Лабораторная работа №6  
Гайворонская Екатерина Александровна   
010304 КМСб-о23

ЛАБОРАТОРНАЯ РАБОТА 6.

Использование программ

Задачи:

1. Изучить порядок описания подпрограмм на языке Python.

2. Научиться использовать глобальные и локальные переменные при применении подпрограмм.

3. Решить задача, составив алгоритмы программы.

Вариант p = 0 q = 1

**Порядок выполнения работы**

**Задача1.** На плоскости задан правильный семиугольник с вершинами А1(x1,y1), A2(x2,y2),..., A7(x7, y7).Найдите его площадь. Для решения использовать подпрограммы вычисления длин сторон и вычисления площади треугольников.

Порядок решения:

1. Создается функцию. Вычисляем расстояние по формуле
2. Формула Герона
3. Вычисляем площадь через треугольники

Программа для решения задачи:

import math

def distance(x1, y1, x2, y2):

    """Вычисляет расстояние между двумя точками"""

    return math.sqrt((x2 - x1)\*\*2 + (y2 - y1)\*\*2)

def triangle\_area(a, b, c):

    """Вычисляет площадь треугольника по длинам его сторон"""

    s = (a + b + c) / 2

    return math.sqrt(s \* (s - a) \* (s - b) \* (s - c))

# Ввод координат вершин семиугольника

x1, y1 = map(float, input("Введите координаты первой вершины (x1 y1): ").split())

x2, y2 = map(float, input("Введите координаты второй вершины (x2 y2): ").split())

x3, y3 = map(float, input("Введите координаты третьей вершины (x3 y3): ").split())

x4, y4 = map(float, input("Введите координаты четвертой вершины (x4 y4): ").split())

x5, y5 = map(float, input("Введите координаты пятой вершины (x5 y5): ").split())

x6, y6 = map(float, input("Введите координаты шестой вершины (x6 y6): ").split())

x7, y7 = map(float, input("Введите координаты седьмой вершины (x7 y7): ").split())

# Вычисление длины стороны семиугольника

a = distance(x1, y1, x2, y2)

# Вычисление площади одного треугольника

h = a \* math.tan(math.pi / 3)

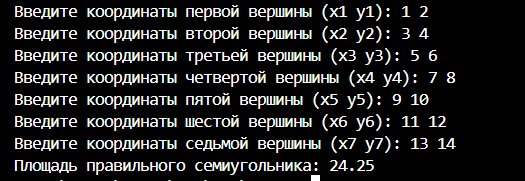
triangle\_area = triangle\_area(a, a, h)

# Вычисление общей площади семиугольника

total\_area = 7 \* triangle\_area

print(f"Площадь правильного семиугольника: {total\_area:.2f}")

Результаты выполнения программы:



**Задача2.** Вычислить , a1, a2,…,a15-заданный массив

Порядок решения:

1. Создаем функцию, которая будет вычислять g
2. Введите массив

Программа для решения задачи:

# Функция g(x, y) = x^2 + xy + y^2

def g(x, y):

    return x \*\* 2 + x \* y + y \*\* 2

# Ввод массива a с клавиатуры

a = []

for i in range(15):

    value = int(input(f"Введите a{i+1}: "))

    a.append(value)

# Вычисление суммы

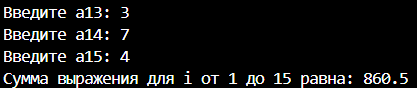
result = 0

for i in range(15):

    result += g(a[i] + 1, a[i] / 2)

print("Сумма выражения для i от 1 до 15 равна:", result)

Результаты выполнения программы:



**Задача3.** Напишите подпрограмму, которая анализирует поток вводимых слов, выделяет слова, которые начинаются и заканчиваются гласной, замещает в таких словах все буквы, кроме первой и последней на знак \*. Слова вводятся, пока не введено слово «стоп».

Порядок решения:

1. Создаем функцию, которая проверяет первую и последнею букву.
2. Если введено стоп, программа прекращается.
3. Заменяем середину слова на \*.

