Отчёт по третьей лабораторной работе   
по Python

(Python LR3)

Выполнил: Соболь Евгений

Группа: ФПММ-ИСТ-19-1бзу

**Вариант: 10**

**Задание 1**

Вводится строка, включающая строчные и прописные буквы. Требуется вывести ту же строку в одном регистре, который зависит от того, каких букв больше. При равном количестве преобразовать в нижний регистр. Например, вводится строка "HeLLo World", она должна быть преобразована в "hello world", потому что в исходной строке малых букв больше. В коде используйте цикл for, строковые методы upper() (преобразование к верхнему регистру) и lower() (преобразование к нижнему регистру), а также методы isupper() и islower(), проверяющие регистр строки или символа.

**Решение:**

# Лабораторная работа № 3. Задание 1. Вариант 10.

niz = 0

ver = 0

st = input("Введите строку:")

for i in range(len(st)):

# Если символ прописной, счётчик +1

if st[i].islower():

niz += 1

# Если символ заглавный, счётчик +1

if st[i].isupper():

ver += 1

# Сравнение счётчиков, и преобразование строки

st = st.lower() if niz >= ver else st.upper()

print(st)

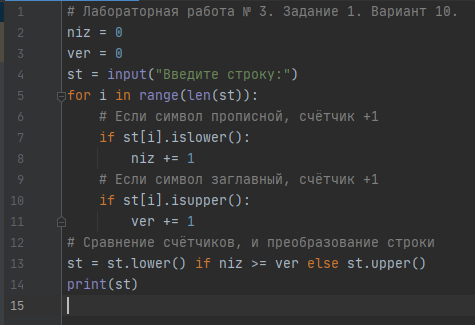


Рисунок -Программный код



Рисунок - Ввод-Вывод

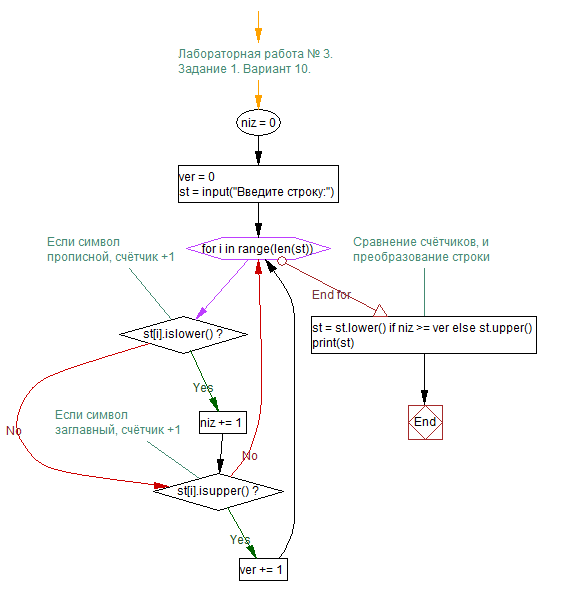


Рисунок -Блок-схема

**Задание 2**

Строковый метод isdigit() проверяет, состоит ли строка только из цифр. Напишите программу, которая запрашивает с ввода два целых числа и выводит их сумму. В случае некорректного ввода программа не должна завершаться с ошибкой, а должна продолжать запрашивать числа.

**Решение:**

# Лабораторная работа № 3. Задание 2. Вариант 10.

while 1:

n1 = input("Введите первое число:")

n2 = input("Введите второе число:")

# Проверяем что оба введённых значения являются числами

if n1.isdigit() and n2.isdigit():

# Выводим сумму

print(int(n1) + int(n2))

break

print("Вы ввели не число! Повторите попытку!")

