ФГБОУ ВО «Сочинский государственный университет»

Факультет Инновационных, Инженерных и Цифровых Технологий

Кафедра информационных технологий

**ОТЧЕТ**

о выполнении лабораторной работы №6

*«Python. Кортежи, словари, множества, функции»*

по дисциплине **«Алгоритмизация и программирование»**

***Выполнил:***

студент 1 курса гр.24-ПИЦ

Нерсесян С. В.

«\_*29*\_»\_\_\_\_*декабря*\_\_\_\_2024г

***Проверил:***

проф. каф. инф. тех., д.т.н.

Попов Дмитрий Иванович

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_202-г

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отзыв, замечания\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Сочи 2024

# Общие задания для вариантов

1. Условие задачи  
     
   Задание 1. Используя операции индексирования и среза выведите на экран первый и третий элементы кортежа (1, 2, 3, 4, 5), а также срез кортежа из первых трех элементов. Реализуйте вывод двумя способами: используя только положительные и только отрицательные индексы.
2. Код программы  
     
   tuple\_data = (1, 2, 3, 4, 5)

# Использование положительных индексов

print("Положительные индексы:")

print("Первый элемент:", tuple\_data[0])

print("Третий элемент:", tuple\_data[2])

print("Срез первых трёх элементов:", tuple\_data[0:3])

print()

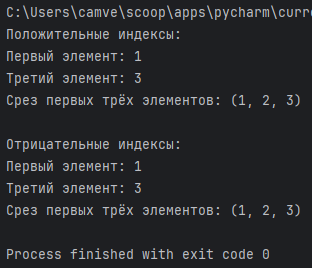
# Использование отрицательных индексов

print("Отрицательные индексы:")

print("Первый элемент:", tuple\_data[-5])

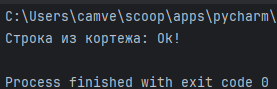
print("Третий элемент:", tuple\_data[-3])

print("Срез первых трёх элементов:", tuple\_data[-5:-2])

1. Скриншоты выполнения программы  
   
2. Условие задачи  
     
   Задание 2. Дан кортеж ((1, 2, ('Ok!', 3)), ('tuple', 4), 5). Выведите на экран строку 'Ok!', использовав синтаксис доступа к элементу кортежа по его индексу.
3. Код программы  
     
   data = ((1, 2, ('Ok!', 3)), ('tuple', 4), 5)

result = data[0][2][0]

print("Строка из кортежа:", result)

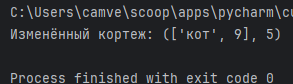
1. Скриншоты выполнения программы  
   
2. Условие задачи  
     
   Задание 3. Дан кортеж (['кит', 1, 3], 5). Замените в списке 'кит' на 'кот', удалите единицу, а также измените значение последнего элемента списка, возведя тройку в квадрат. Выведите кортеж на экран. Попробуйте изменить второй элемент кортежа, умножив его на два.
3. Код программы  
     
   tuple = (['кит', 1, 3], 5)

tuple[0][0] = 'кот' # Заменяем 'кит' на 'кот'

tuple[0].remove(1) # Удаляем единицу из списка

tuple[0][2] = tuple[0][2] \*\* 2 # Возводим тройку в квадрат

print("Изменённый кортеж:", tuple)

1. Скриншоты выполнения программы  
   
2. Условие задачи  
     
   Задание 4. Создайте словарь содержащий данные о человеке. В качестве строковых ключей используйте его имя, возраст и пол. Значения придумайте самостоятельно. Выведите полученный словарь на экран.
3. Код программы  
     
   person = {

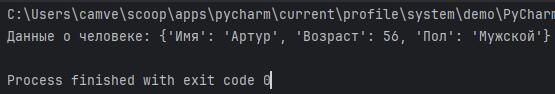
'Имя': 'Артур',

'Возраст': 56,

'Пол': 'Мужской'

}

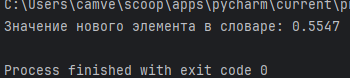
print("Данные о человеке:", person)

1. Скриншоты выполнения программы  
   
2. Условие задачи  
     
   Задание 5. Дан словарь d = {'1': 1.29, '2': 0.43}. Используя доступ к элементам словаря по ключу, найдите произведение 1.29\*0.43, после чего добавьте результат в словарь, а затем выведите значение нового элемента на экран.
3. Код программы  
     
   d = {'1': 1.29, '2': 0.43}

result = d['1'] \* d['2']

d['product'] = result

print("Значение нового элемента в словаре:", d['product'])

