**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Российский Университет Транспорта (МИИТ)»**

**Институт экономики и финансов**

**Кафедра «Информационные системы цифровой экономики»**

**О Т Ч Е Т**

**по дисциплине: «Информатика»**

**о выполнении практических работ**

**на тему «Программирование в Python»**

**Выполнила: студентка ЭБИ-113**

**Эрлингас И. Д.**

.

**Проверила**: **ст. преп. Дмитриева Т.М.**

**Москва 2020 г.**

**Содержание**

[Условные конструкции. 3](#_Toc60071435)

[Пример 1. 3](#_Toc60071436)

[Пример 2. 4](#_Toc60071437)

[Пример 3. 5](#_Toc60071438)

[Пример 4. 8](#_Toc60071439)

[Строки. 11](#_Toc60071440)

[Пример 1. 11](#_Toc60071441)

[Списки. 14](#_Toc60071442)

[Модуль random 14](#_Toc60071443)

[Пример 1. 14](#_Toc60071444)

[Пример 2. 18](#_Toc60071445)

[Пример 3. 18](#_Toc60071446)

[Пример 4. 19](#_Toc60071447)

[Исключения 20](#_Toc60071448)

[Пример 1. 20](#_Toc60071449)

[Словари 21](#_Toc60071450)

[Пример 1. 21](#_Toc60071451)

[Файлы 22](#_Toc60071452)

[Пример 1. 22](#_Toc60071453)

[ЧЕРЕПАШЬЯ ГРАФИКА 23](#_Toc60071454)

[Пример 1. 23](#_Toc60071455)

# Условные конструкции.

## Пример 1.

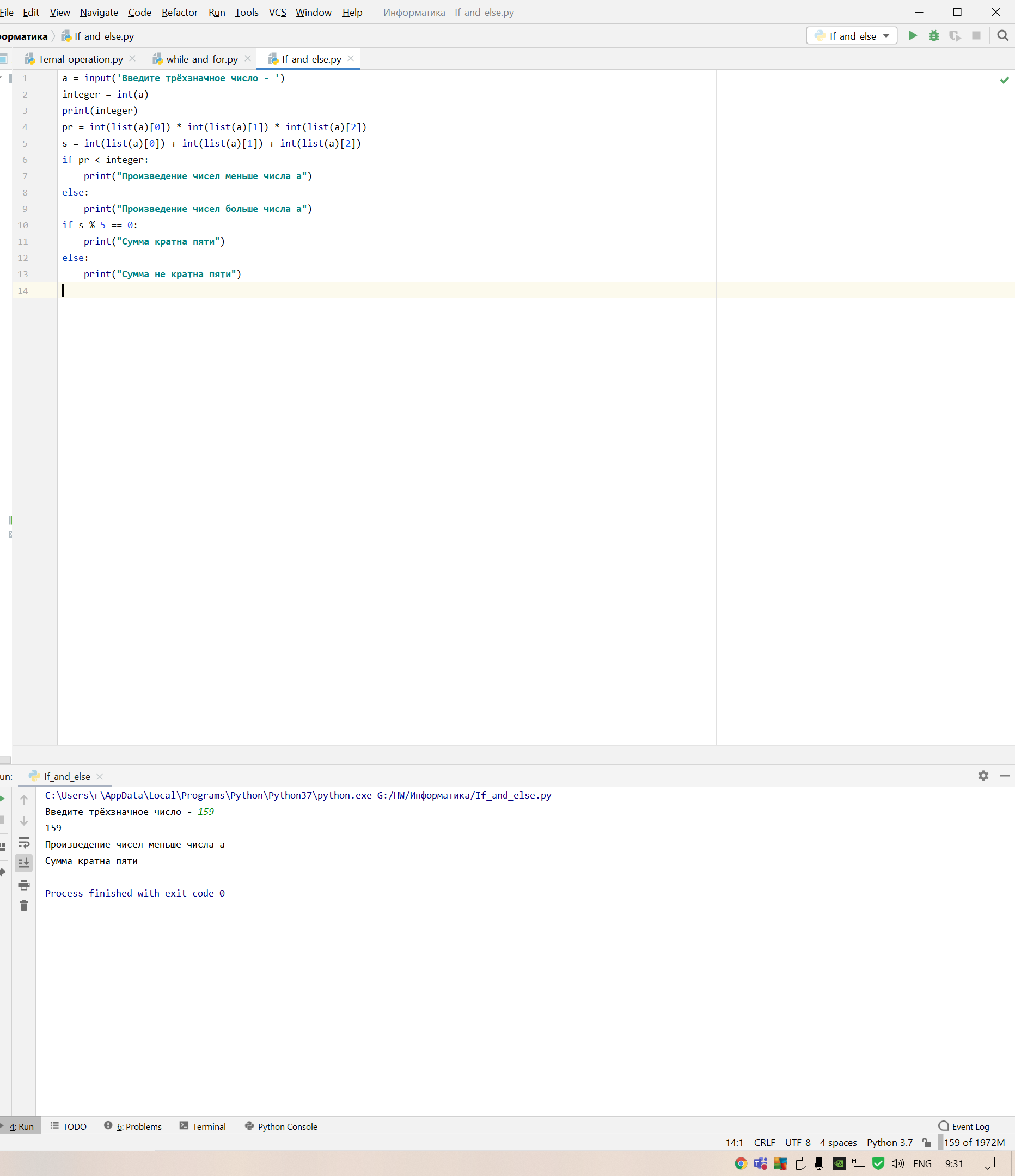
Постановка задачи:

Дано трёхзначное число a. Определить: a) Является ли произведение его цифр меньше числа а; б) кратна ли 5 сумма его цифр

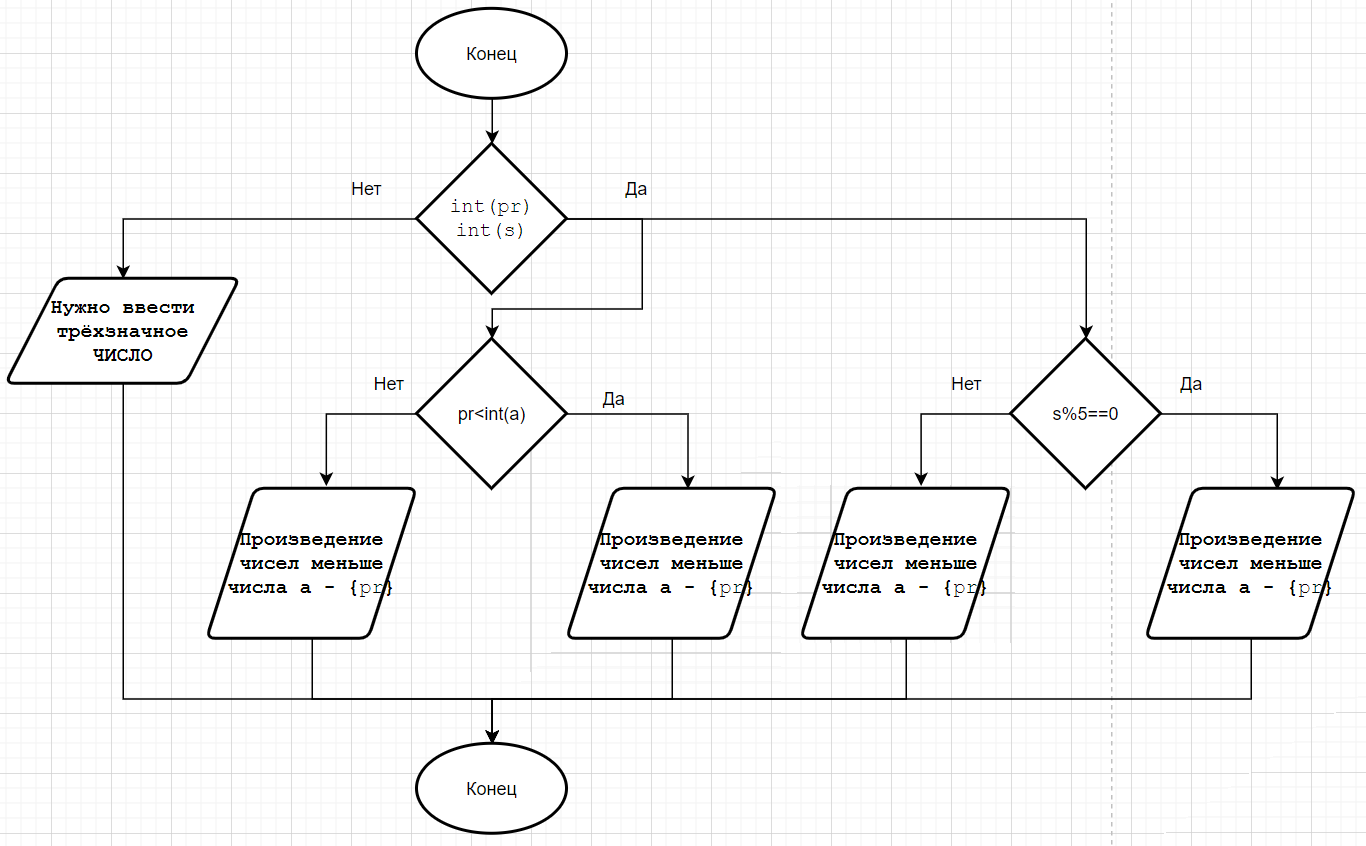
Текст программы:

a = input(**'Введите трёхзначное число - '**)  
**try**:  
 pr = int(list(a)[0]) \* int(list(a)[1]) \* int(list(a)[2])  
 s = int(list(a)[0]) + int(list(a)[1]) + int(list(a)[2])  
 **if** pr < int(a):  
 print(**f"Произведение чисел меньше числа а - {**pr**}"**)  
 **else**:  
 print(**f"Произведение чисел больше числа а - {**pr**}"**)  
 **if** s % 5 == 0:  
 print(**f'Сумма кратна пяти - {**s**}'**)  
 **else**:  
 print(**f'Сумма не кратна пяти - {**s**}'**)  
**except** Exception **as** error:  
 print(**'Нужно ввести трёхзначное ЧИСЛО'**)

Результаты выполнения:



Блок-схема:



## Пример 2.

Постановка задачи:

Ввести с клавиатуры номер месяца. Определить сезон в зависимости от номера месяца и вывести сообщение (весна (3, 4, 5), лето (6, 7, 8), осень (9, 10, 11), зима (12, 1, 2)).

Текст программы:

number = int(input('Введите номер месяца - '))

if number == 3 or 4 or 5:

print('Весна')

elif number == 6 or 7 or 8:

print('Лето')

elif number == 9 or 10 or 11:

print('Осень')

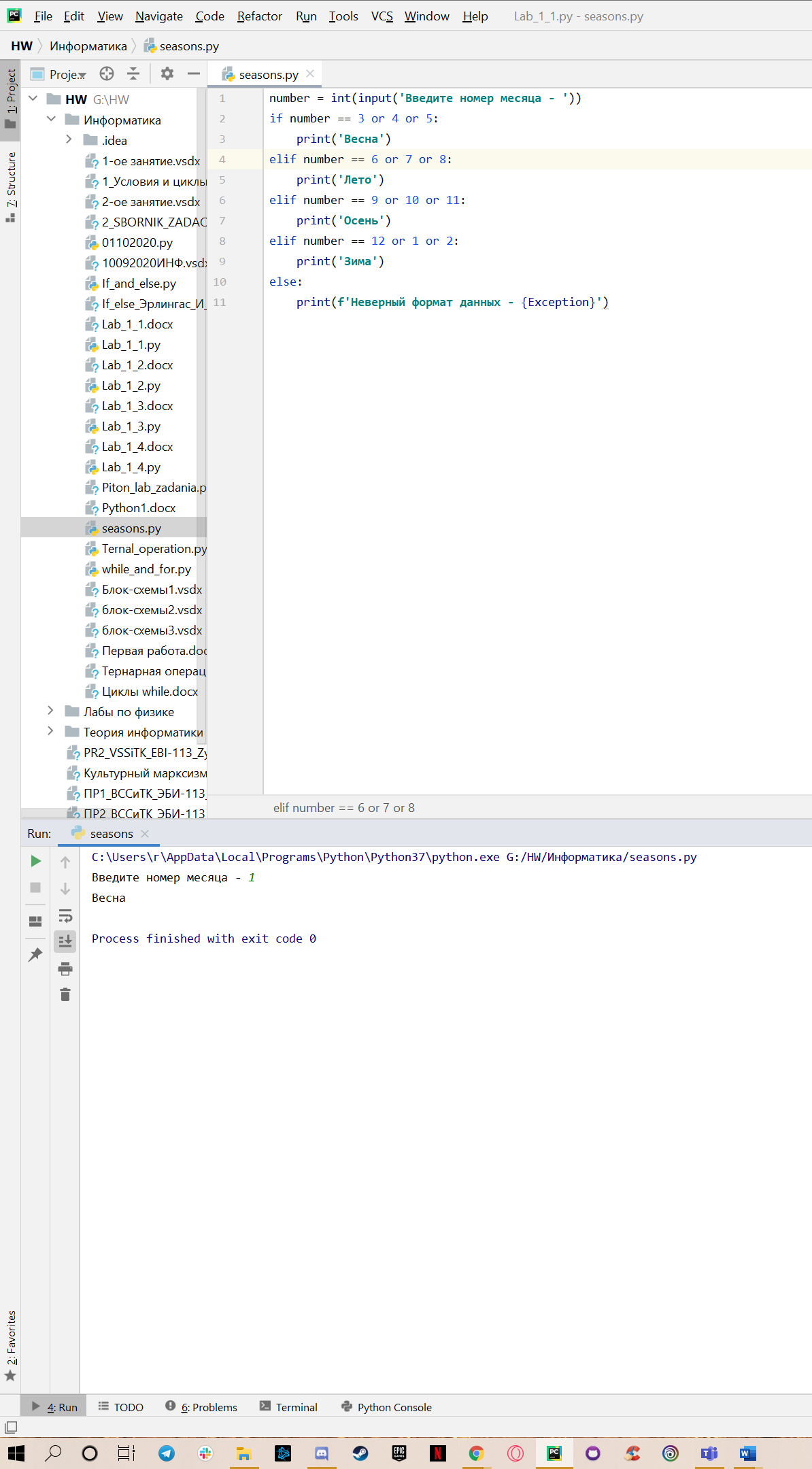
elif number == 12 or 1 or 2:

print('Зима')

else:

print (f'Неверный формат данных - {Exception}')

Результаты выполнения:



## Пример 3.

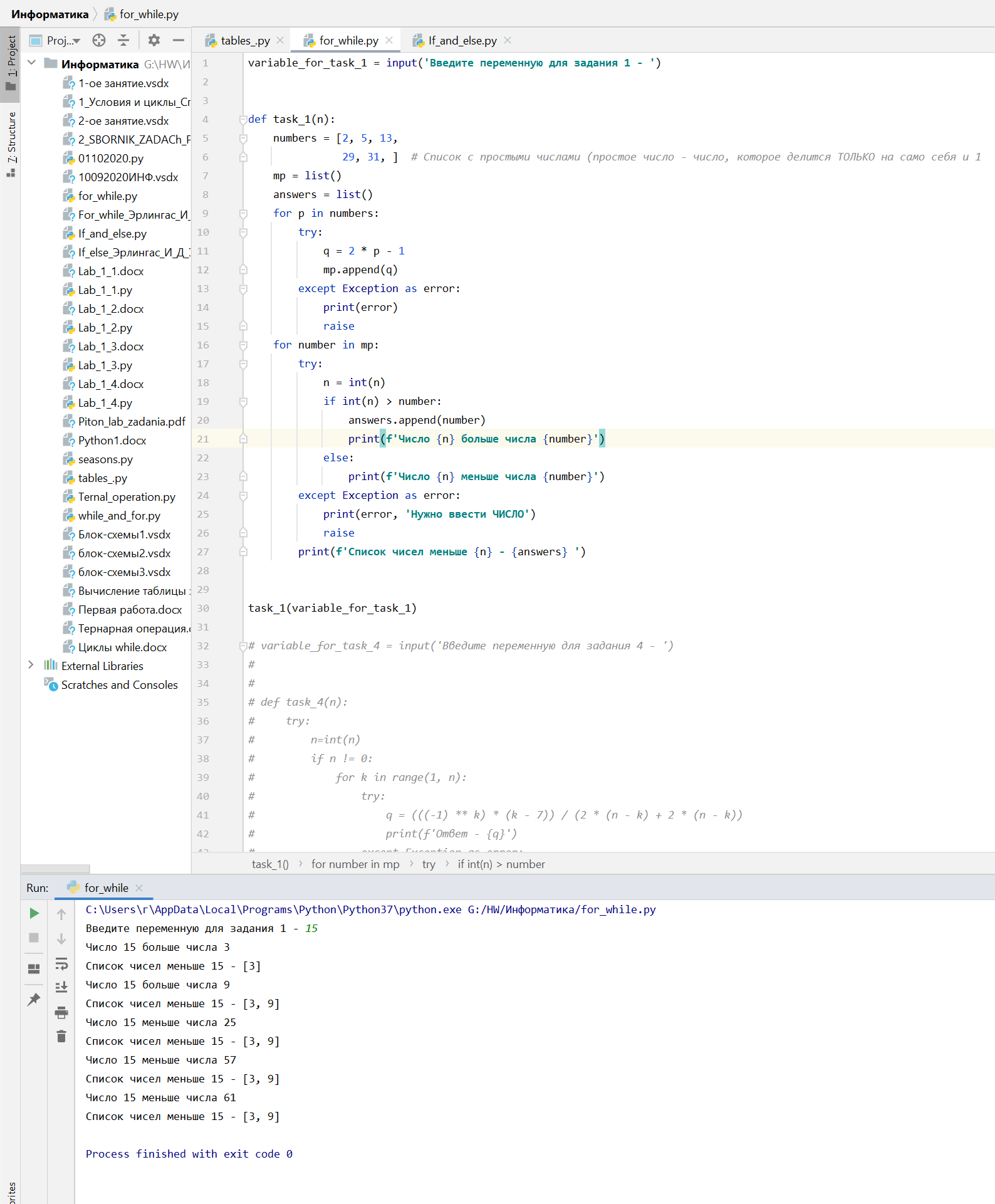
Постановка задачи:

Дано натуральное число n. Найти все числа меньшие Mp числа Мерсенна. Число Мерсенна – это простое число, представленное в виде Mp = 2p – 1, где p – тоже простое число

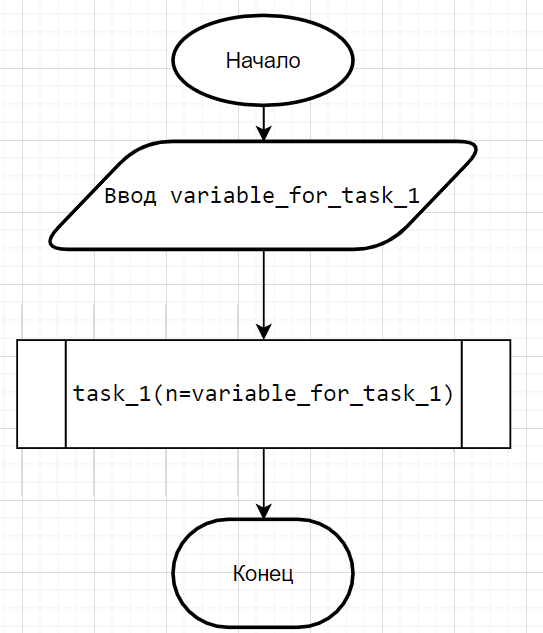
Текст программы:

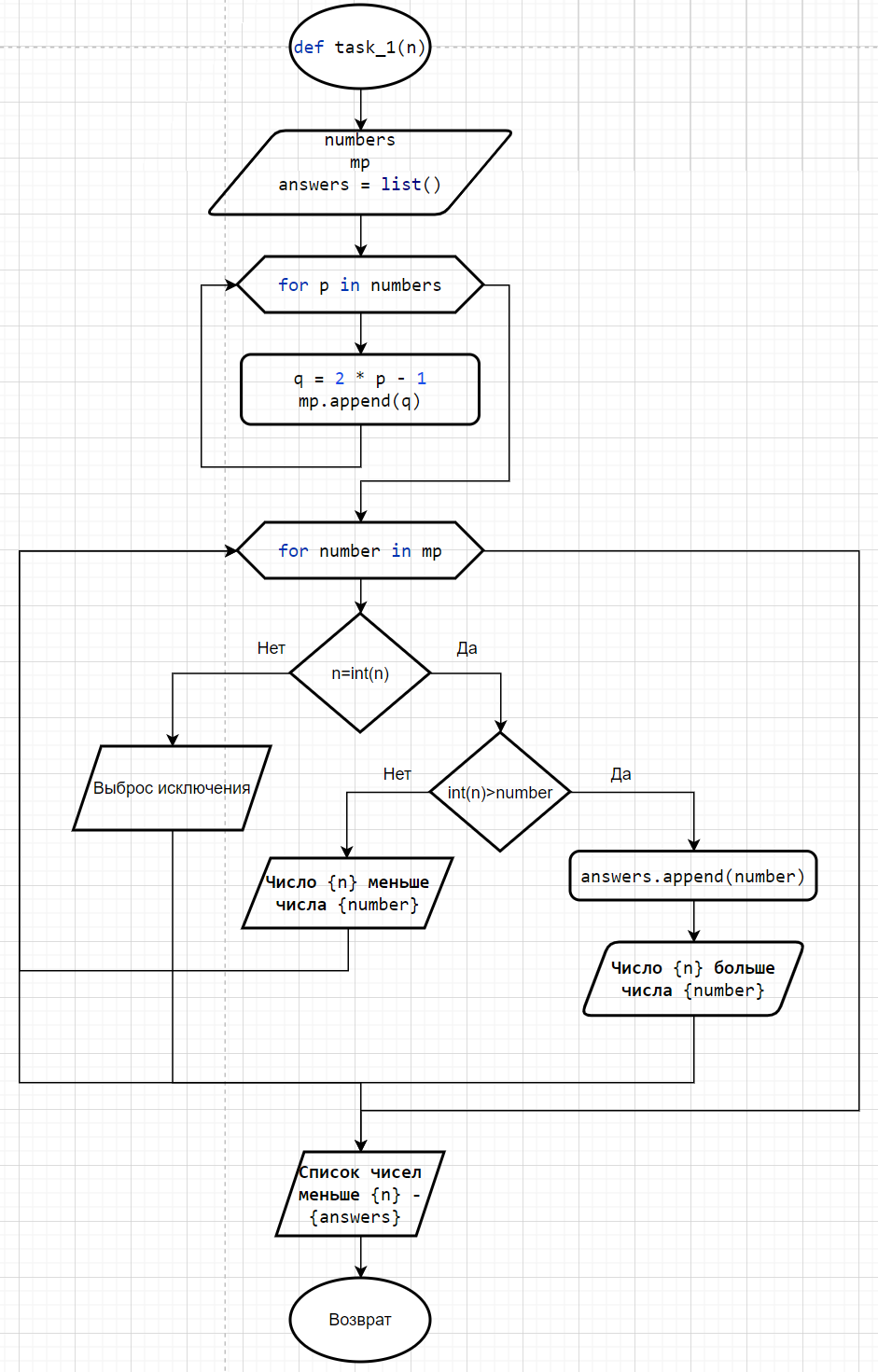
variable\_for\_task\_1 = input(**'Введите переменную для задания 1 - '**)  
  
  
def task\_1(n):  
 numbers = [2, 5, 13,  
 29, 31, ] *# Список с простыми числами (простое число - число, которое делится ТОЛЬКО на само себя и 1* mp = list()  
 answers = list()  
 for p in numbers:   
 q = 2 \* p - 1  
 mp.append(q)   
 for number in mp:  
 try:  
 n = int(n)  
 if int(n) > number:  
 answers.append(number)  
 print(**f'Число** {n} **больше числа** {number}**'**)  
 else:  
 print(**f'Число** {n} **меньше числа** {number}**'**)  
 except Exception as error:  
 print(error, **'Нужно ввести ЧИСЛО'**)  
 raise  
 print(**f'Список чисел меньше** {n} **-** {answers} **'**)  
  
  
task\_1(variable\_for\_task\_1)

Результаты выполнения:



Блок-схема:





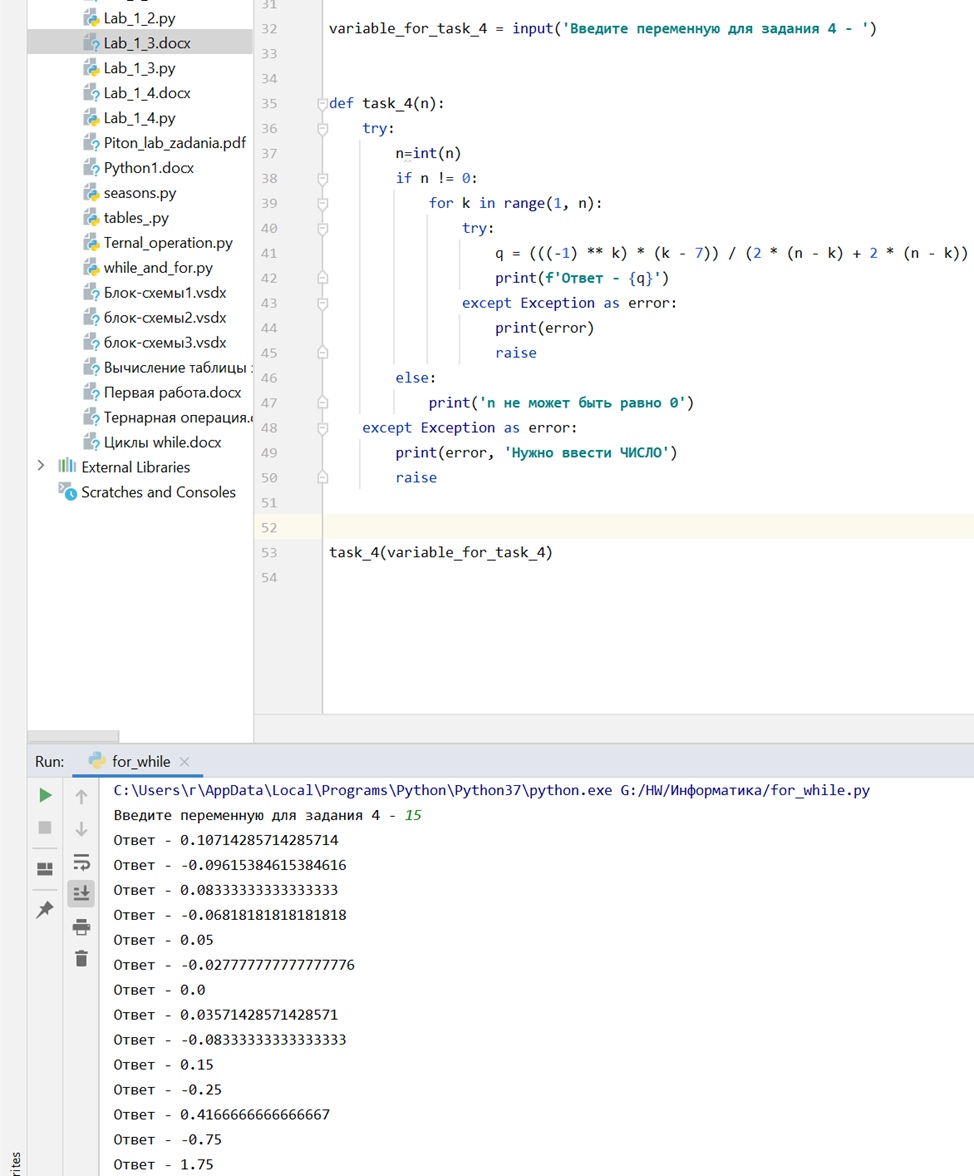
## Пример 4.

Постановка задачи:

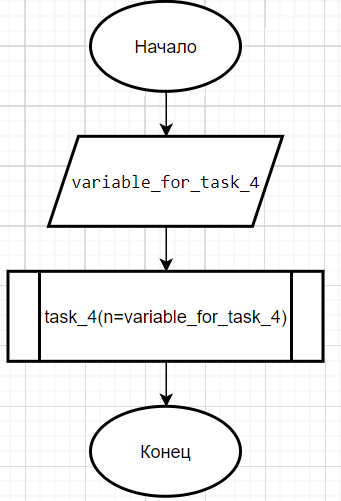
Текст программы:

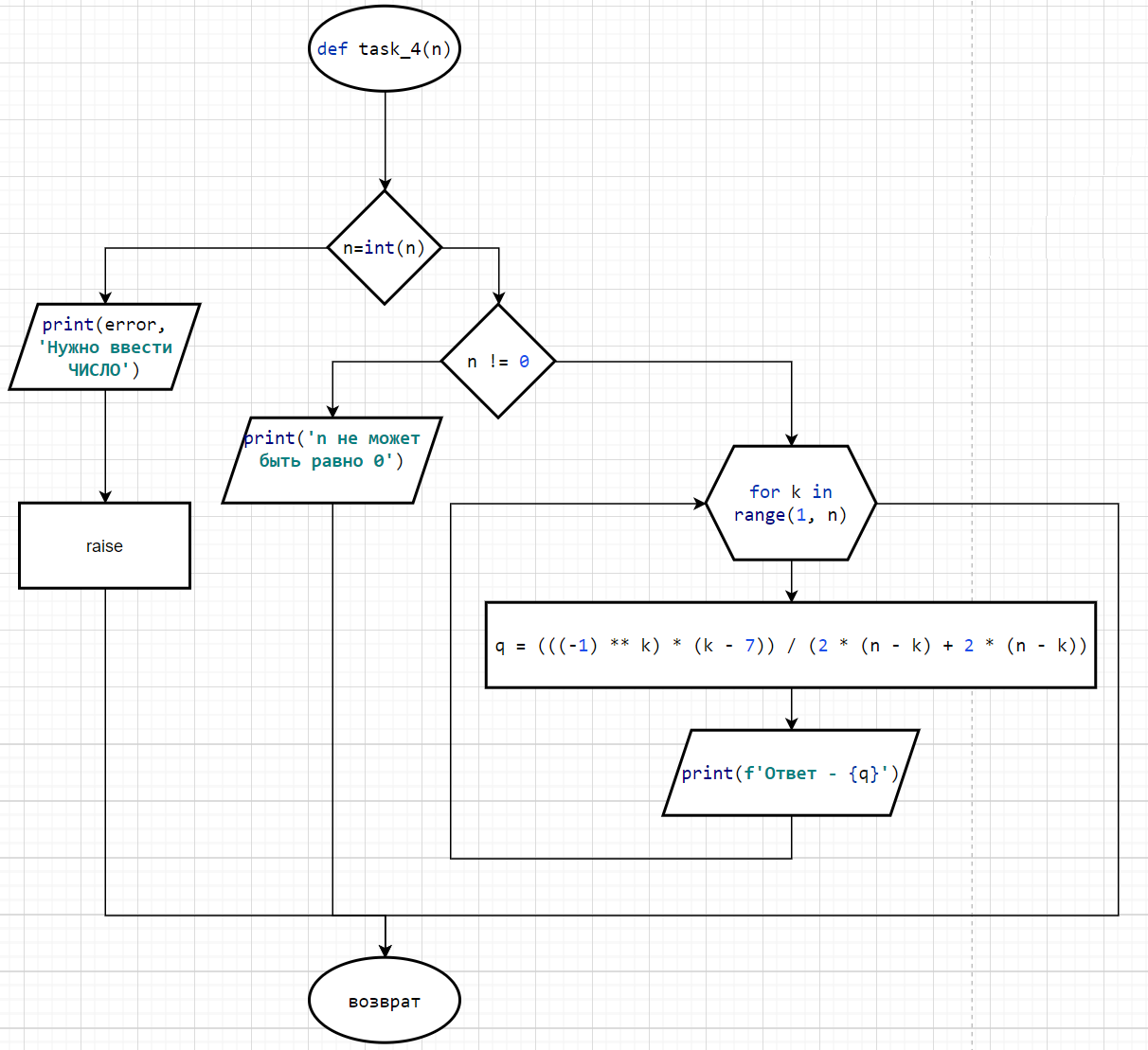
variable\_for\_task\_4 = input(**'Введите переменную для задания 4 - '**)  
  
  
def task\_4(n):  
 try:  
 n=int(n)  
 if n != 0:  
 for k in range(1, n):   
 q = (((-1) \*\* k) \* (k - 7)) / (2 \* (n - k) + 2 \* (n - k))  
 print(**f'Ответ -** {q}**'**)   
 else:  
 print(**'n не может быть равно 0'**)  
 except Exception as error:  
 print(error, **'Нужно ввести ЧИСЛО'**)  
 raise  
task\_4(variable\_for\_task\_4)

Результаты выполнения:



Блок-схема:





# Строки.

## Пример 1.

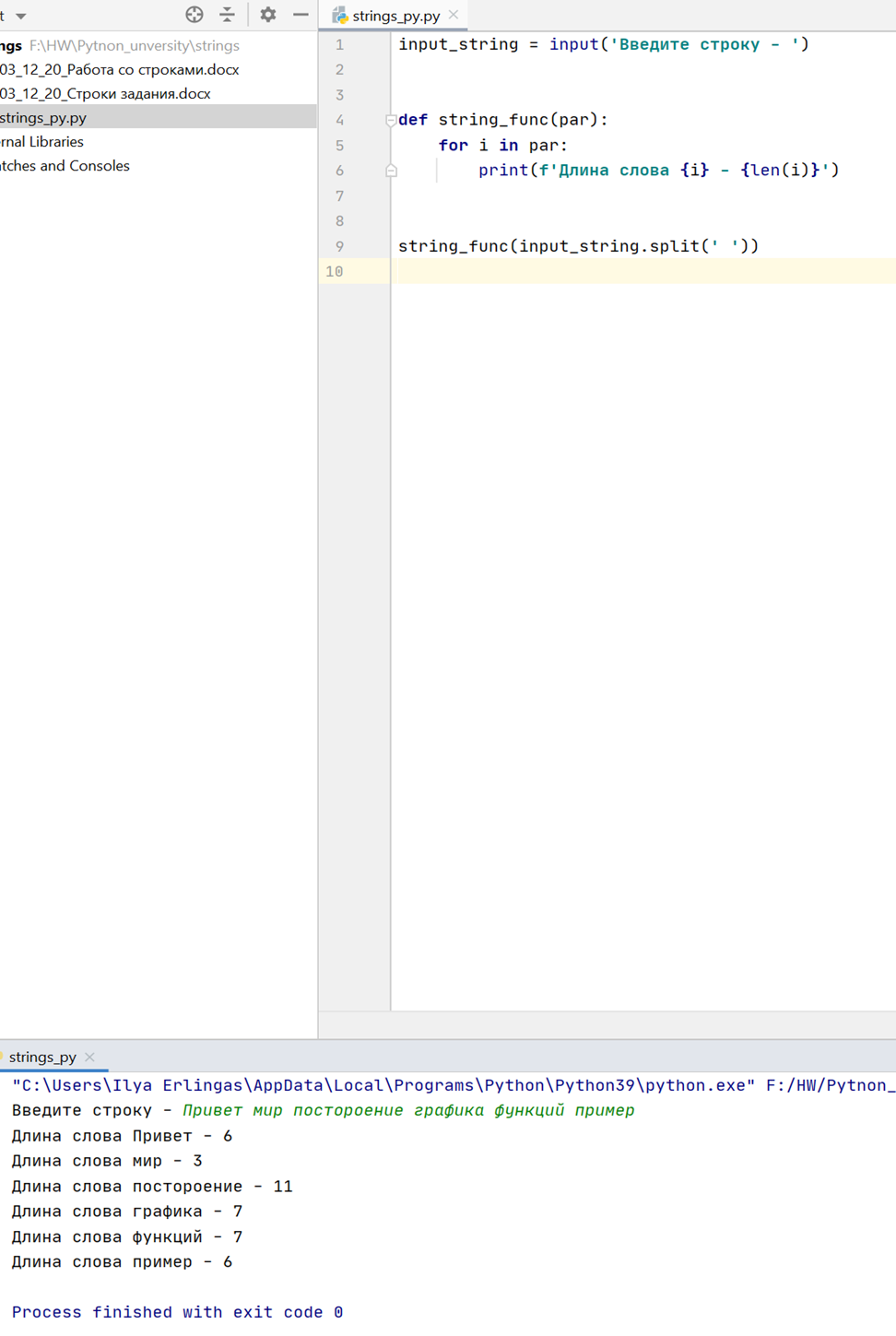
Постановка задачи:

В строке после каждого слова дописать его длину.

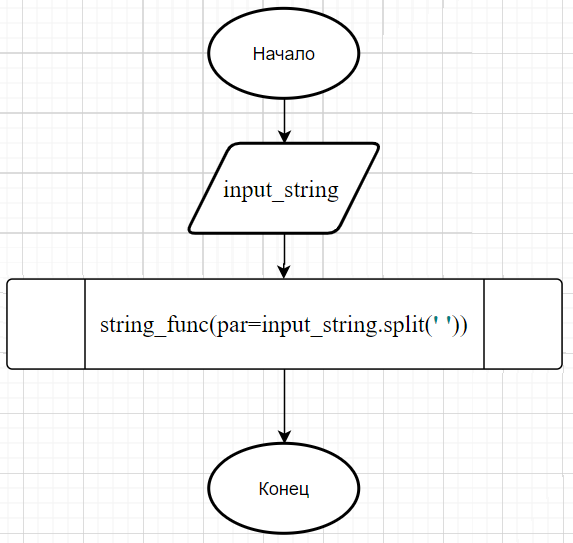
Текст программы:

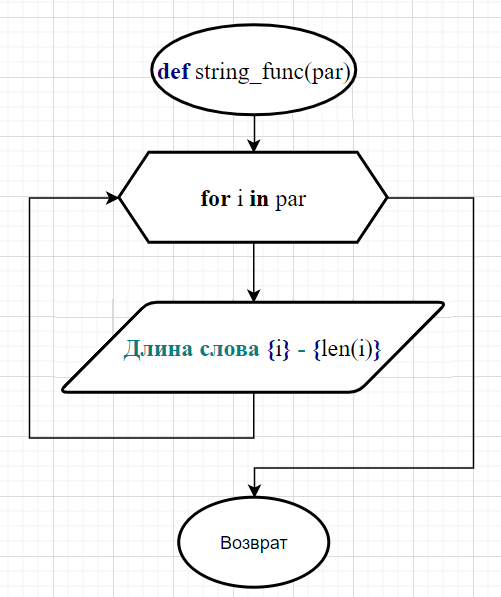
input\_string = input(**'Введите строку - '**)  
  
  
**def** string\_func(par):  
 **for** i **in** par:  
 print(**f'Длина слова {**i**} - {**len(i)**}'**)  
  
  
string\_func(input\_string.split(**' '**))

Результаты выполнения:



Блок-схема:





# Списки.

## Модуль random

Random а в частности randint позволяет генерировать целое число, например, как следующей задаче – от 1 до 99.

## Пример 1.

Постановка задачи:

Сортировка списка методом пузырька.

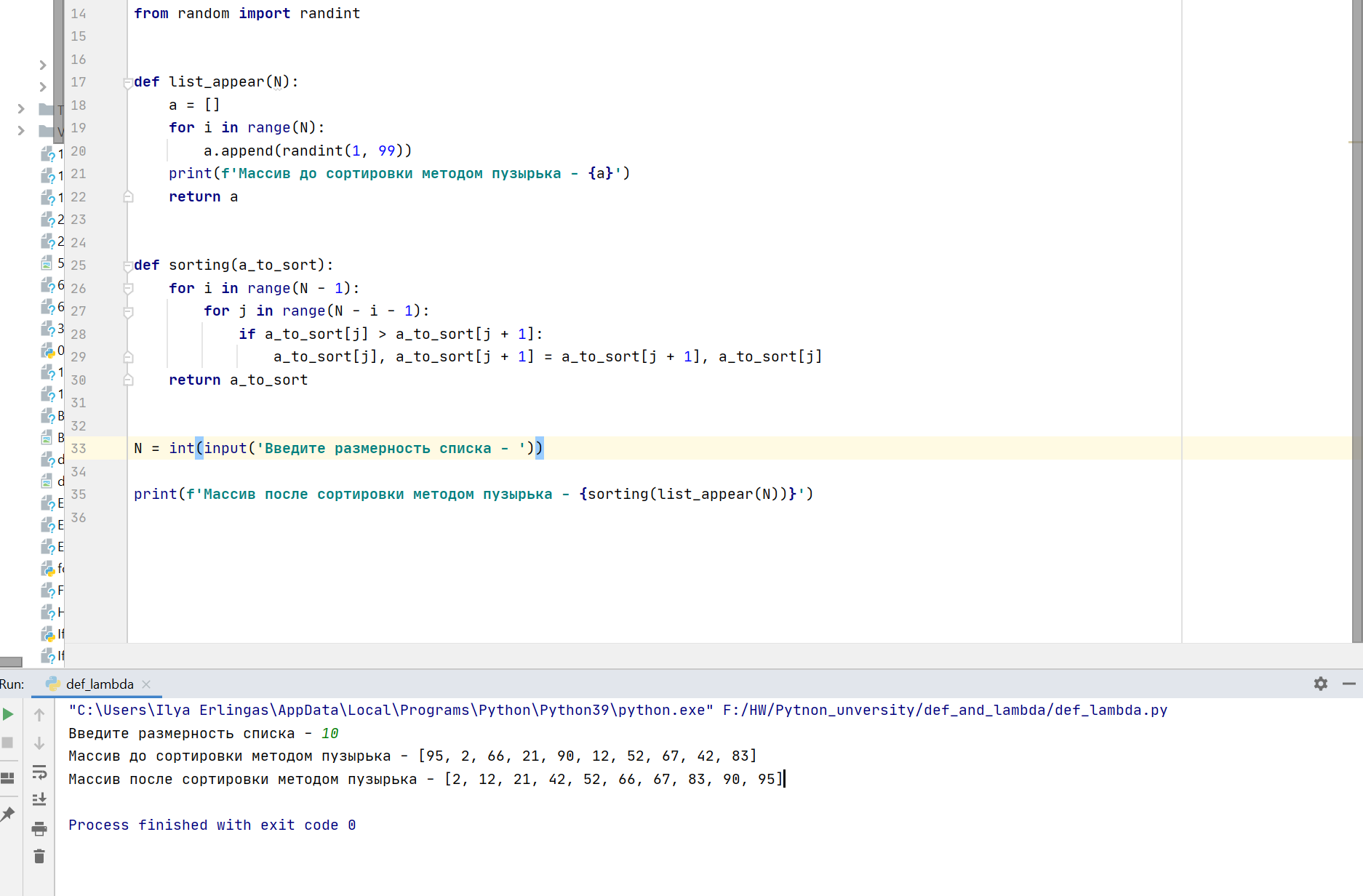
Текст программы:

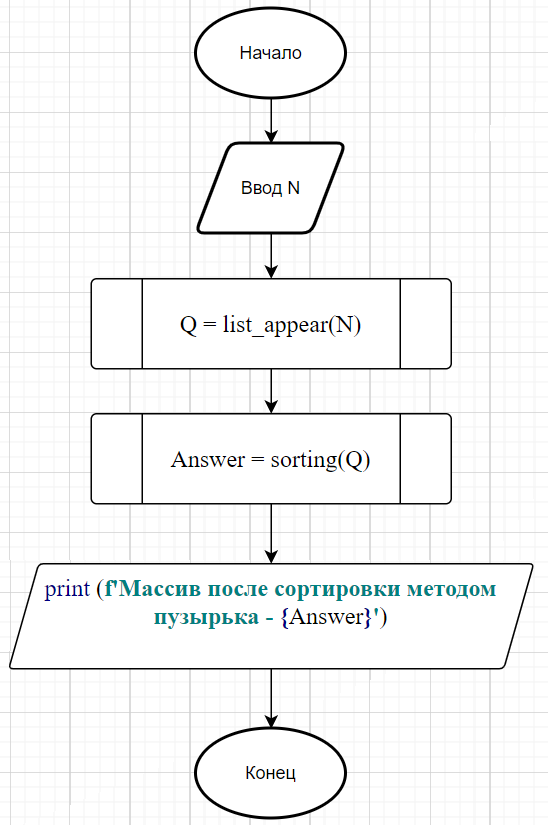
**from** random **import** randint  
  
  
**def** list\_appear(N):  
 a = []  
 **for** i **in** range(N):  
 a.append(randint(1, 99))  
 print(**f'Массив до сортировки методом пузырька - {**a**}'**)  
 **return** a  
  
  
**def** sorting(a\_to\_sort):  
 **for** i **in** range(N - 1):  
 **for** j **in** range(N - i - 1):  
 **if** a\_to\_sort[j] > a\_to\_sort[j + 1]:  
 a\_to\_sort[j], a\_to\_sort[j + 1] = a\_to\_sort[j + 1], a\_to\_sort[j]  
 **return** a\_to\_sort  
  
  
N = int(input(**'Введите размерность списка - '**)

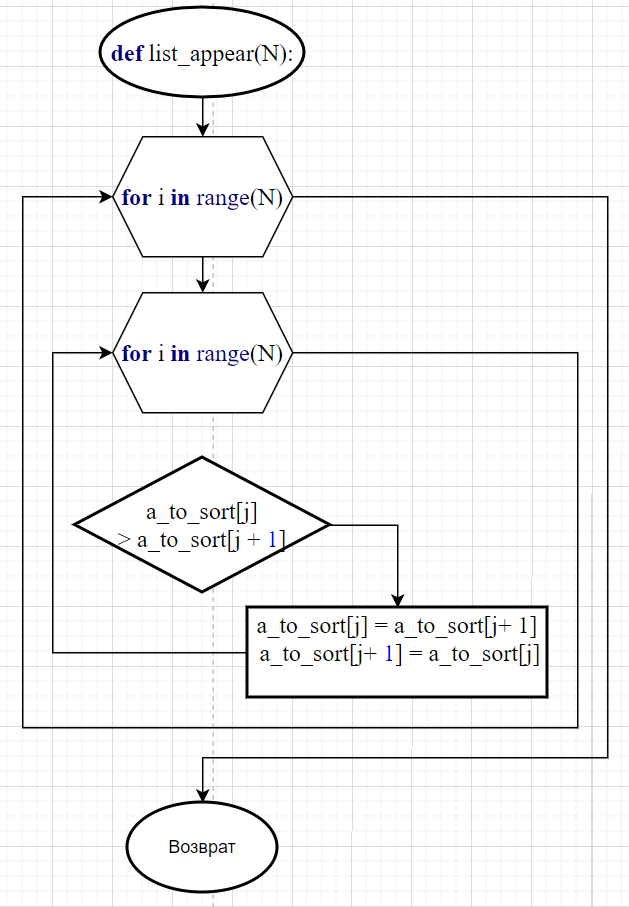
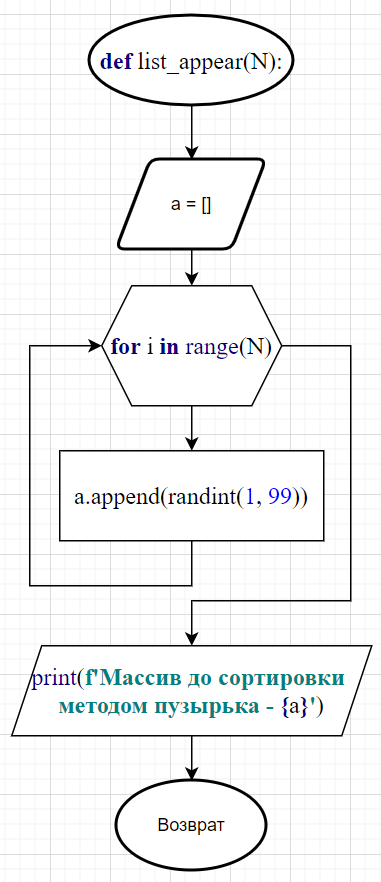
Q = list\_appear(N)

Answer = sorting(Q)  
  
print (**f'Массив после сортировки методом пузырька - {**Answer**}'**)

Результаты выполнения:



Блок-схема 1



## Пример 2.

Постановка задачи:

Задан целочисленный массив. Определить остаток от деления суммы элементов с чётными индексами на сумму элементов с нечётными индексами.