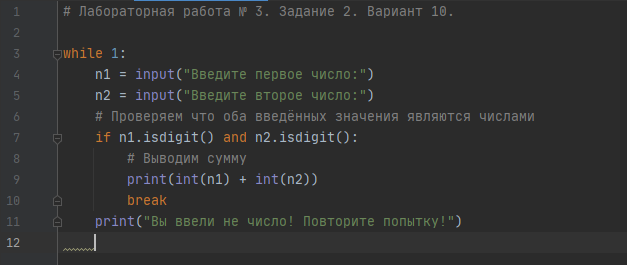


Рисунок -Программный код

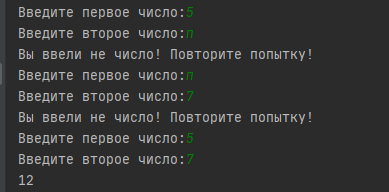


Рисунок -Ввод-вывод

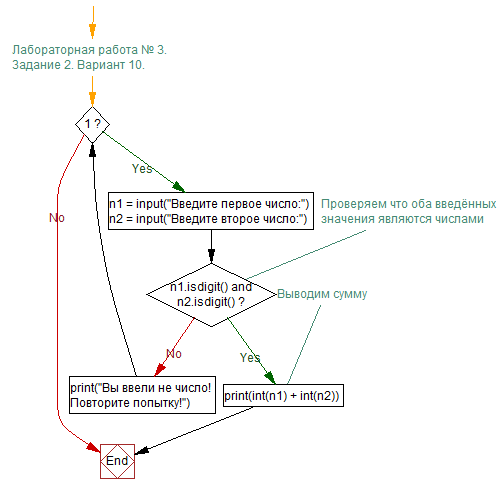


Рисунок -Блок-схема

**Задание 3**

Даны два списка одинаковой длины. Необходимо создать из них словарь таким образом, чтобы элементы первого списка были ключами, а элементы второго — соответственно значениями нашего словаря.

**Решение:**

# Лабораторная работа № 3. Задание 3. Вариант 10.

# Выводим преобразованные в словарь кортежи из соответствующих элементов последовательностей

print(dict(zip(['1', '2', '3', '4'], ["Первый", "Второй", "Третий", "Четвёртый"])))

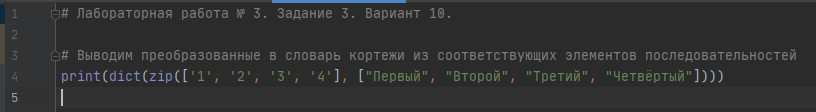


Рисунок - Программный код



Рисунок -Ввод-вывод

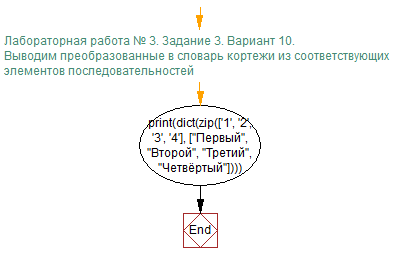


Рисунок -Блок-схема

**Задание 4**

Заполните один кортеж десятью случайными целыми числами от 0 до 5 включительно. Также заполните второй кортеж числами от -5 до 0. Объедините два кортежа с помощью оператора +, создав тем самым третий кортеж. С помощью метода кортежа count() определите в нем количество нулей. Выведите на экран третий кортеж и количество нулей в нем.

**Решение:**

import random

# Функция возвращает случайное число

def ra(a, b):

return random.randint(a, b)

# Функция создаёт кортеж с 10 десятью случайными целыми числами от 0 до 5

def krt10(a, b):

return (ra(a, b), ra(a, b), ra(a, b), ra(a, b), ra(a, b), ra(a, b), ra(a, b), ra(a, b), ra(a, b), ra(a, b))

k = 0

# Создаём первый кортеж

a1 = krt10(0, 5)

# Создаём второй кортеж

a2 = krt10(-5, 0)

# Объединяем кортежы

a3 = a1 + a2

# Считаем количество нулей

for i in range(len(a3)):

if str(a3[i]).count("0"):

k += 1

print("Первый кортеж:", a1)

print("Второй кортеж:", a2)

print("Третий кортеж:", a3)

print("Количество нулей:", k)