Программа для решения задачи:

def words(a):

    if a[0] in 'аоуыэеёиюяАОУЫЭЕЁИЮЯ' and a[-1] in 'аоуыэеёиюяАОУЫЭЕЁИЮЯ':

        words = a[0] + '\*' \* (len(a)-2) + a[-1]

        return words

    else:

        return a

while True:

    a = input("Введите слово:")

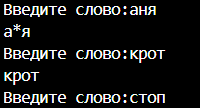
    if a.lower() == 'стоп':

        break

    else:

        print(words(a))

Результаты выполнения программы:



**Задача4** . Напишите подпрограмму, которая для каждой строки матрицы А, размером m×n, находит наибольший элемент. Для подпрограммы в качестве входного параметра взять номер строки. Функцию max не использовать. Все найденные наибольшие элементы напечатать с указанием строк, к которым они относятся

Порядок решения:

1. Создаем функцию для поиска максимального элемента.

Программа для решения задачи:

def find\_max\_element\_in\_row(matrix, row\_number):

    max\_element = matrix[row\_number][0]

    for element in matrix[row\_number]:

        if element > max\_element:

            max\_element = element

    return max\_element

# Ввод размерности матрицы

m = int(input("Введите количество строк матрицы: "))

n = int(input("Введите количество столбцов матрицы: "))

# Ввод элементов матрицы

A = []

for i in range(m):

    row = [int(x) for x in input(f"Введите элементы {i+1}-й строки (через пробел): ").split()]

    A.append(row)

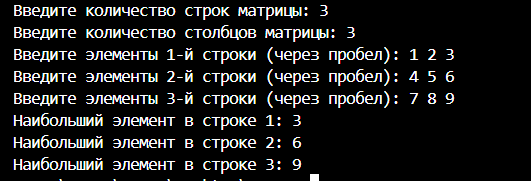
# Поиск и вывод наибольших элементов в каждой строке

for i in range(m):

    max\_element\_in\_row = find\_max\_element\_in\_row(A, i)

    print(f"Наибольший элемент в строке {i + 1}: {max\_element\_in\_row}")

Результаты выполнения программы:



**Задача5.** Подпрограмма занимается шифрованием слов, вводимых с экрана. Правило шифрования следующее. Каждый символ замещается на число - порядковый номер буквы в алфавите, к которому прибавлено значение остатка от деления порядкового номера буквы на 3. Исходные данные для подпрограммы –слово, а результат зашифрованное слово. Для работы основная программа получаем текстовую строку, слова которые разделены не более, чем одним пробелом. Пусть исходная строка состоит только из строчных букв русского алфавита. Выводит результат шифрования в виде строки. Пробелы шифровке не подлежат. Указание: использовать функцию ord (), которая возвращает код символа. Например, строка кода print (ord ('A')) напечатает число 65 – код символа «A». Аргумент функции ord () символ. Функция, которая по числу возвращает символ с данным кодом называется chr ().

Порядок решения:

1. Создаем функцию для шифрования.

Программа для решения задачи:

def encrypt\_word(word):

    encrypted\_word = ""

    for char in word:

        char\_code = ord(char.lower()) - ord('а') + 1

        encrypted\_char\_code = (char\_code + char\_code % 3) % 33

        encrypted\_char = chr(encrypted\_char\_code + ord('а') - 1)

        encrypted\_word += encrypted\_char

    return encrypted\_word

def main():

    user\_input = input("Введите текстовую строку из строчных букв русского алфавита: ")

    words = user\_input.split()

    encrypted\_words = [encrypt\_word(word) for word in words]

    encrypted\_text = " ".join(encrypted\_words)

    print(f"Зашифрованный текст: {encrypted\_text}")

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

    main()

Результаты выполнения программы:



**Задача 6.** Пусть задано произвольное натуральное число N. Находят число - сумму его цифр. Потом для полученного числа находят сумму его цифр. Так, продолжают до тех пор, пока не получится однозначное число. Такое число принято называть цифровым корнем N. Найти для заданного N цифровой корень.

Порядок решения:

1. Создаем функцию для вычисления цифрового корня

Программа для решения задачи:

def digital\_root(n):

    while n > 9:

        i = int (n % 10)

        n = n // 10

        n = n + i

    return n

N = int(input("Введите натуральное число N: "))

result = digital\_root(N)

print("Цифровой корень числа", N, ":", result)

Результаты выполнения программы:

