000

BUFSIZ = 1024

ADDR = (HOST, PORT)

SOCK = socket(AF\_INET, SOCK\_STREAM)

SOCK.bind(ADDR)

def parse\_message(msg):

message = msg.decode()

# pattern\_whirespace = '[\s]+'

# re.sub(pattern\_whirespace, ' ', message)

i = 0

while i != len(message) - 3:

if message[i] == ',' \

and message[i + 1] != ' ':

message = message[:i] + ', ' + message[i + 1:]

i = 0

elif message[i] == ',' \

and message[i + 1] == ' ' \

and message[i + 2] == ' ':

message = message[:i] + ', ' + message[i + 3:]

i = 0

else:

message = message

i += 1

return message.encode()

def accept\_incoming\_connections():

"""Sets up handling for incoming clients."""

while True:

client, client\_address = SOCK.accept()

print(f"{client\_address} has connected.")

client.send("Greetings from the ChatRoom! ".encode("utf8"))

client.send("Now type your name and press enter!".encode("utf8"))

addresses[client] = client\_address

Thread(target=handle\_client, args=(client, client\_address)).start()

def handle\_client(conn, addr): # Takes client socket as argument.

"""Handles a single client connection."""

name = conn.recv(BUFSIZ).decode("utf8")

welcome = f'Welcome {name}! If you ever want to quit, type #quit to exit.'

conn.send(bytes(welcome, "utf8"))

msg = f"{name} from [{addr[0]}:{addr[1]}] has joined the chat!"

broadcast(bytes(msg, "utf8"))

clients[conn] = name

while True:

msg = conn.recv(BUFSIZ)

msg = parse\_message(msg)

if msg != bytes("#quit", "utf8"):

broadcast(msg, name + ": ")

else:

conn.send(bytes("#quit", "utf8"))

conn.close()

del clients[conn]

broadcast(bytes(f"{name} has left the chat.", "utf8"))

break

def broadcast(msg, prefix=""): # prefix is for name identification.

"""Broadcasts a message to all the clients."""

for sock in clients:

sock.send(bytes(prefix, "utf8") + msg)

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

SOCK.listen(3) # Listens for 3 connections at max.

print("Chat Server has Started !!")

print("Waiting for connections...")

ACCEPT\_THREAD = Thread(target=accept\_incoming\_connections)

ACCEPT\_THREAD.start() # Starts the infinite loop.

ACCEPT\_THREAD.join()

SOCK.close()

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 7

на тему: «Разработка приложения, взаимодействующего с базой данных»

по дисциплине: «Программирование на языке Python»

Вариант № 18

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверили: Захарова О.В., Раков В.И.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2019 г.

**Задание:**

Разработать приложение (можно web-приложение), взаимодействующее с базой данных. Приложение должно иметь удобный графический интерфейс. Базу данных разработать в соответствии с темой своего варианта (варианты представлены ниже). База данных должна состоять из 2-3 связанных таблиц; одна таблица основная.

Функционал приложения:

- добавление информации в основную таблица;

- удалении информации из основной таблицы;

- отображение информации из основной таблицы.

Добавление и отображение информации должно быть реализовано в читаемой для пользователя форме (внешние ключи не отображать, вместо них отображать пользователю понятную информацию).

18. База данных «Процессоры».

**Код:**

**«main.py»**

from PyQt5 import QtWidgets

from form import MainWindow

import sys

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication(sys.argv)

main\_window = MainWindow()

main\_window.show()

sys.exit(app.exec())

**«processor.py»**

class Processor:

data = dict

def \_\_init\_\_(self, \*\*kwargs):

self.data = kwargs

def \_\_str\_\_(self):

return f"\nArticle: {self.data.get('article')}\n" \

f"Producer: {self.data.get('producer')}\n" \

f"Name: {self.data.get('name')}\n" \

f"Cores count: {self.data.get('cores')}\n" \

f"Frequency: {self.data.get('frequency')}\n"\

f"Price: {self.data.get('price')}\n"

@staticmethod

def attributes():

return ['article', 'producer', 'name',

'cores', 'frequency', 'price']

def \_\_dict\_\_(self):

return {

'article': self.data.get('article'),

'producer': self.data.get('producer'),

'name': self.data.get('name'),

'cores': self.data.get('cores'),

'frequency': self.data.get('frequency'),

'price': self.data.get('price'),

}

**«form.py»**

import json

import os

import pymysql

from pymysql.cursors import DictCursor

from typing import List

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QMessageBox

import lab7 as view

import processor as proc

global processors

processors = list()

global cores\_table

cores\_table = dict

default\_path = "C:/Users/vscho/PycharmProjects/Python\_Lab5"

user\_path\_to\_save = "C:/Users/vscho/PycharmProject/Python\_Lab5"

class MainWindow(QtWidgets.QMainWindow, view.Ui\_MainWindow):

