МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем и цифровых технологий

ОТЧЕТ

по лабораторным работам

на дисциплине: «Программирование на языке Python»

Вариант 7

Студент: Жозеф В.

Шифр 190362

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 Программная инженерия

Группа 92ПГ

Проверил: Захарова О.В.

Орел 2021

Лабораторная работа №1
«Основные типы данных. Управляющие конструкции»

Задание:

1. Написать программу, вычисляющую выражение: $\sqrt[5]{\sin(x^n + \sqrt[n]{y}) + \sqrt[3]{\frac{e^{x^4}}{\cos y}}}$. Значения всех переменных задавать с клавиатуры. При вводе неподходящих данных программа должна показать сообщение об ошибке. Использовать модуль math и управляющие конструкции (не использовать lambda).

2. Создать список с элементами разных типов. Функционал программы:
1) показать значения списка на экране;
2) добавление нового элементов в конец списка (добавлять элементы разных типов);
3) удаление указанного пользователем элемента из списка;
4) сформировать кортеж, состоящий из элементов, стоящих на нечетных позициях списка; вывести содержимое кортежа на экран;
5) найти произведение всех вещественных элементов списка;
6) сформировать строку из значений элементов списка и посчитать количество арифметических операций в строке;
7) задать с клавиатуры множество M1, сформировать множество M2 из списка; вывести на экран множество, полученное путем пересечения множеств M1 и M2;
8) получить из списка словарь, ключом каждого элемента сделать позицию элемента в словаре; построчно отобразить на экране элементы словаря с ключом меньше 3.
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: «R» (площадь прямоугольного треугольника), «T» (площадь трапеции), «S» (площадь квадрата), «E» (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.

Решение:

1)

```
import math


print("Enter the x value: ")
x = int(input())
print("Enter the n value: ")
n = int(input())
while n<= 0:
    print("n cannot be negative or 0\n Enter it again")
    n=int(input())

print("Enter the y value: ")
y = int(input())
```

```
result = (math.sin(math.pow(x, n)+y**(1/n)) + ((math.exp(x**4)/math.cos(y))**(1/3)))**(1/5)
print(result)
```

2)

```
listDone = ["wen", '+', 12.5, 2, 1, '-', 16, "ok"]
print("Choose an option\n 1. exo1\n 2.exo2\n 3.exo3\n 4.exo4\n 5.exo5\n 6.exo6\n 7.exo7\n
8.exo8\n 9.quit")
option = int(input())
while option !=9:
    if option ==1:
        print("The list: ", listDone)
    elif option ==2:
        print("Add an element! \nEnter the element: ")
        newEl = input()
        listDone.append(newEl)
        print("The list: ", listDone)
    elif option ==3:
        print("Enter the value you want to delete: ")
        removeEl = input()
        listDone.remove(removeEl)
        print("The list with remove element: ", listDone)
    elif option == 4:
        cortege = (listDone[1::2])
        print("The cortege", cortege)
    elif option ==5:
     prodList = 1
     for i in range(len(listDone)):
         if type(listDone[i]) == int or type(listDone[i]) == float:
            prodList *= listDone[i]

     print("Le produit est: ", prodList)

    elif option ==6:
        stringMake = ' '.join([str(item) for item in listDone])
        print(stringMake)
        print("The number of operator in the string: ", len([i for i in stringMake if i in '+-:*=']))
    elif option ==7:
        print("Enter the set")
        M1 = set(map(int, input()))
        print(M1)
        M2= set(listDone)
        print(M2)
        M3= M1 &M2
        print("The intersection between the set: ", M3)
    elif option ==8:
        my_dict = dict(zip(range(len(listDone)), listDone))
        print("Mon dictionnaire \n", my_dict)
        for i in range(3):
         print(my_dict[i])
    else:
```

```
        print("Enter a valid command")
    print("Choose an option\n 1. exo1\n 2.exo2\n 3.exo3\n 4.exo4\n 5.exo5\n 6.exo6\n 7.exo7\n
8.exo8\n 9.quit")
    option = int(input())
```