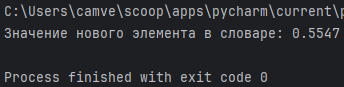
1. Скриншоты выполнения программы  
   
2. Условие задачи  
     
   Задание 6. Сколько элементов будет содержать словарь d\_1 = {'1': 1, '2': 2} после добавления к нему элементов словаря d\_2 = {'2': 'два', '3': 3}? Каковы будут значения элементов итогового словаря? Проверьте свой ответ программно.
3. Код программы  
     
   d\_1 = {'1': 1, '2': 2}

d\_2 = {'2': 'два', '3': 3}

d\_1.update(d\_2)

print("Количество элементов в итоговом словаре:", len(d\_1))

print("Итоговый словарь:", d\_1)

1. Скриншоты выполнения программы  
   
2. Условие задачи  
     
   Задание 7 (множества). Даны три слова 'аквариум', 'мармелад' и 'рама'. Выведите на экран сперва все виды букв, которые присутствуют во всех словах сразу, а затем все виды букв, которые присутствуют в любом из них.
3. Код программы  
     
   word1 = 'аквариум'

word2 = 'мармелад'

word3 = 'рама'

set1 = set(word1)

set2 = set(word2)

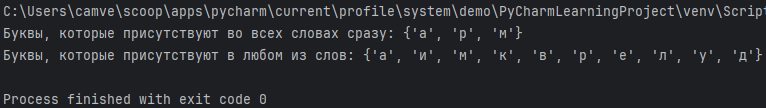
set3 = set(word3)

intersection = set1 & set2 & set3

union = set1 | set2 | set3

print("Буквы, которые присутствуют во всех словах сразу:", intersection)

print("Буквы, которые присутствуют в любом из слов:", union)

1. Скриншоты выполнения программы  
   
2. Условие задачи  
     
   Задание 8. Напишите две функции S(r) и l(r), принимающие в качестве аргумента радиус окружности и возвращающие площадь круга и длину этой окружности соответственно. Затем напишите функцию krug(), которая спрашивает у пользователя радиус окружности, а затем при помощи функций S(r) и l(r) выводит на экран площадь круга и длину окружности, разделённые пробелом.
3. Код программы  
     
   import math

def s(r):

return math.pi \* r \*\* 2

def l(r):

return 2 \* math.pi \* r

def krug():

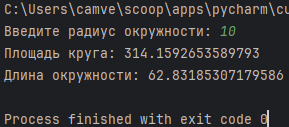
r = float(input("Введите радиус окружности: "))

area = s(r)

circumference = l(r)