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

super().\_\_init\_\_()

self.setupUi(self)

self.init\_actions()

self.init\_table()

self.init\_values()

self.init\_menu()

self.init\_db()

self.load\_processors()

self.update\_table\_content()

def init\_actions(self):

# set actions on buttons

self.buttonAdd.clicked.connect(self.add\_processor)

self.buttonDelete.clicked.connect(self.delete\_processor)

self.buttonSearch.clicked.connect(self.search\_processor)

self.buttonPrintAll.clicked.connect(self.print\_all)

self.buttonPrintByCheckBox.clicked.connect(self.print\_by\_combo\_box)

def init\_db(self):

self.connection = pymysql.connect(

host='localhost',

user='root',

password='1234',

db='processors',

charset='utf8mb4',

cursorclass=DictCursor

)

self.cursor = self.connection.cursor()

def load\_processors(self):

query = """

SELECT \*

FROM cores

"""

self.cursor.execute(query)

global cores\_table

cores\_table = self.cursor.fetchall()

query = """

SELECT \*

FROM processor

"""

self.cursor.execute(query)

global processors

processors = list()

for row in self.cursor:

for elem in cores\_table:

if elem['idCores'] == row['cores\_idCores']:

cores = elem['value']

processor = proc.Processor(

article=row['article'],

producer=row['producer'],

name=row['name'],

cores=cores,

frequency=row['frequency'],

price=row['price']

)

processors.append(processor)

self.update\_table\_content()

def init\_values(self):

self.comboBoxCores.clear()

self.comboBoxCores.addItems(['None', '2', '4', '6'])

processors.append(proc.Processor(

article=1,

producer=1,

name=1,

cores=1,

frequency=1,

price=1

))

processors.append(proc.Processor(

article=2,

producer=2,

name=2,

cores=2,

frequency=2,

price=1

))

processors.append(proc.Processor(

article=3,

producer=3,

name=3,

cores=3,

frequency=3,

price=333

))

processors.append(proc.Processor(

article=4,

producer=4,

name=4,

cores=4,

frequency=4,

price=444

))

processors.append(proc.Processor(

article=5,

producer=5,

name=5,

cores=5,

frequency=5,

price=555

))

processors.append(proc.Processor(

article=6,

producer=6,

name=6,

cores=6,

frequency=6,

price=6

))

def add\_processor(self):

processor = proc.Processor(

article=self.editArticle.text(),

producer=self.editProducer.text(),

name=self.editName.text(),

cores=self.editCores.text(),

frequency=self.editFrequency.text(),

price=self.editPrice.text()

)

processors.append(processor)

cores = self.editCores.text()

self.cursor.execute(

f"select idCores from cores where idCores = {cores}"

)

cores = self.cursor.fetchall()

query = f"insert into processors.processor("\

f"article, producer, name, "\

f"cores\_idCores, frequency, price)"\

f"value ('{self.editArticle.text()}', '{self.editProducer.text()}',"\

f"{self.editName.text()}, {cores[0]['idCores']},"\

f"{self.editFrequency.text()}, {self.editPrice.text()})"

self.cursor.execute(query)

self.connection.commit()

self.clear\_add\_edits()

self.update\_table\_content()

def delete\_processor(self):

article = self.editDeleteArticle.text()

for processor in processors:

if processor.data.get('article') == article:

processors.remove(processor)

self.cursor.execute(f'delete from processors.processor where article = {article}')

self.connection.commit()

self.editDeleteArticle.clear()

self.update\_table\_content()

def search\_processor(self):

producer = self.editSearch.text()

data = list()

for processor in processors:

if str(processor.data.get('producer')) == producer:

data.append(processor)

self.editSearch.clear()

if len(data) == 0:

msg\_box\_no = QtWidgets.QMessageBox.question(self, 'Empty', "There are no such processors", QMessageBox.Ok)

if msg\_box\_no == QMessageBox.Ok:

pass

self.update\_table\_content()

pass

else:

self.update\_table\_content(data)

def print\_all(self):

self.update\_table\_content()

def init\_table(self):

self.tableResult.setColumnCount(6)

self.tableResult.setRowCount(1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

def update\_table\_content(self, data: List[proc.Processor] = None):

if data is None:

data = processors.copy()

global msg\_box

if len(data) == 0:

msg\_box = QtWidgets.QMessageBox.question(self, 'Empty', "There are no processors", QMessageBox.Ok)

if msg\_box == QMessageBox.Ok:

pass

self.tableResult.clear()

rows = len(data)

self.tableResult.setRowCount(rows + 1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

if self.checkBoxPrice.isChecked():

check\_list = list()

for row, processor in enumerate(data):

if float(data[row].data['price'] > 100):

check\_list.append(processor)

if len(check\_list) == 0:

msg\_box = QtWidgets.QMessageBox.question(self, 'Empty', "There are no processors", QMessageBox.Ok)

if msg\_box == QMessageBox.Ok:

pass

else:

self.tableResult.clear()

rows = len(check\_list)

self.tableResult.setRowCount(rows + 1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

for row, processor in enumerate(check\_list):

for column, attribute in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

row + 1,

column,

QtWidgets.QTableWidgetItem(

str(check\_list[row].data[attribute])

)

)

else:

for row, processor in enumerate(data):

for column, attribute in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

row + 1,

column,

QtWidgets.QTableWidgetItem(

str(data[row].data[attribute])

)

)

def print\_by\_combo\_box(self):

index = self.comboBoxCores.currentIndex()

value = self.comboBoxCores.currentText()

if index == 0:

self.update\_table\_content()

elif index == 1:

self.print\_by\_cb\_index(value)

elif index == 2:

self.print\_by\_cb\_index(value)

elif index == 3:

self.print\_by\_cb\_index(value)

def print\_by\_cb\_index(self, value):

self.tableResult.clear()

cores\_list = list()

for row, processor in enumerate(processors):

if str(processors[row].data['cores']) == value:

if self.checkBoxPrice.isChecked():

if float(processors[row].data['price']) > 100:

cores\_list.append(processor)

else:

cores\_list.append(processor)

if len(cores\_list) == 0:

msg\_box\_cores = QtWidgets.QMessageBox.question(self, 'Empty', "There are no processors", QMessageBox.Ok)

if msg\_box\_cores == QMessageBox.Ok:

pass

rows = len(cores\_list)

self.tableResult.setRowCount(rows + 1)

for i, attr in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

0, i, QtWidgets.QTableWidgetItem(attr)

)

for row, processor in enumerate(cores\_list):

for column, attribute in enumerate(proc.Processor.attributes()):

self.tableResult.setItem(

row + 1,

column,

QtWidgets.QTableWidgetItem(

str(cores\_list[row].data[attribute])

)

)

def clear\_add\_edits(self):

self.editArticle.clear()

self.editProducer.clear()

self.editName.clear()

self.editCores.clear()

self.editFrequency.clear()

self.editPrice.clear()

def init\_menu(self):

menu = self.menuBar()

menu.setNativeMenuBar(False)

self.actionCreate.triggered.connect(self.on\_create)

self.actionOpen.triggered.connect(self.on\_open)

self.actionSave.triggered.connect(self.on\_save)

self.actionSave\_as.triggered.connect(self.on\_save\_as)

self.actionExit.triggered.connect(QtWidgets.qApp.exit)

self.actionAbout.triggered.connect(self.on\_about)

def on\_create(self):

self.clear\_add\_edits()

self.editDeleteArticle.clear()

self.editSearch.clear()

processors.clear()

self.update\_table\_content()

def on\_open(self):

path = QtWidgets.QFileDialog.getOpenFileName(

self,

'Open file',

default\_path,

"\*.json"

)[0]

if os.path.isfile(path):

with open(path, 'r') as f:

data = json.load(f)

global processors

processors = list()

for elem in data:

processor = proc.Processor(

article=elem['article'],

producer=elem['producer'],

name=elem['name'],

cores=elem['cores'],

frequency=elem['frequency'],

price=elem['price']

)

processors.append(processor)

self.update\_table\_content()

def on\_save(self):

if os.path.exists(f'{user\_path\_to\_save}/db.json'):

data = []

for p in processors:

data.append(p.\_\_dict\_\_())

with open(f'{user\_path\_to\_save}/db.json', "w", encoding='utf-8') as fp:

json.dump(data, fp)

else:

self.on\_save\_as()

def on\_save\_as(self):

path = QtWidgets.QFileDialog.getSaveFileName(

self,

'Save file',

default\_path,

"\*.json"

)[0]

if path != '':

global user\_path\_to\_save

user\_path\_to\_save = path

data = []

for p in processors:

data.append(p.\_\_dict\_\_())

with open(user\_path\_to\_save, "w", encoding='utf-8') as fp:

json.dump(data, fp)

def on\_about(self):

about = QtWidgets.QMessageBox.question(self, 'Info', "Vladislav Shorin", QMessageBox.Ok)

if about == QMessageBox.Ok:

pass