3)

```
print("Choose an option\n R pour triangle rectangle\n T pour trapeze\n S pour carre\n E quitter")
option = input()
while option != 'E':
    if option == 'R':
        print("Enter the value of a: ")
        A = int(input())
        print("Enter the value of b: ")
        B = int(input())
        print("The surface is: ", (A * B) / 2)
    elif option == 'T':
        print("Entrer la grande base: ")
        B = int(input())
        print("Entrer la petite base: ")
        b = int(input())
        print("Entrer la hauteur: ")
        h = int(input())
        print("la surface du trapeze: ", ((B + b) * h) / 2)
    elif option == 'S':
        print("Entrer le cote du carre")
        C = int(input())
        print("La surface du carre: ", C * C)
    else:
        print("Enter a valid command")
    print("Choose an option\n R pour triangle rectangle\n T pour trapeze\n S pour carre\n E
quitter")
    option = input()
```

Лабораторная работа №2
«Функциональное программирование»

Задание:

1. Написать программу, вычисляющую выражение (лабораторная работа № 1, задание 1). Значения всех переменных задавать с клавиатуры. При вводе 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 3ем заданием первой лабораторной работы (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 >]

Решение:
1)
```
import math


def Operation():
    print("Enter the x value: ")
    x = int(input())
    print("Enter the n value: ")
    n = int(input())
    while n <= 0:
        print("n cannot be negative or 0\n Enter it again")
        n = int(input())

    print("Enter the y value: ")
    y = int(input())

    result = (math.sin(math.pow(x, n) + y ** (1 / n)) + ((math.exp(x ** 4) / math.cos(y)) ** (1 /
3))) ** (1 / 5)
    return result


print("Le resultat est: ", Operation())
```

2)
```
def surface(option):

    if option == 'R' or option== 'r':
        print("Enter the value of a: ")
        a = int(input())
```

```
        print("Enter the value of b: ")
        b = int(input())
        print("The surface is triangle is: ", (a * b) / 2)
    elif option == 'T' or option== 't':
        print("Entrer la grande base: ")
        b = int(input())
        print("Entrer la petite base: ")
        b1 = int(input())
        print("Entrer la hauteur: ")
        h = int(input())
        print("la surface du trapeze: ", ((b + b1) * h) / 2)
    elif option == 'S' or option == 's':
        print("Entrer le cote du carre")
        c = int(input())
        print("La surface du carre: ", c * c)
    elif option == 'E' or option == 'e':

        return
    else:
        print("There is no action for this command")


print("Choose the options\n R pour triangle rectangle\n T pour trapeze\n S pour carre\n E
quitter")
GivenList = input().split()
resultat = list(map(surface, GivenList))
```

Лабораторная работа №3
«Объектно-ориентированное программирование»

Задание:

В программе разработать класс «Лекарство» с обязательными атрибутами: артикул, название, стоимость, наличие рецепта, описание. Вводимую информацию о лекарствах хранить в списке объектов класса «Лекарство».
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 лекарств, продающихся без рецепта.
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.

Решение:

```
class medicament:
    code = 1
    name = 'paracetamol'
    cost = 1000
    receipt = True
    description = 'braise, amoxiciline'

    def __init__(self, code1, name1, cost1, receipt1, description1):
        self.code = code1
        self.name = name1
        self.cost = cost1
        self.receipt = receipt1
        self.description = description1

    def show(self):
        print(self.code, self.name, self.cost, self.receipt, self.description)

    def Get_code(self):
        return self.code

    def Get_receipt(self):
        return self.receipt

medList = []
print("1.Add information in the list\n"
      "2.Delete information \n"
      "3.Show information about all the medicine\n"
      "4.Search for all the medicine that can be sell without receipt\n"
      "Enter your choice")
c = int(input())

while c != 0:
```

```
if c == 1:
    print("Enter the code")
    cod = int(input())
    for i in range(len(medList)):
        if cod == medList[i].Get_code():
            print("This code already exist! Choose another one")
            cod = int(input())

    print("Enter the name")
    nom = input()
    print("Enter the cost")
    prix = int(input())
    print("Enter T if the medicine can be sell without receipt and F if not ")
    verif = False
    rec = input()
    while not verif:
        if rec == 'T' or rec == 't':
            presc = True
            verif = True
        elif rec == 'F' or rec == 'f':
            presc = False
            verif = True
        else:
            print("Wrong input. Enter T or F")
            rec = input()
    print("Enter the description")
    descr = input()
    med = medicament(cod, nom, prix, presc, descr)
    medList.append(med)

elif c == 2:
    print("Enter the code of the medicine you want to delete")
    askDel = int(input())
    for i in range(len(medList)):
        if askDel == medList[i].Get_code():
            del medList[i]
            break
        else:
            print("This object doesn't exist")

elif c == 3:
    for i in range(len(medList)):
        medList[i].show()
        print(medList[i].code)
    print("")
elif c == 4:
    print("The list of medicine that can be sell without receipt")
    for i in range(len(medList)):
        if not medList[i].Get_receipt():
            medList[i].show()
    print("")
```

```
print("1.Add information in the list\n"
      "2.Delete information \n"
      "3.Show information about all the medicine\n"
      "4.Search for all the medicine that can be sell without receipt\n"
      "Enter your choice")
c = int(input())
```