Текст программы:

whole\_integer = [1, 2, 3, 4, 5,

6, 7, 8, 9, 10,

11, 12, 13, 14, 15,

16, 17, 299, 78]

i\_counter = 0

sum\_odd = 0

sum\_even = 0

for i in whole\_integer:

i\_counter += 1

try:

if i\_counter % 2 == 0:

sum\_odd += whole\_integer[i\_counter]

else:

sum\_even += whole\_integer[i\_counter]

except IndexError as error:

pass

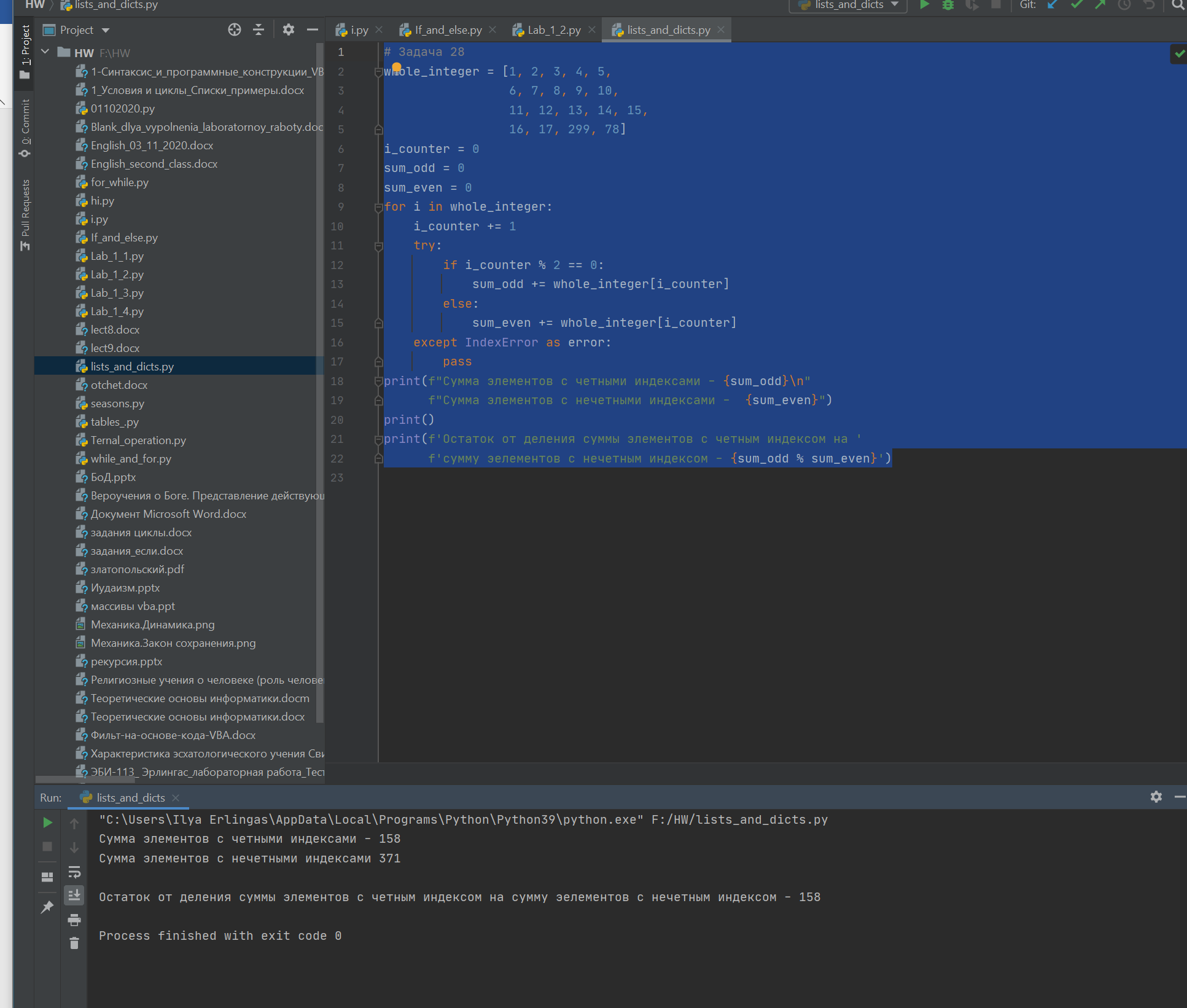
print(f"Сумма элементов с четными индексами - {sum\_odd}\n"

f"Сумма элементов с нечетными индексами - {sum\_even}")

print()

print(f'Остаток от деления суммы элементов с четным индексом на '

f'сумму эелементов с нечетным индексом - {sum\_odd % sum\_even}')



## Пример 3.

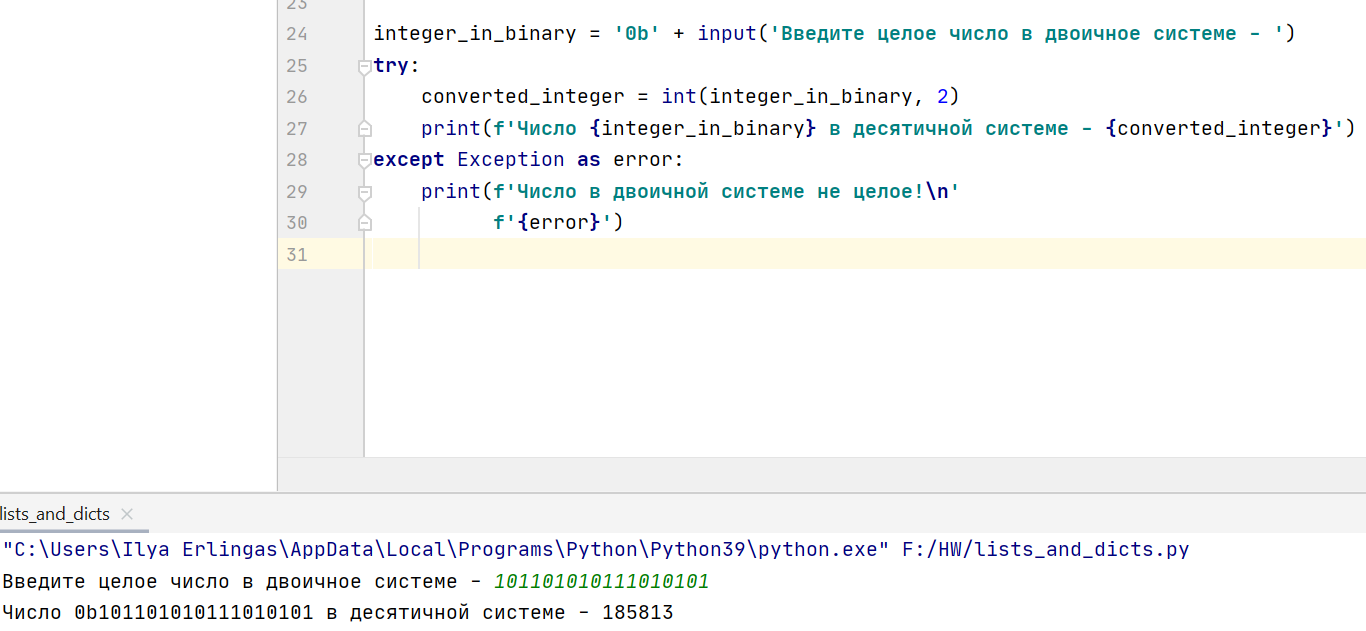
Постановка задачи:

Дано целое число в двоичной системе счисления, т.е. последовательность цифр 0 и 1. Составить программу перевода числа в десятичную систему счисления.

Текст программы:

integer\_in\_binary = **'0b'** + input(**'Введите целое число в двоичное системе - '**)  
**try**:  
 converted\_integer = int(integer\_in\_binary, 2)  
 print(**f'Число {**integer\_in\_binary**} в десятичной системе - {**converted\_integer**}'**)  
**except** Exception **as** error:  
 print(**f'Число в двоичной системе не целое!\n'  
 f'{**error**}'**)

Результаты выполнения:



## Пример 4.

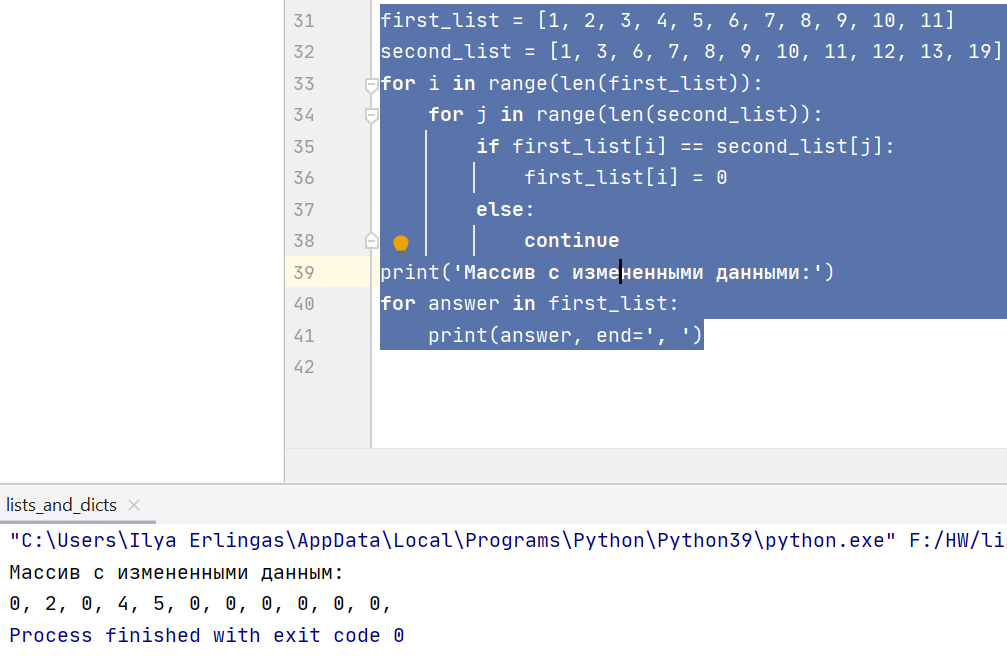
Постановка задачи:

Даны два массива действительных чисел по 12 элементов в каждом. Заменить нулями те элементы первого массива, которые есть во втором.

