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 5

Студент: Данилевич Е.Д.

Шифр 211116

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.03 Прикладная информатика

Группа 11ПИ

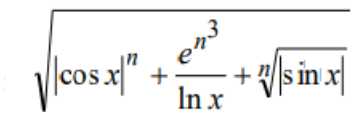
Проверил: Захарова О.В. , Валухов В.А.

Орел 2023

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

Задание:

1. Написать программу, вычисляющую выражение:



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 вещественных положительных элементов списка; вывести содержимое кортежа на экран;
5) найти произведение всех целочисленных элементов списка;
6) сформировать строку из значений элементов списка и посчитать сколько раз встречается в строке указанное пользователем слово;

Решение:

Задание 1:

```
# -*- coding: utf-8 -*-
"""
Created on Wed Sep 20 08:41:38 2023

@author: Egor
"""

import math as m

# Ввод значений с клавиатуры и проверка на ошибки
try:
    n = float(input("Введите значение n: "))
    x = float(input("Введите значение x: "))
except ValueError:
    print("Ошибка: Введите корректные числовые значения.")
    exit(1)
if x != 1 and x != m.e:
    pass
else:
    print("Ошибка: Введите корректные числовые значения.")
    exit(1)

result = m.sqrt(abs(m.cos(x))**n + (m.e**(n**3))/(m.log(x)) + (abs(m.sin(x)))**(1/n))
print(result)
```

Задание 2:

```
    def show_list(my_list):
```

```
    print("Список:", my_list)

def add_element(my_list):
    var_type = input("Выберите тип значения (int, float, string): ")
    value = None

    if var_type == 'int':
        value = int(input("Введите целочисленный элемент: "))
        my_list.append(value)

    elif var_type == 'float':
        value = float(input("Введите вещественный элемент: "))
        my_list.append(value)

    elif var_type == 'string':
        value = input("Введите символьный элемент: ")
        my_list.append(value)
    else:
        print("Неверный тип.")
        return

def remove_element(my_list):
    var_type = input("Выберите тип значения (int, float, string): ")
    value = None

    if var_type == 'int':
        element = int(input("Введите элемент для удаления: "))
        if element in my_list:
            my_list.remove(element)
        else:
            print("Такой элемент не найден в списке.")

    elif var_type == 'float':
        element = float(input("Введите элемент для удаления: "))
        if element in my_list:
            my_list.remove(element)
        else:
            print("Такой элемент не найден в списке.")

    elif var_type == 'string':
        element = input("Введите элемент для удаления: ")
        if element in my_list:
            my_list.remove(element)
        else:
            print("Такой элемент не найден в списке.")
    else:
        print("Неверный тип.")
        return
```

```
def create_float_tuple(my_list):
    float_elements = tuple([x for x in my_list if( isinstance(x,float) and x > 0)])
    print("Кортеж вещественных положительных элементов:", float_elements)

def calculate_product(my_list):
    product = 1
    for element in my_list:
        if isinstance(element, int):
            product *= element
    print("Произведение всех целочисленных элементов:", product)

def find_word_count(my_list):
    word = input("Введите слово для поиска в строке: ")
    joined_string = ' '.join(my_list)
    count = joined_string.count(word)
    print("Строка:", joined_string)
    print(f"Слово '{word}' встречается {count} раз(а) в строке.")

def symmetric_difference(my_list):

    temp_list = []
    print("Введите множество:")

    while 1 :
        var_type = input("Выберите тип значения (int, float, string) иначе выход: ")
        value = None
        if var_type == 'int':
            value = int(input("Введите целочисленный элемент: "))
            temp_list.append(value)

        elif var_type == 'float':
            value = float(input("Введите вещественный элемент: "))
            temp_list.append(value)

        elif var_type == 'string':
            value = input("Введите символьный элемент: ")
            temp_list.append(value)
        else:
            break

    m1 = set(temp_list)
    m2 = set(my_list)
    symmetric_diff = m1.symmetric_difference(m2)
    print("Симметричная разница множеств M1 и M2:", symmetric_diff)

def create_dict_with_odd_keys(my_list):
    my_dict = {i: element for i, element in enumerate(my_list, 1) if i % 2 != 0}
    print("Словарь с элементами, у которых нечетные ключи:")
    for key, value in my_dict.items():
        print(f"{key}: {value}")
```

```
my_list = [1.2,5.3,"str",1,2,3,-1]

while True:
    print("Список команд:")
    print("1 - Показать значения списка")
    print("2 - Добавить элемент в список")
    print("3 - Удалить элемент из списка")
    print("4 - Сформировать кортеж из вещественных положительных элементов")
    print("5 - Найти произведение всех целочисленных элементов")
    print("6 - Сформировать строку и посчитать вхождения слова")
    print("7 - Задать множество M1 и найти симметричную разницу с M2")
    print("8 - Сформировать словарь и отобразить элементы с нечетными ключами")
    print("9 - Выйти из программы")

    command = input("Выберите действие: ")

    if command == "1":
        show_list(my_list)
    elif command == "2":
        add_element(my_list)
    elif command == "3":
        remove_element(my_list)
    elif command == "4":
        create_float_tuple(my_list)
    elif command == "5":
        calculate_product(my_list)
    elif command == "6":
        find_word_count(my_list)
    elif command == "7":
        symmetric_difference(my_list)
    elif command == "8":
        create_dict_with_odd_keys(my_list)
    elif command == "9":
        break
    else:
        print("Команда не определена")
    print(" ")
```