Лабораторная работа №4
«Разработка графического интерфейса»

Задание:

Разработать и реализовать графический интерфейс для задания лабораторной работы № 3.
Обязательные элементы графического интерфейса: надписи, кнопки, текстовые поля, выпадающий список, чекбоксы и/или радиокнопки, всплывающее окно с сообщением.

Решение:

```
from tkinter import *
from tkinter import messagebox
from tkinter.ttk import Radiobutton
from tkinter.ttk import Combobox


class Medicament:
    code = 1
    name = 'paracetamol'
    cost = 1000
    receipt = True
    description = 'braise, amoxiciline'

    def __init__(self, code1, name1, cost1, receipt1, description1):
        self.code = code1
        self.name = name1
        self.cost = cost1
        self.receipt = receipt1
        self.description = description1

    def show(self):
        print(self.code, self.name, self.cost, self.receipt, self.description)

    def Get_code(self):
        return self.code

    def Get_name(self):
        return self.name

    def Get_cost(self):
        return self.cost

    def Get_description(self):
        return self.description

    def Get_receipt(self):
        return self.receipt
```

```
def click():
    prix = 0
    cod = int(comb1.get())
    for i in range(len(medList)):
        if cod == medList[i].Get_code():
            messagebox.showerror('Error input', 'This code already exist! Choose another one')
            cod = 0
    nom = ten2.get()
    try:
        prix = int(ten3.get())
    except ValueError:
        try:
            prix = float(ten3.get())
        except ValueError:
            messagebox.showerror('Error input', 'The price can be an integer or a float')

    descr = ten4.get()
    recp = selected.get()
    if not cod or not nom or not prix or not descr:
        messagebox.showerror('Title', 'Fill all the blank space')
    else:
        med = Medicament(cod, nom, prix, recp, descr)
        medList.append(med)


def DeList():
    verif.set(False)
    try:
        delCod = int(ten5.get())
    except ValueError:
        messagebox.showerror('Error input', 'The code should be an integer')
        delCod = 0
    if not delCod:
        messagebox.showerror('Error', 'Fill the blank place')
    else:
        for i in range(len(medList)):
            if delCod == medList[i].code:
                del medList[i]
                verif.set(True)
                break
    v=verif.get()
    if not v:
        messagebox.showerror('Error input', 'This code doesnt exit')
    else:
        messagebox.showinfo('Success', 'The code is successfully delete! Press Show again to see
the change')


def show():
    global e1, e2, e3, e4, e5
    v = verif.get()
```

```
    r=len(medList)
    if v :
        e1.grid(row=r+14, column=0, sticky=NSEW)
        e1.delete(0, END)
        e2.grid(row=r+14, column=1, sticky=NSEW)
        e2.delete(0, END)
        e3.grid(row=r+14, column=2, sticky=NSEW)
        e3.delete(0, END)
        e4.grid(row=r+14, column=3, sticky=NSEW)
        e4.delete(0, END)
        e5.grid(row=r+14, column=4, sticky=NSEW)
        e5.delete(0, END)
        verif.set(False)
    rows, cols = (len(medList), 5)
    arr = [[0] * cols] * rows
    r = 14
    for i in range(len(medList)):
        arr[i][0] = medList[i].code
        e1 = Entry(relief=GROOVE)
        e1.grid(row=r, column=0, sticky=NSEW)
        e1.insert(END, '%s' % arr[i][0])
        arr[i][1] = medList[i].name
        e2 = Entry(relief=GROOVE)
        e2.grid(row=r, column=1, sticky=NSEW)
        e2.insert(END, '%s' % arr[i][1])
        arr[i][2] = medList[i].cost