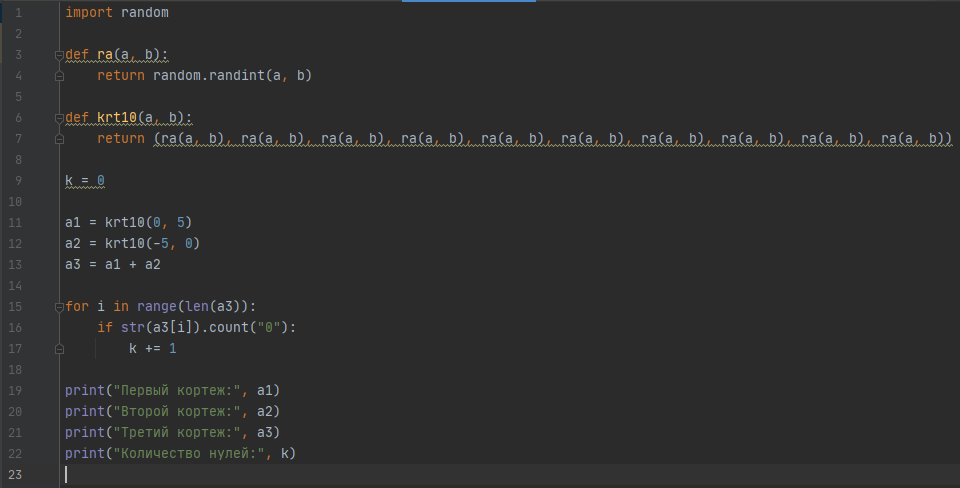


Рисунок - Программный код

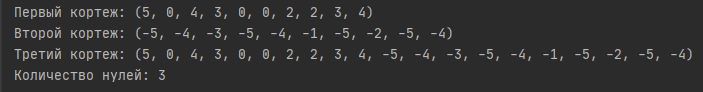


Рисунок -Вывод

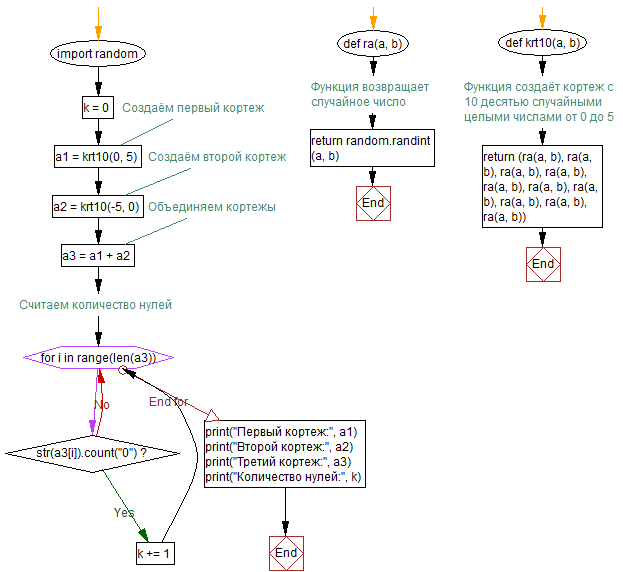


Рисунок -Блок-схема

**Задание 5**

Создайте словарь, связав его с переменной school, и наполните данными, которые бы отражали количество учащихся в разных классах (1а, 1б, 2б, 6а, 7в и т. п.). Внесите изменения в словарь согласно следующему:

а) в одном из классов изменилось количество учащихся,

б) в школе появился новый класс,

с) в школе был расформирован (удален) другой класс.

Для каждого случая вычислите общее количество учащихся в школе.

**Решение:**

import random

masKey = []

masKey1 = []

i = 0

# Функция вывода списка учащихся

def kUch(a):

for key in a:

print("В {0} классе {1} учащ.".format(key, a[key]))

return

# Функция подсчёта учеников в школе

def sUch(c1, a, mas):

s = 0

# Вычисление суммы учеников

for i in range(c1):

s += a[mas[i]]

return s

# Фунция для генерации классов

def randClass(coun):

d = []

sl = {}

i = 0

while i < coun:

k = 0

# Массив с бувками для генерации названия классов

a = ['А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'И', 'Й', 'К', 'Л', 'М', 'Н', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Э', 'Ю', 'Я']

# Генерания цифры для названия класса

b = random.randint(1, 11)

# Генерация названия класса

c = str(b) + a[random.randint(0, len(a)-1)]

# Проверка уникальности сгенерированного названия класса

if c in d:

k += 1

# Если название уникальное, то добавляется в массив уникальных названий

if k == 0:

i += 1

d.append(c)

# Генерация словаря из массива уникальных названий классов

for i in range(coun):

sl[d[i]] = random.randint(15, 35)

return sl

# Генерация количества классов

c = random.randint(7, 15)

# Генерируем классы

school = randClass(c)

# Заполняю массив ключами от сгенерированного словаря

for key in school:

masKey.append(key)

i += 1

# Вывожу список всех классов с количеством учеников в их

print("Список классов в школе:")

kUch(school)

# Вывожу сумму учеников в классах до изменений

print("-------------------------------------")

print("Всего учеников в школе до изменений: ", sUch(c, school, masKey))

print("-------------------------------------")

# Выбираю случайный класс, в котором изменится количество учащихся

masi = masKey[random.randint(0, c-1)]

# Сгенерировал случайное число, на которое поменяется количество учащихся

r = random.randint(35, 50)