Задание 3:

```
def square_area(side):
    return side * side

def trapezoid_area(base1, base2, height):
    return 0.5 * (base1 + base2) * height

def parallelogram_area(base, height):
    return base * height

while True:
    print("\nВыберите фигуру для вычисления площади:")
    print("S - Квадрат")
    print("T - Трапеция")
```

```
print("P - Параллелограмм")
print("E - Выход")

command = input("Введите команду выбора: ").upper()

if command == 'S':
    try:
        side = float(input("Введите длину стороны квадрата: "))
        area = square_area(side)
        print(f"Площадь квадрата: {area}")
    except ValueError:
        print("Ошибка: Неверный ввод. Введите число.")
elif command == 'T':
    try:
        base1 = float(input("Введите длину большей основы трапеции: "))
        base2 = float(input("Введите длину меньшей основы трапеции: "))
        height = float(input("Введите высоту трапеции: "))
        area = trapezoid_area(base1, base2, height)
        print(f"Площадь трапеции: {area}")
    except ValueError:
        print("Ошибка: Неверный ввод. Введите число.")
elif command == 'P':
    try:
        base = float(input("Введите длину основания параллелограмма: "))
        height = float(input("Введите высоту параллелограмма: "))
        area = parallelogram_area(base, height)
        print(f"Площадь параллелограмма: {area}")
    except ValueError:
        print("Ошибка: Неверный ввод. Введите число.")
elif command == 'E':
    break
else:
    print("Ошибка: Неверный выбор. Попробуйте еще раз.")
```

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:

Задание 1:
```
import math as m
from sys import exit

def get_float_input(prompt):
    try:
        value = float(input(prompt))
        return value
    except ValueError:
        print("Ошибка: Введите корректные числовые значения.")
        exit(1)

def compute_result(get_float_input):

    n = get_float_input("Введите значение n: ")
    x = get_float_input("Введите значение x: ")
    try:
        return m.sqrt(abs(m.cos(x))**n + (m.e**(n**3))/(m.log(x)) + (abs(m.sin(x)))**(1/n))
    except ZeroDivisionError:
        print("Ошибка: Деление на ноль.")
        exit(1)



result = compute_result(get_float_input)
print(result)
```