        e3 = Entry(relief=GROOVE)
        e3.grid(row=r, column=2, sticky=NSEW)
        e3.insert(END, '%s' % arr[i][2])
        arr[i][3] = medList[i].receipt
        e4 = Entry(relief=GROOVE)
        e4.grid(row=r, column=3, sticky=NSEW)
        e4.insert(END, '%s' % arr[i][3])
        arr[i][4] = medList[i].description
        e5 = Entry(relief=GROOVE)
        e5.grid(row=r, column=4, sticky=NSEW)
        e5.insert(END, '%s' % arr[i][4])
        r += 1

def hidespec():
    r=14
    rows, cols = (len(medList), 5)
    arr = [[0] * cols] * rows
    for i in range(len(medList)):
        if not medList[i].receipt:
            arr[i][0] = medList[i].code
            e1 = Entry(relief=GROOVE)
            e1.grid(row=r, column=0, sticky=NSEW)
            e1.insert(END, '%s' % arr[i][0])
            arr[i][1] = medList[i].name
            e2 = Entry(relief=GROOVE)
            e2.grid(row=r, column=1, sticky=NSEW)
```

```
            e2.insert(END, '%s' % arr[i][1])
            arr[i][2] = medList[i].cost
            e3 = Entry(relief=GROOVE)
            e3.grid(row=r, column=2, sticky=NSEW)
            e3.insert(END, '%s' % arr[i][2])
            arr[i][3] = medList[i].receipt
            e4 = Entry(relief=GROOVE)
            e4.grid(row=r, column=3, sticky=NSEW)
            e4.insert(END, '%s' % arr[i][3])
            arr[i][4] = medList[i].description
            e5 = Entry(relief=GROOVE)
            e5.grid(row=r, column=4, sticky=NSEW)
            e5.insert(END, '%s' % arr[i][4])
        r += 1


window = Tk()

window.geometry('900x700')
selected = BooleanVar()
selected.set(False)
medList = []
verif = BooleanVar()

lb = Label(window, text="Information's card ", fg='red', font=('Times New Roman Bold', 18))
lb.grid(column=1, row=0)
l1 = Label(window, text="Code of the medicine ", font=('Times New Roman', 14))
l2 = Label(window, text="Name of the medicine ", font=('Times New Roman', 14))
l3 = Label(window, text="Price of the medicine ", font=('Times New Roman', 14), )
l4 = Label(window, text="Description  ", font=('Times New Roman', 14))
l5 = Label(window, text=" ", font=('Times New Roman', 14))
l6 = Label(window, text="Enter the code of the medicine to delete", font=('Times New Roman',
14))

l1.grid(column=0, row=1)
l2.grid(column=0, row=3)
l3.grid(column=0, row=5)
l4.grid(column=0, row=7)
l5.grid(column=0, row=9)
l6.grid(column=3, row=10)
comb1 = Combobox(window)
comb1['values'] = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
comb1.current(0)
ten2 = Entry(window, width=20, bd=3, justify=LEFT)
ten3 = Entry(window, width=20, bd=3, justify=LEFT)
ten4 = Entry(window, width=20, bd=3, justify=LEFT)
ten5 = Entry(window, width=20, bd=3, justify=LEFT)
rad1 = Radiobutton(window, text='Need receipt', value=True, variable=selected)
rad2 = Radiobutton(window, text="Don't need receipt", value=False, variable=selected)
comb1.grid(column=0, row=2)
ten2.grid(column=0, row=4)
ten3.grid(column=0, row=6)
```

```
ten4.grid(column=0, row=8)
ten5.grid(column=3, row=11)
rad1.grid(column=0, row=10)
rad2.grid(column=1, row=10)
btn = Button(window, text='Add new medicine', command=click)
btn1 = Button(window, text='Show list', command=show)
btn2 = Button(window, text='Delete', command=DeList)
btn3 = Button(window, text='Show medicine', command=hidespec)
btn.grid(column=0, row=11)
btn1.grid(column=1, row=12)
btn2.grid(column=4, row=11)
btn3.grid(column=2, row=12)

window.mainloop()
```

Лабораторная работа №5
«Работа с файлами»

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 (по указанному пути) 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.