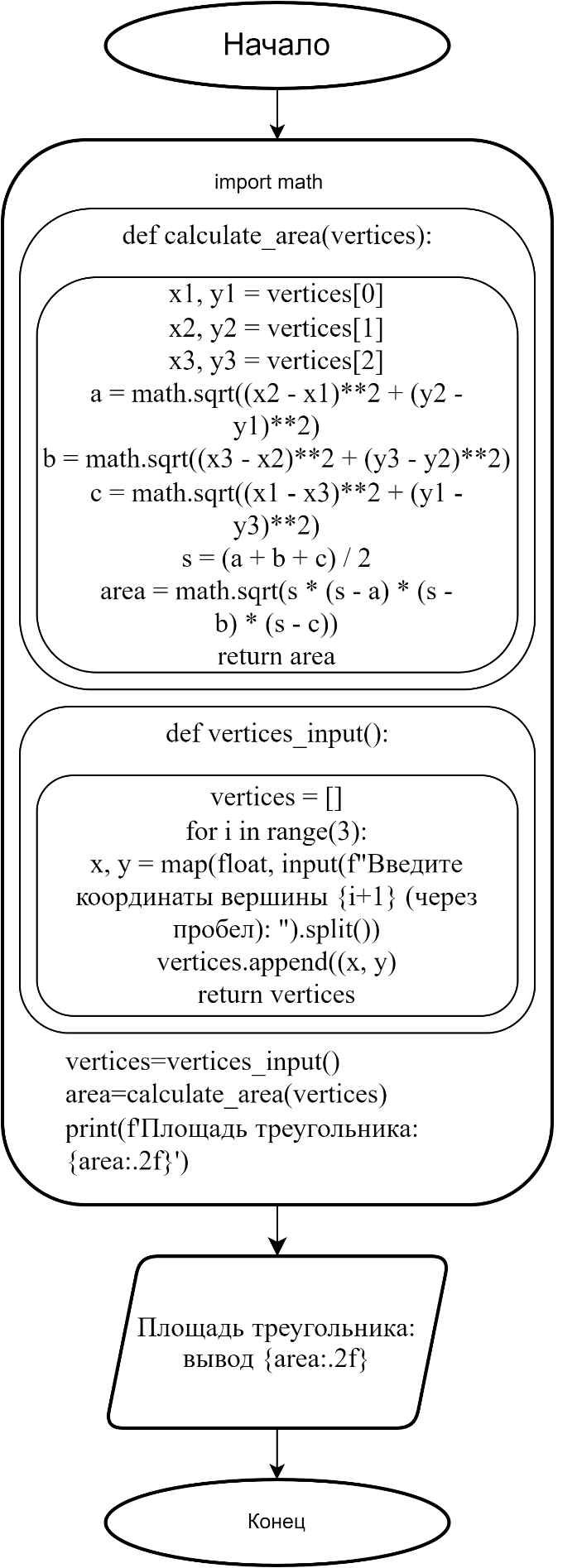
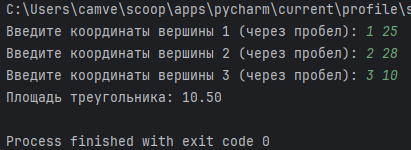
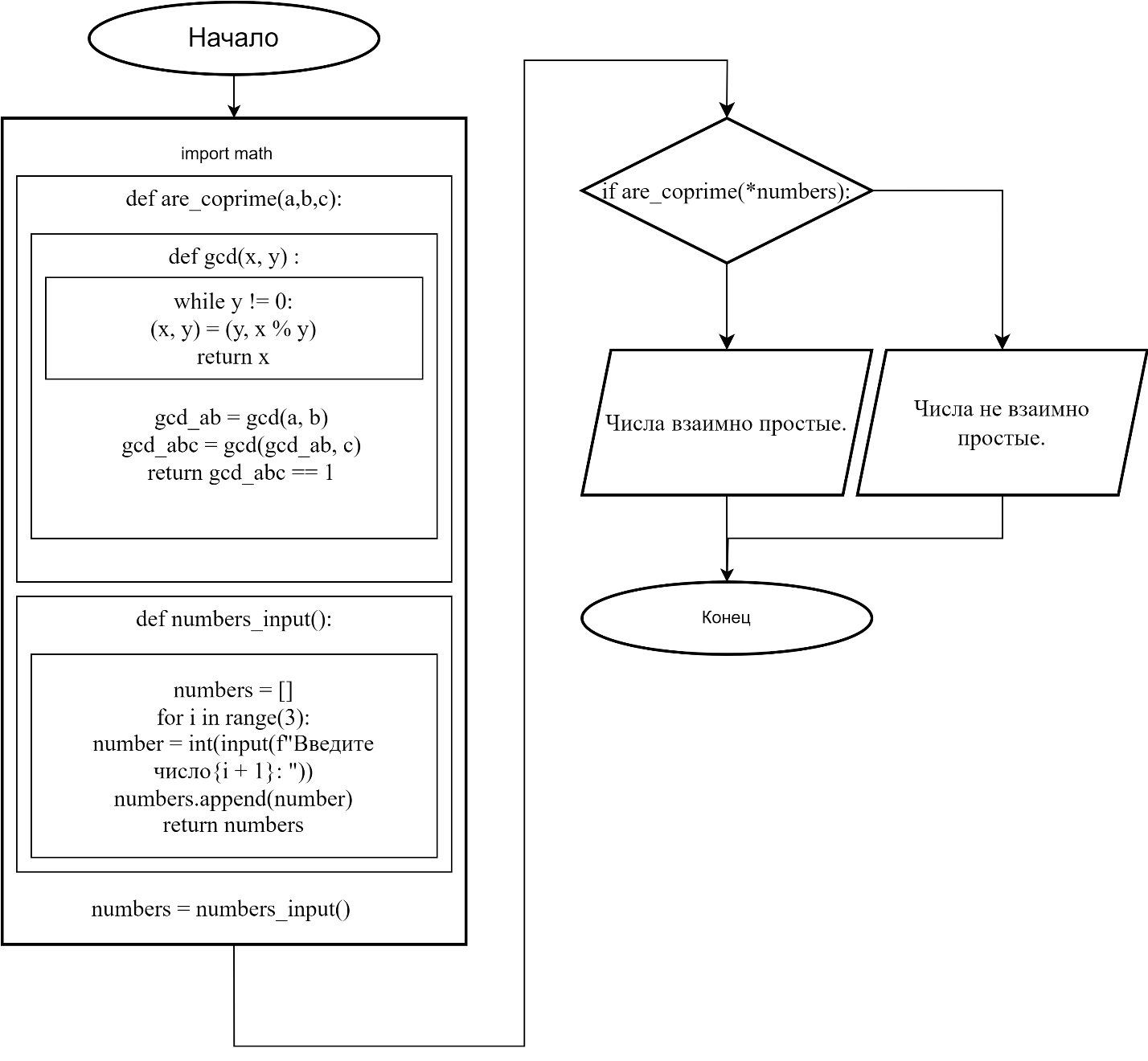
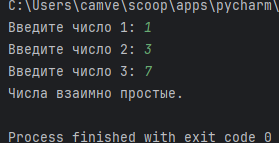
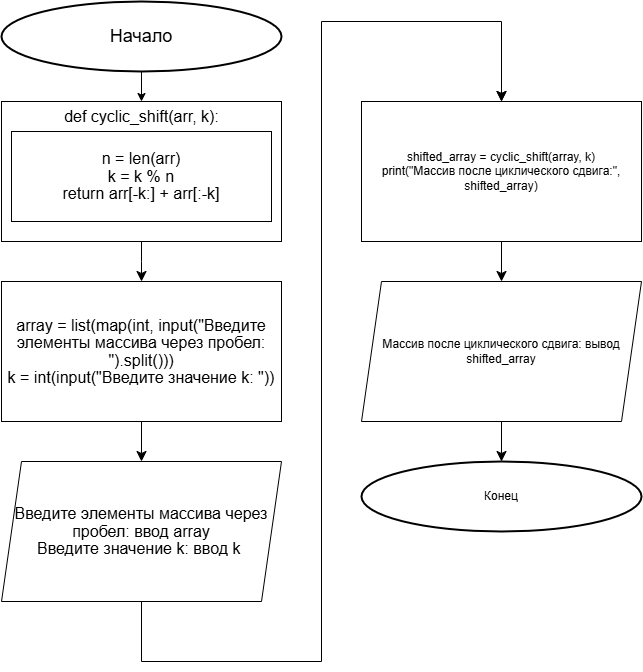
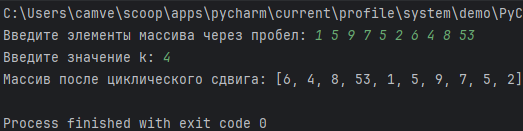