# Вывожу информацию об изменениях

print("В классе {0} было {1} учащ. ,а стало {2} учащ.".format(masi, school[masi], r))

# Изменяю количество учащихся в классе

school[masi] = r

# Вывожу количество учащихся в школе после изменения А

print("-------------------------------------")

print("Всего учеников в школе после изменения A: ", sUch(c, school, masKey))

print("-------------------------------------")

# Генерация класса с уникальным названием

while 1:

b = 0

masKey1 = []

# Создаю один класс

cl1 = randClass(1)

# Узнаю его номер (ключ)

for key in cl1:

masKey1.append(key)

# Проверяю на уникальность

if masKey1[0] in masKey:

b = 1

# Если название уникально, завершаем процесс генерации

if b == 0:

break

# Вывожу информацию о классе который появится

print("В школе появился новый класс", masKey1[0])

# Добавляю в словарь school новый класс

school.update(cl1)

# Вывожу список классов после появления нового

print("Список классов в школе после появления нового:")

kUch(school)

# Вывожу количество учащихся в школе после изменения Б

print("-------------------------------------")

print("Всего учеников в школе после изменения Б: ", sUch(c+1, school, masKey + masKey1))

print("-------------------------------------")

# Расформировываю случайный класс

detCl = school.popitem()

# Вывожу информацию о расформированном классе

print("В школе был расформирован {0} класс".format(detCl[0]))

# Вычисляю массив названий классов без удалённого

masO = masKey+masKey1

masJ = [detCl[0]]

masJ = list(set(masO)-set(masJ))

# Вывожу количество учащихся в школе после изменения C

print("-------------------------------------")

print("Всего учеников в школе после изменения С: ", sUch(c, school, masJ))

print("-------------------------------------")

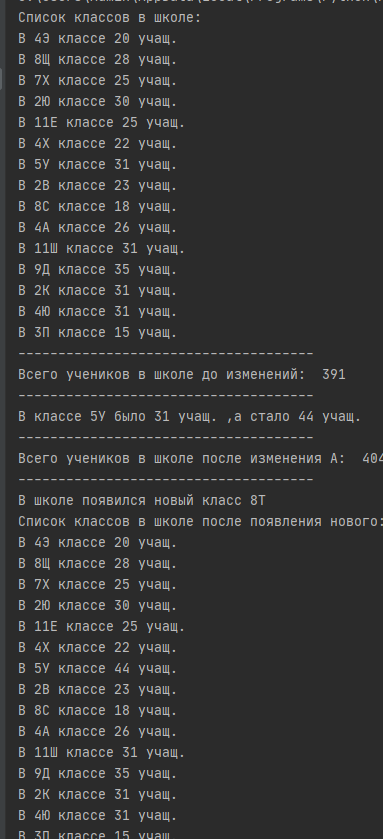


Рисунок -Вывод (1)

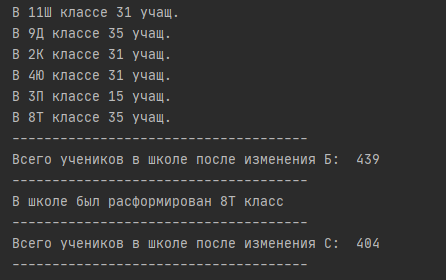


Рисунок -Вывод (2)