Решение:

```
import distutils.util
from tkinter import *
from tkinter import messagebox
from tkinter.ttk import Radiobutton
from tkinter.ttk import Combobox
from tkinter import ttk
from collections import namedtuple
from tkinter import filedialog
import csv


class Medicament:
    code = 1
    name = 'paracetamol'
    cost = 1000
    receipt = True
    description = 'braise, amoxiciline'

    def __init__(self, code1, name1, cost1, receipt1, description1):
        self.code = code1
        self.name = name1
        self.cost = cost1
        self.receipt = receipt1
        self.description = description1

    def __str__(self):
        return str(self.code) + ' ' + self.name + ' ' + str(self.cost) + ' ' + str(
            self.receipt) + ' ' + self.description

    def show(self):
        print(self.code, self.name, self.cost, self.receipt, self.description)

    def Get_code(self):
```

```
        return self.code

    def Get_name(self):
        return self.name

    def Get_cost(self):
        return self.cost

    def Get_description(self):
        return self.description

    def Get_receipt(self):
        return self.receipt


def click():
    prix = 0
    cod = int(comb1.get())
    for i in range(len(medList)):
        if cod == medList[i].Get_code():
            messagebox.showerror('Error input', 'This code already exist! Choose another one')
            cod = 0
    nom = ten2.get()
    try:
        prix = int(ten3.get())
    except ValueError:
        try:
            prix = float(ten3.get())
        except ValueError:
            messagebox.showerror('Error input', 'The price can be an integer or a float')

    descr = ten4.get()
    recp = selected.get()
    if not cod or not nom or not prix or not descr:
        messagebox.showerror('Title', 'Fill all the blank space')
    else:
        med = Medicament(cod, nom, prix, recp, descr)
        medList.append(med)


def DeList():
    verif = False
    try:
        delCod = int(ten5.get())
    except ValueError:
        messagebox.showerror('Error input', 'The code should be an integer')
        delCod = 0
    if not delCod:
        messagebox.showerror('Error', 'Fill the blank place')
    else:
        for i in range(len(medList)):
            if delCod == medList[i].code:
```

```
            del medList[i]
            verif = True
            break
    if not verif:
        messagebox.showerror('Error input', 'This code doesnt exit')
    else:
        messagebox.showinfo('Success', 'The code is successfully delete! Press Show again to see
the change')


def show():
    for i in tree.get_children():
        tree.delete(i)
    for i in range(len(medList)):
        tree.insert(parent='', index=0, values=(
            medList[i].code, medList[i].name, medList[i].cost, medList[i].receipt,
medList[i].description))


def showSpec():
    for i in tree.get_children():
        tree.delete(i)
    for i in range(len(medList)):
        value=distutils.util.strtobool(medList[i].receipt)
        verif=bool(value)
        print(verif)
        if not verif:
            tree.insert(parent='', index=0, values=(
                medList[i].code, medList[i].name, medList[i].cost, medList[i].receipt,
medList[i].description))


def saveAs():
    global file_name
    file_name = filedialog.asksaveasfilename(title='Save File', filetypes=(("CSV files", "*.csv"),
("All Files", "*.*")))
    if file_name:
        if file_name.endswith(".csv"):
            pass
        else:
            file_name = f'{file_name}.csv'
    print(file_name)
    myFile = open(file_name, "w+", newline="")
    writeIn = csv.writer(myFile, delimiter=',')
    for item in medList:
        writeIn.writerow([item.code, item.name, item.cost, item.receipt, item.description])
    myFile.close()

def saveFile():
    global file_name
    if not file_name:
```

```
        file_name = filedialog.asksaveasfilename(title='Save File', filetypes=(("CSV files",
"*.csv"), ("All Files", "*.*")))
        if file_name:
            if file_name.endswith(".csv"):
                pass
            else:
                file_name = f'{file_name}.csv'
        print(file_name)
        myFile = open(file_name, "w", newline="")
        writeIn = csv.writer(myFile, delimiter=',')
        for item in medList:
            writeIn.writerow([item.code, item.name, item.cost, item.receipt, item.description])
    else:
        myFile = open(file_name, "w", newline="")