Задание 2:
```
from functools import reduce
```

```
from math import pi, tan


def S(values):
    print(L)
    try:
        side = values.pop(0)
        area = side * side
        print("Площадь квадрата: ", area)
        return True
    except:
        print("Заданы неверные значения!")


def T(values):
    print(L)

    try:
        base1, base2, height = values.pop(0), values.pop(0), values.pop(0)
        area = 0.5 * (base1 + base2) * height
        print("Площадь трапеции: ", area)
        return True
    except:
        print("Заданы неверные значения!")


def P(values):
    print(L)

    try:
        base, height = values.pop(0), values.pop(0)
        area = base * height
        print("Площадь параллелограмма: ", area)
        return True
    except:
        print("Заданы неверные значения!")


def E():
    exit(0)


L = [['S', 'T', 'P', 'E'], [2], [2, 3, 4], [3, 2],]

reduce(lambda a, b: ((L[0][0].upper() == "S" and S(b))
                     or (L[0][0] == "T".upper() and T(b))
                     or (L[0][0] == "P".upper() and P(b))
                     or (L[0][0] == "E".upper() and E()))
       and (L[0].pop(0)), L[1:], True)
```

Лабораторная работа №3
«Объектно-ориентированное программирование»

Задание:

В программе разработать класс «Преподаватель» с обязательными атрибутами: табельный номер, ФИО, пол, дата рождения, адрес, телефон, преподаваемая дисциплина, стаж работы. Вводимую информацию о преподавателях хранить в списке объектов класса «Преподаватель».

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 преподавателей преподающих указанную дисциплину.

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 Teacher:
    def __init__(self, emp_id, full_name, gender, birthdate, address, phone, subject, experience,
*args):
        self.emp_id = emp_id # табельный номер
        self.full_name = full_name  # ФИО
        self.gender = gender # Пол
        self.birthdate = birthdate # Дата рождения
        self.address = address # Адрес
        self.phone = phone # Телефон
        self.subject = subject # Преподаваемая дисциплина
        self.experience = experience # Стаж работы
        self.additional_info = args  # Сохраняем

    def display_info(self):
        print(f"Табельный номер: {self.emp_id}")
        print(f"ФИО: {self.full_name}")
        print(f"Пол: {self.gender}")
        print(f"Дата рождения: {self.birthdate}")
        print(f"Адрес: {self.address}")
        print(f"Телефон: {self.phone}")
        print(f"Преподаваемая дисциплина: {self.subject}")
        print(f"Стаж работы: {self.experience} лет")
        print(self.additional_info)

teachers_list = [] # Список для хранения объектов "Преподаватель"


def add_teacher():
    """
    1. Добавление записи о преподавателе

    Returns
    -------
    None.
```

```
    """
    emp_id = input("Введите табельный номер: ")
    full_name = input("Введите ФИО: ")
    gender = input("Введите пол: ")
    birthdate = input("Введите дату рождения: ")
    address = input("Введите адрес: ")
    phone = input("Введите телефон: ")
    subject = input("Введите преподаваемую дисциплину: ")
    experience = input("Введите стаж работы: ")
    another = input("Введите дополнительную информацию: ")

    teacher = Teacher(emp_id, full_name, gender, birthdate, address, phone, subject, experience,
another)
    teachers_list.append(teacher)
    print("Информация о преподавателе добавлена.")

def remove_teacher():
    """
    2. Удаление записи о преподавателе

    Returns
    -------
    None.

    """
    if not teachers_list:
        print("Список преподавателей пуст.")
        return

    emp_id = input("Введите табельный номер преподавателя, которого нужно удалить: ")

    for teacher in teachers_list:
        if teacher.emp_id == emp_id:
            teachers_list.remove(teacher)
            print(f"Преподаватель с табельным номером {emp_id} удален.")
            return

    print(f"Преподаватель с табельным номером {emp_id} не найден.")

def display_all_teachers():
    """
    3. вывести список преподавателей

    Returns
    -------
    None.