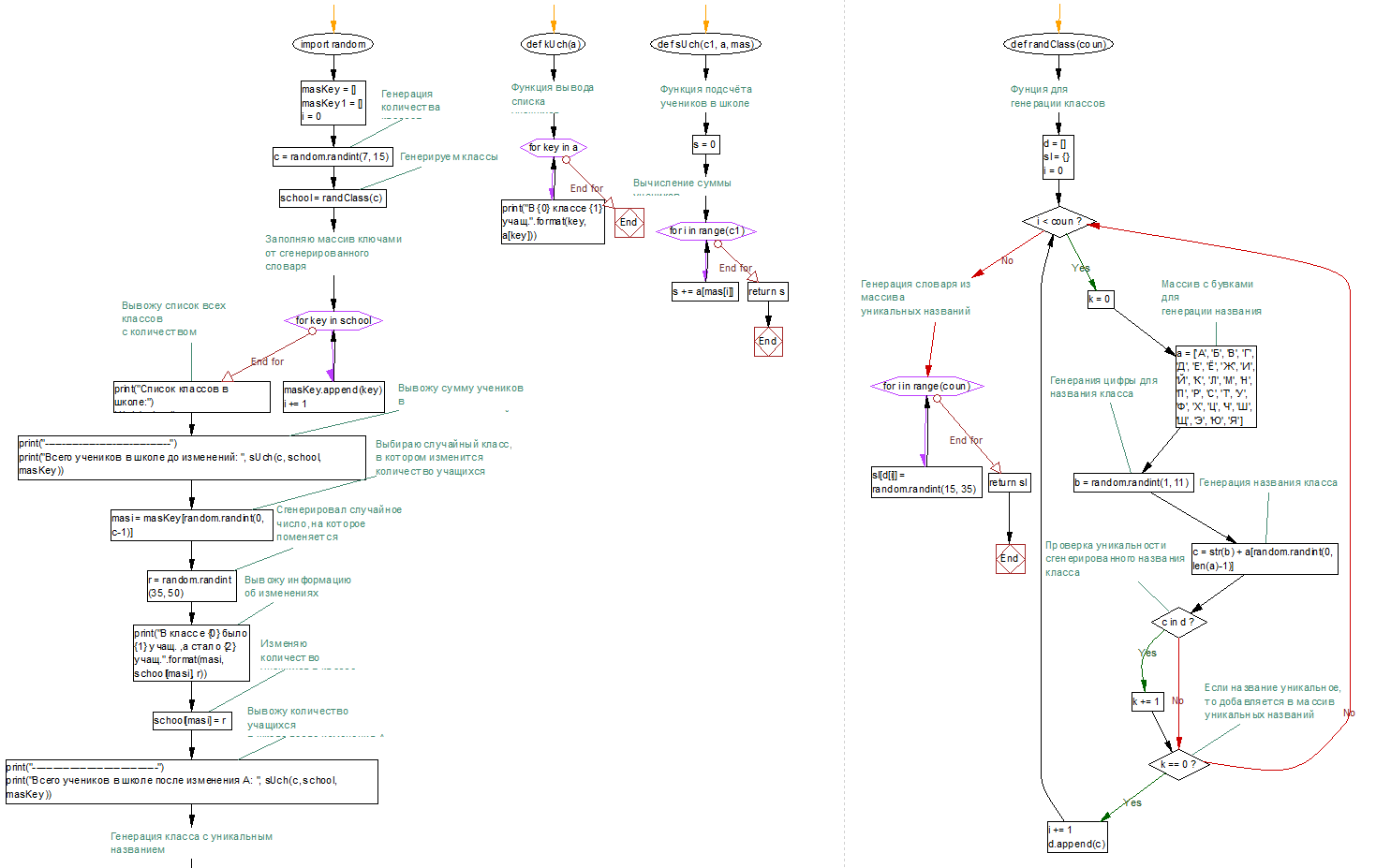


Рисунок -Блок-схема (1)

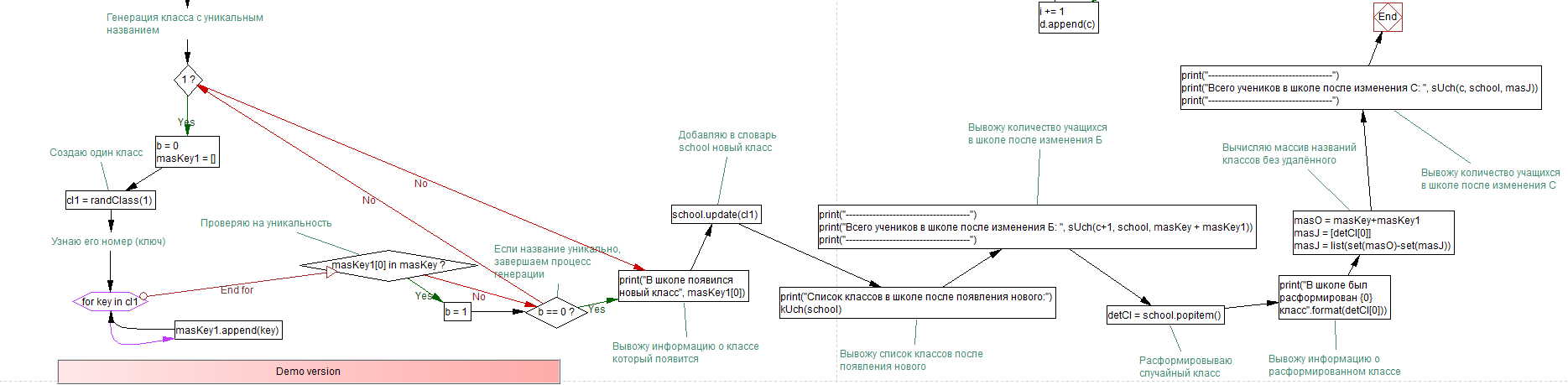


Рисунок - Блок-схема (2)

**Задание 6**

Создайте словарь с количеством элементов не менее 5-ти. Поменяйте местами первый и последний элемент объекта. Удалите второй элемент. Добавьте в конец ключ «new\_key» со значением «new\_value». Выведите на печать итоговый словарь. Важно, чтобы словарь остался тем же.