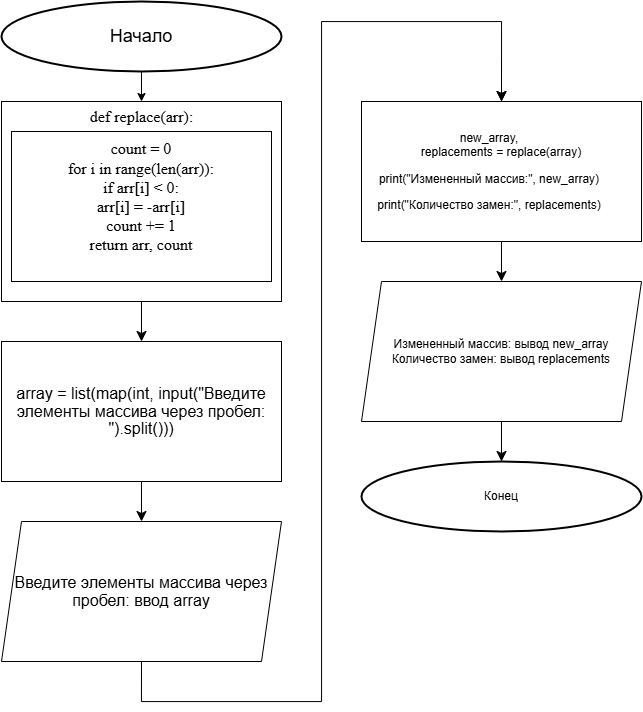
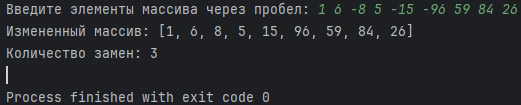
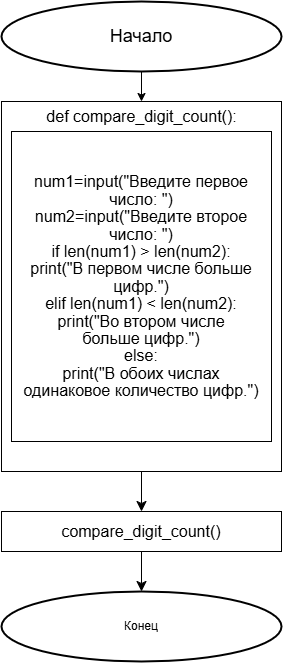
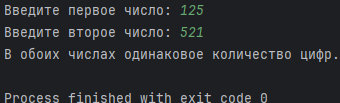
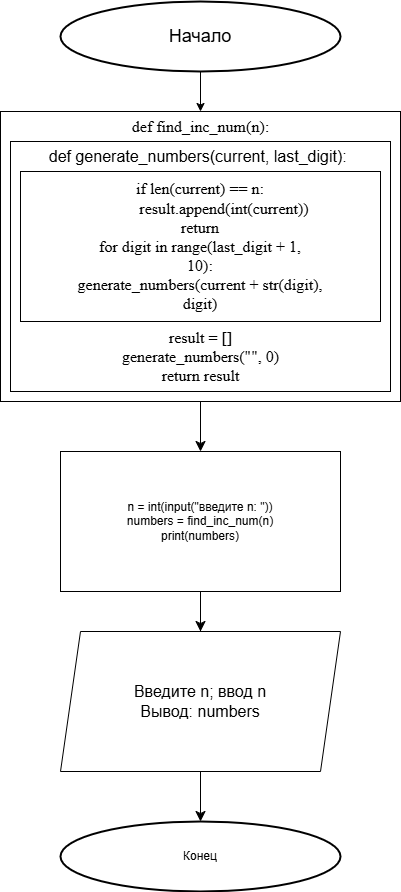
print(f"Площадь круга: {area} \nДлина окружности: {circumference}")

krug()

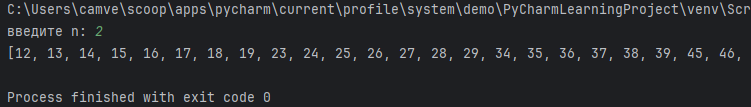
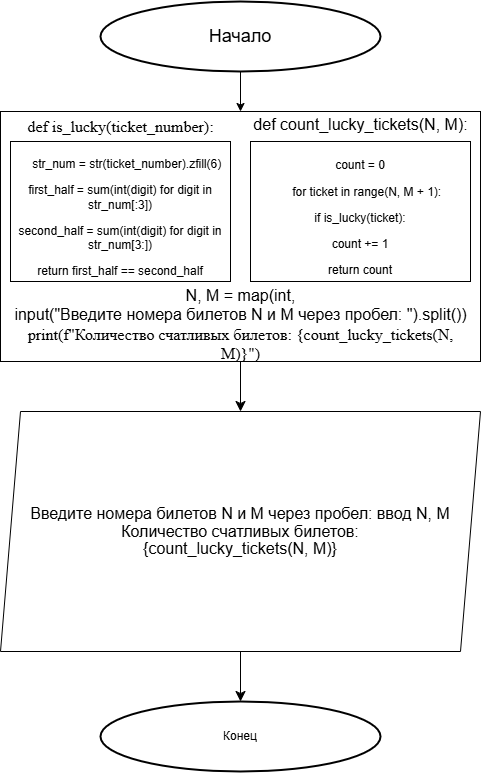
1. Скриншоты выполнения программы  
   