        writeIn = csv.writer(myFile, delimiter=',')
        for item in medList:
            writeIn.writerow([item.code, item.name, item.cost, item.receipt, item.description])
        myFile.close()

def readFile():
    file_name = filedialog.askopenfilename(title='Open File', filetypes=(("CSV files", "*.csv"),
("All Files", "*.*")))
    medList.clear()
    output_file = open(file_name, "r")
    readIn = csv.reader(output_file)
    for item in readIn:
        medList.append(Medicament(int(item[0]), item[1], item[2], item[3], item[4]))
    for i in range(len(medList)):
        tree.insert(parent='', index=0, values=(
        medList[i].code, medList[i].name, medList[i].cost, medList[i].receipt,
medList[i].description))

def newFile():
    medList.clear()
    for i in tree.get_children():
        tree.delete(i)

def exitProg():
    window.destroy()

def info():
    messagebox.showinfo('Success', 'This amazing program was written by WENCHENESSE
JOSEPH\nThis program execute a lot of function related to '
                                'widget and file management')


window = Tk()

window.geometry('1000x700')
selected = BooleanVar()
selected.set(False)
medList = []
```

```
file_name=""
menu = Menu(window)
newFunc = Menu(menu)
newFunc.add_command(label='New', command=newFile)
newFunc.add_separator()
newFunc.add_command(label='Open', command=readFile)
newFunc.add_separator()
newFunc.add_command(label='Save', command=saveFile)
newFunc.add_separator()
newFunc.add_command(label='Save as', command=saveAs)
newFunc.add_separator()
newFunc.add_command(label='Exit', command=exitProg)
menu.add_cascade(label='FILE', menu=newFunc)

newFunc1 = Menu(menu)
newFunc1.add_command(label='About', command=info)
menu.add_cascade(label='About the program', menu=newFunc1)
window.config(menu=menu)

col = ('code', 'name', 'price', 'receipt', 'description')

tree = ttk.Treeview(window, columns=col, show='headings')
tree.grid(row=14, column=0, sticky='nsew')

# define headings
tree.heading('code', text='Code')
tree.heading('name', text='Name')
tree.heading('price', text='price')
tree.heading('receipt', text='receipt')
tree.heading('description', text='description')

tree.column("code", anchor=W, width=80)
tree.column("name", anchor=W, width=100)
tree.column("price", anchor=W, width=80)
tree.column("receipt", anchor=W, width=80)
tree.column("description", anchor=W, width=100)

lb = Label(window, text="Information's card ", fg='red', font=('Times New Roman Bold', 18))
lb.grid(column=0, row=0)
l1 = Label(window, text="Code of the medicine ", font=('Times New Roman', 14))
l2 = Label(window, text="Name of the medicine ", font=('Times New Roman', 14))
l3 = Label(window, text="Price of the medicine ", font=('Times New Roman', 14), )
l4 = Label(window, text="Description  ", font=('Times New Roman', 14))
l5 = Label(window, text=" ", font=('Times New Roman', 14))
l6 = Label(window, text="Enter the code of the medicine to delete", font=('Times New Roman',
14))
l7 = Label(window, text="See the medicine without receipt", font=('Times New Roman', 14))

l1.grid(column=0, row=1)
l2.grid(column=0, row=3)
l3.grid(column=0, row=5)
l4.grid(column=0, row=7)
```

```
l5.grid(column=0, row=9)
l6.grid(column=1, row=3)
l7.grid(column=1, row=12)
comb1 = Combobox(window)
comb1['values'] = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
comb1.current(0)
ten2 = Entry(window, width=20, bd=3, justify=LEFT)
ten3 = Entry(window, width=20, bd=3, justify=LEFT)
ten4 = Entry(window, width=20, bd=3, justify=LEFT)
ten5 = Entry(window, width=20, bd=3, justify=LEFT)
rad1 = Radiobutton(window, text='Need receipt', value=True, variable=selected)
rad2 = Radiobutton(window, text="Don't need receipt", value=False, variable=selected)
comb1.grid(column=0, row=2)
ten2.grid(column=0, row=4)
ten3.grid(column=0, row=6)
ten4.grid(column=0, row=8)
ten5.grid(column=1, row=4)
rad1.grid(column=0, row=10)
rad2.grid(column=1, row=10)
btn = Button(window, text='Add new medicine', bg='#2186C1', fg='white', command=click)
btn1 = Button(window, text='Show all', bg='#2186C1', fg='white', command=show)
btn2 = Button(window, text='Delete', bg='#2186C1', fg='white', command=DeList)
btn3 = Button(window, text='No receipt', bg='#2186C1', fg='white', command=showSpec)
btn.grid(column=0, row=11)
btn1.grid(column=1, row=14)
btn2.grid(column=1, row=5)
btn3.grid(column=1, row=13)

window.mainloop()
```

Лабораторная работа №6
«Сетевое программирование»

Задание:

Разработать клиент-серверное приложение для вычисления выражений.
Требования к клиенту:
- отправка на сервер введенного пользователем выражения (числа в восьмеричном представлении, знаки «+» и «-»), например: «33+7-10+2»;
- получение результата вычисления выражения;
- удобный графический интерфейс.

Требования к серверу:
- вычисление полученного от клиента выражения и отправка результата клиенту.

Решение:

Сервер

```
import socket

sock= socket.socket()
sock.bind(('', 3333))
sock.listen(1)
conn, addr=sock.accept()

print('Connected:', addr)
while True:
    data = conn.recv(1024)
    if not data:
        break
    str_r=(str(data.decode('utf-8')))

    print('La phrase recu: ', str_r)
    num=[]
    operand=[]

    operation= str_r.split(',')
    for item in operation:
        print(item)
        if item.isdigit():
            num.append(item)
        else:
            operand.append(item)

    sum = int(num[0])
    i=1
    for item in operand:
        if item == "+":
            sum += int(num[i])
```

```
        else:
            sum -= int(num[i])

        i += 1
    print(sum)
    str_ret=str(sum)
    conn.send(bytes(str_ret,'utf-8'))
conn.close()
```

Клиент

```
import socket
from tkinter import *
from tkinter import messagebox


def OctToDec(value):
    try:
        return int(value, 8)
    except ValueError:
        messagebox.showerror('Error input', 'Enter only octal number')


def click():
    number1 = OctToDec(num1.get())
    print(number1, 'first number')
    number2 = OctToDec(num2.get())
    print(number2, 'second number')
    number3 = OctToDec(num3.get())
    print(number3, 'third number')
    number4 = OctToDec(num4.get())
    print(number4, 'fourth number')
    # somme= number1+number2-number3+number4
    # print(somme)
    if number1 and number2 and number3 and number4:
        numList = [number1,'+', number2,'-', number3,'+', number4]
        str_d=','.join([str(item) for item in numList])
        s.send(bytes(str_d, 'utf-8'))
        str_result=s.recv(1024)
        sum_rec=(int(str(str_result.decode('utf-8'))))
        print('The result in 10 system: ', sum_rec)
        sum_oct=oct(sum_rec)
        res= sum_oct[2:]
        print('The result in octal: ', sum_oct[2:])
        lb7 = Label(window, text=res, font=('Times New Roman', 19))
        lb7.grid(column=1, row=4)
    else:
        messagebox.showerror('Error input', 'Enter the numbers')
```

```
window = Tk()
window.geometry('700x500')

lb1 = Label(window, text='Client part', font=('Times New Roman Bold', 18))
lb1.grid(column=1, row=0)
lb2 = Label(window, text='Enter the numbers', font=('Times New Roman', 14))
lb2.grid(column=0, row=1)
num1 = Entry(window, width=10, bd=3, justify=LEFT)
num2 = Entry(window, width=10, bd=3, justify=LEFT)
num3 = Entry(window, width=10, bd=3, justify=LEFT)
num4 = Entry(window, width=10, bd=3, justify=LEFT)
num1.grid(column=0, row=2)
num2.grid(column=2, row=2)
num3.grid(column=4, row=2)
num4.grid(column=6, row=2)
lb3 = Label(window, text='+', font=('Times New Roman', 14))
lb4 = Label(window, text='-', font=('Times New Roman', 14))
lb5 = Label(window, text='+', font=('Times New Roman', 14))
lb6 = Label(window, text='THE RESULT IS: ', font=('Times New Roman', 16))
lb3.grid(column=1, row=2)
lb4.grid(column=3, row=2)
lb5.grid(column=5, row=2)
lb6.grid(column=0, row=4)

btn = Button(window, text='Send to the server', command=click)
btn.grid(column=1, row=3)

s=socket.socket()
s.connect(('localhost', 3333))
# Main code


window.mainloop()
```

Лабораторная работа №7
«Разработка приложения, взаимодействующего с базой данных»

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)
База данных «Лекарства»