    """
    if not teachers_list:
        print("Список преподавателей пуст.")
        return
```

```
    print("Информация о всех преподавателях:")
    for teacher in teachers_list:
        teacher.display_info()
        print()

def search_teachers_by_subject():
    """
    4. поиск преподавателей по предмету

    Returns
    -------
    None.

    """
    subject = input("Введите дисциплину для поиска преподавателей: ")
    found_teachers = [teacher for teacher in teachers_list if teacher.subject == subject]

    if not found_teachers:
        print(f"Преподавателей, преподающих дисциплину '{subject}', не найдено.")
    else:
        print(f"Преподаватели, преподающие дисциплину '{subject}':")
        for teacher in found_teachers:
            teacher.display_info()
            print()

# Главное меню программы
while True:
    print("\n  Меню программы:")
    print("1. Добавление информации о преподавателе")
    print("2. Удаление информации о преподавателе")
    print("3. Отображение информации обо всех преподавателях")
    print("4. Поиск преподавателей по дисциплине")
    print("5. Выход")

    command = input("Выберите действие: ")

    if command == "1":
        add_teacher()
    elif command == "2":
        remove_teacher()
    elif command == "3":
        display_all_teachers()
    elif command == "4":
        search_teachers_by_subject()
    elif command == "5":
        print("Программа завершена.")
        break
    else:
        print("Неверный выбор.")
```

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:

```
import tkinter as tk
from tkinter import messagebox

class Teacher:
    def __init__(self, emp_id, full_name, gender, birthdate, address, phone, subject, experience, additional_info):
        self.emp_id = emp_id
        self.full_name = full_name
        self.gender = gender
        self.birthdate = birthdate
        self.address = address
        self.phone = phone
        self.subject = subject
        self.experience = experience
        self.additional_info = additional_info

    def display_info(self):
        return f"Табельный номер: {self.emp_id}\nФИО: {self.full_name}\nПол: {self.gender}\nДата рождения: {self.birthdate}\n" \
               f"Адрес: {self.address}\nТелефон: {self.phone}\nПреподаваемая дисциплина: {self.subject}\n" \
               f"Стаж работы: {self.experience} лет\nДополнительная информация: {self.additional_info}\n"

class Application(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Система управления преподавателями")

        self.teachers_list = []

        # Создание элементов интерфейса
        self.label_emp_id = tk.Label(self, text="Табельный номер:")
        self.entry_emp_id = tk.Entry(self)

        self.label_full_name = tk.Label(self, text="ФИО:")
        self.entry_full_name = tk.Entry(self)

        self.label_gender = tk.Label(self, text="Пол:")
        self.entry_gender = tk.Entry(self)
```

```
        self.label_birthdate = tk.Label(self, text="Дата рождения:")
        self.entry_birthdate = tk.Entry(self)

        self.label_address = tk.Label(self, text="Адрес:")
        self.entry_address = tk.Entry(self)

        self.label_phone = tk.Label(self, text="Телефон:")
        self.entry_phone = tk.Entry(self)

        self.label_subject = tk.Label(self, text="Преподаваемая дисциплина:")
        self.entry_subject = tk.Entry(self)

        self.label_experience = tk.Label(self, text="Стаж работы:")
        self.entry_experience = tk.Entry(self)

        self.label_additional_info = tk.Label(self, text="Дополнительная информация:")
        self.entry_additional_info = tk.Entry(self)

        self.button_add_teacher = tk.Button(self, text="Добавить преподавателя",
command=self.add_teacher)
        self.button_remove_teacher = tk.Button(self, text="Удалить преподавателя",
command=self.remove_teacher)
        self.button_display_teachers = tk.Button(self, text="Отобразить преподавателей",
command=self.display_all_teachers)
        self.button_search_teachers = tk.Button(self, text="Поиск по дисциплине",
command=self.search_teachers_by_subject)
        self.button_exit = tk.Button(self, text="Выход", command=self.quit)