Текст программы:

first\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]  
second\_list = [1, 3, 6, 7, 8, 9, 10, 11, 12, 13, 19]  
**for** i **in** range(len(first\_list)):  
 **for** j **in** range(len(second\_list)):  
 **if** first\_list[i] == second\_list[j]:  
 first\_list[i] = 0  
 **else**:  
 **continue**print(**'Массив с измененными данными:'**)  
**for** answer **in** first\_list:  
 print(answer, end=**', '**)

Результаты выполнения:



# Исключения

Исключения позволяют обработать разнообразные ошибки, возникшие в коде – при вводе или каких-либо других ошибках.

## Пример 1.

Постановка задачи:

Реализация «Польской нотации» - знак вычисления всегда идёт перед операндами

Текст программы:

user\_input = input(**"Введите комманду и числа оперции: "**)  
**"""Ввод пользователем поочередно: сначала функция => потом два числа"""**input\_list = user\_input.split(**" "**)  
**"""Разбитие ввода на части через пробел"""  
  
  
def** poland\_():  
 **try**:  
 **"""Проверка условий ввода и приравнивание к integers чисел"""** mark = (input\_list[0])  
 a = int(input\_list[1])  
 b = int(input\_list[2])  
 result = 0  
 **if** mark == **"+"**:  
 result = a + b  
 **elif** mark == **"/"**:  
 result = a / b  
 **elif** mark == **"-"**:  
 result = a - b  
 **elif** mark == **"\*"**:  
 result = a \* b  
 **else**:  
 result = **'Переменная с результатом пуста'** print(result)  
 **"""Проверка на ошибки ввода - /0, ввод букв вместо чисел, недостаточное количество чисел"""  
 except** ZeroDivisionError **as** Z:  
 print(Z, type(Z))  
 print(**"Делить на ноль нельзя!"**)  
 **except** ValueError **as** V:  
 print(V, type(V))  
 print(**"Неверный ввод данных"**)  
 **except** IndexError **as** Index:  
 print(Index, type(Index))  
 print(**"Маловато операндов"**)  
  
result = poland\_()  
print(result)

Результаты выполнения:



# Словари

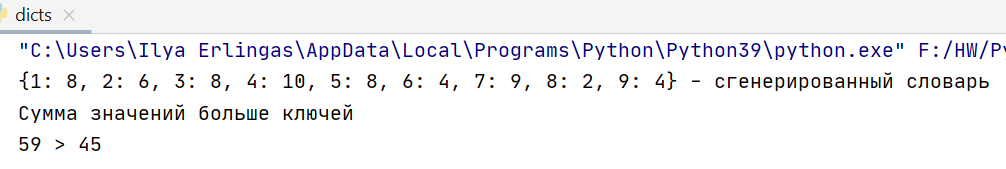
## Пример 1.

Постановка задачи:

Сгенерировать словарь, сравнить сумму ключей и их значений.

Текст программы:

**from** random **import** randint  
  
dict\_ = {}  
  
  
**def** generating\_dict():  
 **for** i **in** range(1, 10):  
 dict\_[i] = randint(1, 10)  
 **return** dict\_  
  
  
**def** sum\_values():  
 summ = 0  
 **for** i **in** dict\_.values():  
 summ += i  
 **return** summ  
  
  
**def** summ\_keys():  
 summ = 0  
 **for** i **in** dict\_.keys():  
 summ += i  
 **return** summ  
  
  
generated\_dict, values\_summ, keys\_summ = generating\_dict(), sum\_values(), summ\_keys()  
print(**f'{**generated\_dict**} - сгенерированный словарь'**)  
**if** values\_summ > keys\_summ:  
 print(**'Сумма значений больше ключей'**)  
 print(**f'{**values\_summ**} > {**keys\_summ**}'**)  
**elif** values\_summ < keys\_summ:  
 print(**'Сумма ключей больше значений'**)  
 print(**f'{**values\_summ**} < {**keys\_summ**}'**)  
**else**:  
 print(**'Сумма ключей равна значений'**)  
 print(**f'{**values\_summ**} = {**keys\_summ**}'**)

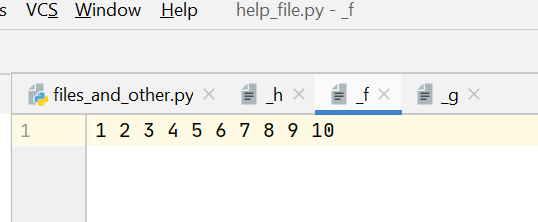


# Файлы

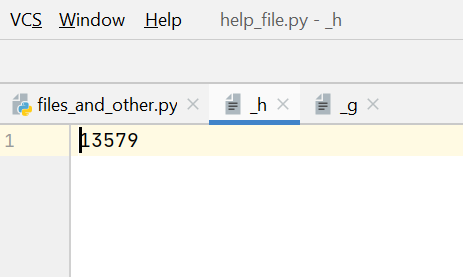
## Пример 1.

**def** even\_file(i, list\_to\_compare):  
 **with** open(**"F:\HW\Pytnon\_unversity\\_g"**, **'a'**, encoding=**'utf-8'**) **as** g:  
 g.write(list\_to\_compare[i])  
  
  
**def** not\_even\_file(i, list\_to\_compare):  
 **with** open(**"F:\HW\Pytnon\_unversity\\_h"**, **'a'**, encoding=**'utf-8'**) **as** h:  
 h.write(list\_to\_compare[i])  
  
  
**def** begin():  
 **with** open(**"F:\HW\Pytnon\_unversity\\_f"**, **'r'**, encoding=**'utf-8'**) **as** f:  
 list\_f = list(f.read().split(**' '**))  
 **for** i **in** range(len(list\_f)):  
 **if** int(list\_f[i]) % 2 == 0:  
 even\_file(i, list\_f)  
 **else**:  
 not\_even\_file(i, list\_f)  
  
  
begin()

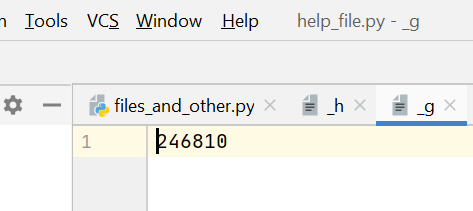
Изначальный файл с числами



Отфильтрованный файл с нечётными числами



Файл с чётными числами



# ЧЕРЕПАШЬЯ ГРАФИКА

## Пример 1.

**from** turtle **import** Turtle, Screen  
**import** math  
COLORS = [**"red"**, **"yellow"**, **"blue"**, **"brown"**, **"pink"**, **"green"**, **"black"**, **"orange"**, **"purple"**]  
  
  
**def** draw\_polygons(sides, area):  
 **for** i, sd **in** enumerate(range(sides, 2, -1)):  
 side\_length = math.sqrt(area / sd \* 4 \* math.atan(math.pi / sd))  
 a\_color = COLORS[i % len(COLORS)]  
 rest.fillcolor(a\_color)  
 rest.pendown()  
 rest.begin\_fill()  
 **for** \_ **in** range(sd):  
 rest.forward(side\_length)  
 rest.left(360 / sd)  
 rest.end\_fill()  
 rest.penup()  
 rest.forward(side\_length / 2)  
 rest.right(30)  
  
  
wn = Screen()  
rest = Turtle()  
rest.speed(**'fastest'**)  
draw\_polygons(20, 40\_000)  
rest.hideturtle()  
wn.exitonclick()