Решение:

```
import pymysql
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from tkinter.ttk import Combobox
import numpy as np



def add():
    compo_n= ten2.get()
    nom=ten4.get()
    prix= int(ten5.get())
    recp= selected.get()

    if not compo_n or not nom or not prix:
        messagebox.showerror('Title', 'Fill all the blank space')
    else:
        addQuery= "INSERT INTO Description (`compo`) VALUES (%s) "
        values=(compo_n)
        curs.execute(addQuery, values)
        cur.commit()
        id=curs.lastrowid
        addQuery1= "INSERT INTO Medicament(`nom`,`prix`,`need_pres`,`id_descr`) VALUES
(%s, %s, %s, %s)"
        values1=(nom, prix, recp, id)
        curs.execute(addQuery1, values1)
        cur.commit()
        print(id)
        if curs.rowcount>0:
            messagebox.showinfo('Success', 'Data added successfully!')
```

```
        else:
            messagebox.showinfo('Failed', 'Data not added successfully!')


def show():
    curs.execute("SELECT  Medicament.nom, Medicament.prix, Medicament.need_pres,
Description.compo FROM "
                 "(Medicament inner join Description on Medicament.id_descr=
Description.ID_desc)")
    data=curs.fetchall()
    curs.execute("SELECT * FROM Description")
    dat=curs.fetchall()
    print(dat)
    curs.execute("SELECT * FROM Medicament")
    dat=curs.fetchall()
    print(dat)
    for item in tree.get_children():
        tree.delete(item)
    for item in data:
        medlist=np.asarray(item)
        if medlist[2]== '0':
            tree.insert(parent='', index=0, values= (medlist[0], medlist[1], "No need", medlist[3]))
        else:
            tree.insert(parent='', index=0, values= (medlist[0], medlist[1], "Need", medlist[3]))
    #print(data)

def delete():
    delname= delEntry.get()
    if not delname:
        messagebox.showerror('Title', 'Fill the blank space')
    else:
        query="DELETE FROM Medicament WHERE nom = '"+delname+"'"
        curs.execute(query)
        cur.commit()
        if curs.rowcount>0:
            messagebox.showinfo('Success', 'Data deleted successfully!')


window = Tk()
window.geometry('900x700')

cur= pymysql.connect(host='localhost', user='root', password='password', db='Pharmacie')
curs=cur.cursor()
selected = BooleanVar()
selected.set(False)

col = ('name', 'price', 'receipt', 'description')

tree = ttk.Treeview(window, columns=col, show='headings')
tree.grid(row=17, column=0, sticky='nsew')

# define headings
```

```
tree.heading('name', text='Name')
tree.heading('price', text='price')
tree.heading('receipt', text='receipt')
tree.heading('description', text='description')

tree.column("name", anchor=W, width=100)
tree.column("price", anchor=W, width=80)
tree.column("receipt", anchor=W, width=80)
tree.column("description", anchor=W, width=100)


l4 = Label(window, text="Name of the medicine ", font=('Times New Roman', 14))
l5 = Label(window, text="Price of the medicine ", font=('Times New Roman', 14), )
l6 = Label(window, text="Description  ", font=('Times New Roman', 14))
l7= Label(window, text="Need prescription", font=('Times New Roman', 14))
l8= Label(window, text="Name of the medicine to delete", font=('Times New Roman', 14))


l4.grid(column=0, row=7)
l5.grid(column=0, row=9)
l6.grid(column=0, row=11)
l7.grid(column=0, row=13)
l8.grid(column=2, row=3)


ten2 = Entry(window, width=20, bd=3, justify=LEFT)
ten4 = Entry(window, width=20, bd=3, justify=LEFT)
ten5 = Entry(window, width=20, bd=3, justify=LEFT)
delEntry = Entry(window, width=20, bd=3, justify=LEFT)

rad1 = Radiobutton(window, text='Need receipt', value=True, variable=selected)
rad2 = Radiobutton(window, text="Don't need receipt", value=False, variable=selected)


ten2.grid(column=0, row=12)
ten4.grid(column=0, row=8)
ten5.grid(column=0, row=10)
rad1.grid(column=0, row=14)
rad2.grid(column=1, row=14)

btn = Button(window, text='Add new medicine', bg='#2186C1', fg='white', command=add)
btn.grid(column=1, row=15)
btn1 = Button(window, text='Show', bg='#2186C1', fg='white', command=show)
btn1.grid(column=1, row=17)
btn2 = Button(window, text='Delete', bg='#2186C1', fg='white', command=delete)
btn2.grid(column=2, row=5)
delEntry.grid(column=2, row=4)



window.mainloop()
```