        # Расположение элементов интерфейса на сетке
        self.label_emp_id.grid(row=0, column=0, sticky="e")
        self.entry_emp_id.grid(row=0, column=1)

        self.label_full_name.grid(row=1, column=0, sticky="e")
        self.entry_full_name.grid(row=1, column=1)

        self.label_gender.grid(row=2, column=0, sticky="e")
        self.entry_gender.grid(row=2, column=1)

        self.label_birthdate.grid(row=3, column=0, sticky="e")
        self.entry_birthdate.grid(row=3, column=1)

        self.label_address.grid(row=4, column=0, sticky="e")
        self.entry_address.grid(row=4, column=1)

        self.label_phone.grid(row=5, column=0, sticky="e")
        self.entry_phone.grid(row=5, column=1)

        self.label_subject.grid(row=6, column=0, sticky="e")
        self.entry_subject.grid(row=6, column=1)

        self.label_experience.grid(row=7, column=0, sticky="e")
```

```
        self.entry_experience.grid(row=7, column=1)

        self.label_additional_info.grid(row=8, column=0, sticky="e")
        self.entry_additional_info.grid(row=8, column=1)

        self.button_add_teacher.grid(row=9, column=0, columnspan=2)
        self.button_remove_teacher.grid(row=10, column=0, columnspan=2)
        self.button_display_teachers.grid(row=11, column=0, columnspan=2)
        self.button_search_teachers.grid(row=12, column=0, columnspan=2)
        self.button_exit.grid(row=13, column=0, columnspan=2)

    def add_teacher(self):
        emp_id = self.entry_emp_id.get()
        full_name = self.entry_full_name.get()
        gender = self.entry_gender.get()
        birthdate = self.entry_birthdate.get()
        address = self.entry_address.get()
        phone = self.entry_phone.get()
        subject = self.entry_subject.get()
        experience = self.entry_experience.get()
        additional_info = self.entry_additional_info.get()

        teacher = Teacher(emp_id, full_name, gender, birthdate, address, phone, subject,
experience, additional_info)
        self.teachers_list.append(teacher)

        self.clear_entries()
        messagebox.showinfo("Информация", "Информация о преподавателе добавлена.")

    def remove_teacher(self):
        emp_id = self.entry_emp_id.get()
        for teacher in self.teachers_list:
            if teacher.emp_id == emp_id:
                self.teachers_list.remove(teacher)
                messagebox.showinfo("Информация", f"Информация о преподавателе
{teacher.full_name} удалена.")
                break
        else:
            messagebox.showinfo("Информация", f"Преподаватель с табельным номером
{emp_id} не найден.")

    def display_all_teachers(self):
        if not self.teachers_list:
            messagebox.showinfo("Информация", "Список преподавателей пуст.")
            return

        info = "\n\n".join([teacher.display_info() for teacher in self.teachers_list])
        messagebox.showinfo("Информация о преподавателях", info)

    def search_teachers_by_subject(self):
        subject = self.entry_subject.get()
        found_teachers = [teacher for teacher in self.teachers_list if teacher.subject == subject]
```

```
        if not found_teachers:
            messagebox.showinfo("Информация", f"Преподавателей, преподающих дисциплину
'{subject}', не найдено.")
        else:
            info = "\n\n".join([teacher.display_info() for teacher in found_teachers])
            messagebox.showinfo("Информация о преподавателях", info)

    def clear_entries(self):
        self.entry_emp_id.delete(0, tk.END)
        self.entry_full_name.delete(0, tk.END)
        self.entry_gender.delete(0, tk.END)
        self.entry_birthdate.delete(0, tk.END)
        self.entry_address.delete(0, tk.END)
        self.entry_phone.delete(0, tk.END)
        self.entry_subject.delete(0, tk.END)
        self.entry_experience.delete(0, tk.END)
        self.entry_additional_info.delete(0, tk.END)

if __name__ == "__main__":
    app = Application()
    app.mainloop()
```