**Решение:**

# Лабораторная работа № 3.Задание №6.Вариант 10

dct = {1: "Первый", 2: "Второй", 3: "Третий", 4: "Четвёртый", 5: "Пятый"}

mas = []

# Состовляю список объектов

for i in dct.items():

mas.append(list(i))

# В созданом списке объектов меняю первый с последним

mas[-1], mas[0] = mas[0], mas[-1]

# Чищу словарь

dct.clear()

# Переношу список в словарь

for i in range(len(mas)):

dct[mas[i][0]] = mas[i][1]

# Удаляю всторой элимент

del dct[mas[1][0]]

# Добавляю новый элимент

dct["new\_key"] = "new\_value"

# Вывожу словарь на печать

print(dct)

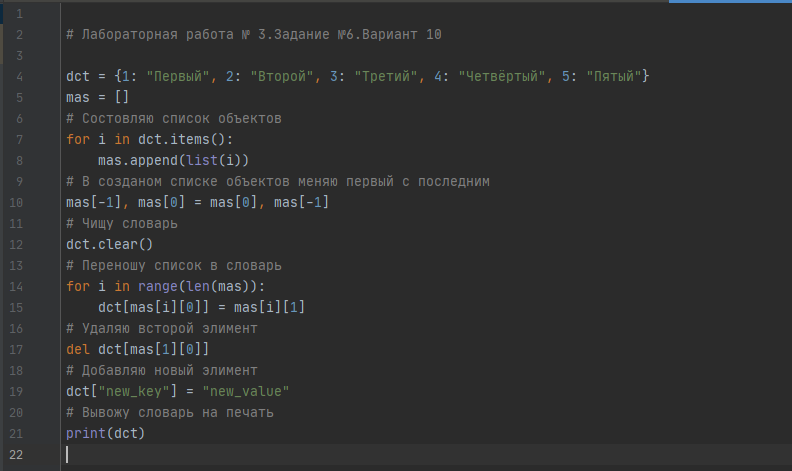


Рисунок -Программный код



Рисунок -Вывод

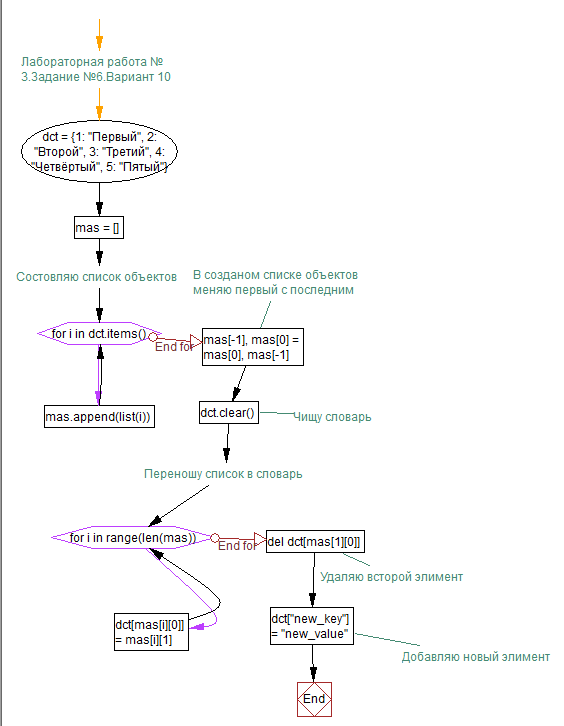
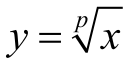
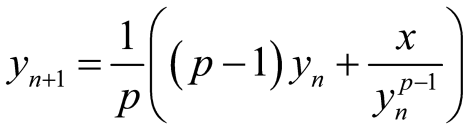
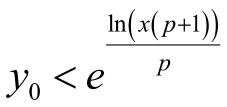


Рисунок -Блок-схема

**Задание 7**

Написать программу вычисление корня р-й степени (степень вводиться с клавиатуры) в рамках итерационной процедуры . Для определения используется итерационная процедура на основе формулы Ньютона , https://lh6.googleusercontent.com/e7ONCV0J1WUxZUPUQQToHjHvKGDNZu8UW8nHjDWyDabyNw2uukln96LpEgjVMyuFrDOy3VTfQfxA8b8SiysB9s49S2hoXgc_j4_RzVWQOy6IA2poeExeFd8j1GkMHGvFfWPBOuf2y19O2ZxAfQ, при этом . Остановка итерационной процедуры https://lh3.googleusercontent.com/9Z02H-PqQ-g2fm1lRAp_aAoiDkPftXXUROoD4J0rqeNT38-tfskOr9YkLqt77FAvVJ-RRyRi_dbxrZAGTglZbJHFACZHNQyDVMcmeBExoIUOaT-2JoG5AuI7pe0XzBz9Ikaf0gnZ74kn7pcOrQ, где https://lh5.googleusercontent.com/oTC3-NnQD1bc613JASUyHFSWOAqKEYVDyk2gKx89rzXahqijvG5bBCs9574PGZDAONEZ6gN1AcGcKenNb1WBkZ763oXE0cuRQGf9iy1IJe7VFl2v4cVZezq11udaWy3nkJmefNkYK6ofXTSOfQ – точность вычисления. В рамках программы определить число итераций, которые потребовались для отыскания корня р-й степени в рамках цикла с параметром для точности от 10-2 до 10-6, шаг 10-1. Организовать форматированный вывод результатов в виде:

Точность Корень Число итераций.

**Решение:**

# Лабораторная работа №3.Задание №7.Вариант 10.

# Вычисляем (Yn)^p-1 (Алгоритм быстрого возведения в степень)

def func(x1, n1):

# При n = 0, возвращаем 1

if n1 == 0:

return 1

# При нечётном n, понижаем степень на 1

elif n1 % 2 == 1:

return func(x1, n1-1) \* x1

else:

# При чётном n, делим степень на 2

x2 = func(x1, n1 // 2)

return x2\*\*2

# Функция вычисления корня и количества итераций

def krP(x, p, chag):

# Первая итерация в особом порядке

x1 = x

# Вычисление на основе формулы Ньютона

Y = (1/p)\*((p-1)\*x1 + (x / func(x1, p-1)))

n = 1

# Пока точность вычисления больше, либо равна |Yn+1 - Yn|, то высисляю

while abs(Y - x1) > chag:

x1 = Y

# Вычисление на основе формулы Ньютона

Y = (1/p)\*((p-1)\*x1 + (x / func(x1, p-1)))

# Cчетчик итераций

n += 1

return x1, n

# Ввод данных

x = float(input("Введите число X: "))

p = int(input("Введите степень P: "))

# Шаг 10^-1 или 0.1

chag = 0.1

# Параметр для точности от 10^(-2) до 10^(-6). Всего 5 итераций.

for i in range(5):

# Изменение шага

chag /= 10

# Отыскание корня P-й степени цикла с параметром для точности

res, n = krP(x, p, chag)

# Вывод результата точность, корень, число итераций

print("Точность:{:f} Корень:{:f} Числ.итераций: {}".format(chag, res, n))

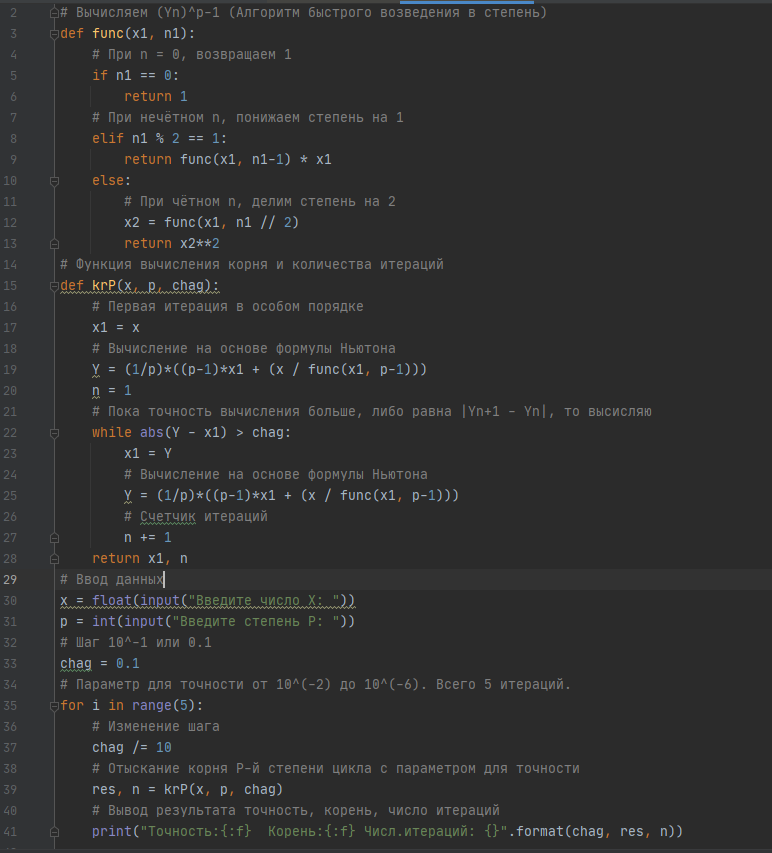


Рисунок -Программный код

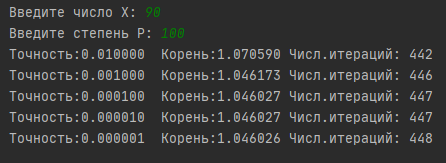


Рисунок -Ввод-вывод

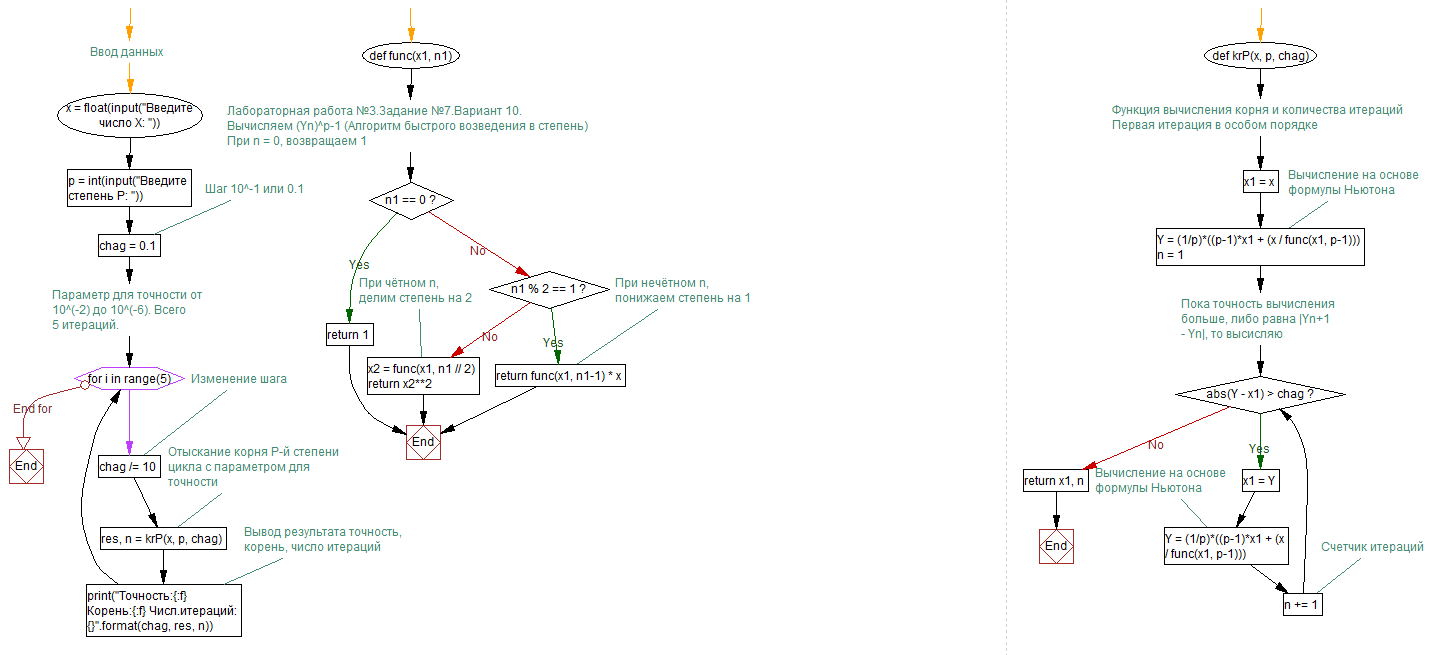


Рисунок -Блок-схема