# Задания по вариантам

## Вариант 19, Задачи 1, 8, 12, 15, 21, 30, 34, 37, 41, 42

1. Условие задачи  
   Треугольник задан координатами своих вершин. Составить функцию вычисления его площади.
2. Блок-схема  
   
3. Код программы  
     
   import math  
     
   def calculate\_area(vertices):  
    x1, y1 = vertices[0]  
    x2, y2 = vertices[1]  
    x3, y3 = vertices[2]  
     
    a = math.sqrt((x2 - x1)\*\*2 + (y2 - y1)\*\*2)  
    b = math.sqrt((x3 - x2)\*\*2 + (y3 - y2)\*\*2)  
    c = math.sqrt((x1 - x3)\*\*2 + (y1 - y3)\*\*2)  
    s = (a + b + c) / 2  
     
    area = math.sqrt(s \* (s - a) \* (s - b) \* (s - c))  
    return area  
     
   def vertices\_input():  
    vertices = []  
    for i in range(3):  
    x, y = map(float, input(f"Введите координаты вершины {i+1} (через пробел): ").split())  
    vertices.append((x, y))  
    return vertices  
     
   vertices = vertices\_input()  
   area = calculate\_area(vertices)  
   print(f'Площадь треугольника: {area:.2f}')
4. Скриншоты выполнения программы  
   
5. Условие задачи  
   Проверить, являются ли данные три числа взаимно простыми. Написать функцию с тремя параметрами.
6. Блок-схема   
   
7. Код программы  
     
   import math  
     
   def are\_coprime(a, b, c) :  
    def gcd(x, y) :  
    while y != 0 :  
    (x, y) = (y, x % y)  
    return x  
     
    gcd\_ab = gcd(a, b)  
    gcd\_abc = gcd(gcd\_ab, c)  
     
    return gcd\_abc == 1  
     
     
   def numbers\_input() :  
    numbers = []  
    for i in range(3) :  
    number = int(input(f"Введите число {i + 1}: "))  
    numbers.append(number)  
    return numbers  
     
   numbers = numbers\_input()  
     
   if are\_coprime(\*numbers) :  
    print("Числа взаимно простые.")  
   else :  
    print("Числа не взаимно простые.")
8. Скриншоты выполнения программы  
   
9. Условие задачи  
   Написать функцию, в которую передается одномерный массив и целое число k. Функция возвращает новый массив, элементы которого циклически сдвинуть на k мест вправо.
10. Блок-схема   
    
11. Код программы  
      
    def cyclic\_shift(arr, k):  
     n = len(arr)  
     k = k % n   
     return arr[-k:] + arr[:-k]  
      
    array = list(map(int, input("Введите элементы массива через пробел: ").split()))  
    k = int(input("Введите значение k: "))  
    shifted\_array = cyclic\_shift(array, k)  
    print("Массив после циклического сдвига:", shifted\_array)
12. Скриншоты выполнения программы  
    
13. Условие задачи  
    Заменить отрицательные элементы линейного массива их модулями, не пользуясь стандартной функцией вычисления модуля. Подсчитать количество произведенных замен. Использовать функции.
14. Блок-схема   
    
15. Код программы  
      
    def replace(arr):  
     count = 0  
     for i in range(len(arr)):  
     if arr[i] < 0:  
     arr[i] = -arr[i]  
     count += 1  
     return arr, count  
      
    array = list(map(int, input("Введите элементы массива через пробел: ").split()))  
    new\_array, replacements = replace(array)  
    print("Измененный массив:", new\_array)  
    print("Количество замен:", replacements)
16. Скриншоты выполнения программы  
    
17. Условие задачи  
    Составить функцию, определяющую, в каком из данных двух чисел больше цифр.
18. Блок-схема  
    
19. Код программы  
      
    def compare\_digit\_count():  
     num1 = input("Введите первое число: ")  
     num2 = input("Введите второе число: ")  
     if len(num1) > len(num2):  
     print("В первом числе больше цифр.")  
     elif len(num1) < len(num2):  
     print("Во втором числе больше цифр.")  
     else:  
     print("В обоих числах одинаковое количество цифр.")  
      
    compare\_digit\_count()
20. Скриншоты выполнения программы  
    
21. Условие задачи  
    Найти все натуральные n-значные числа, цифры в которых образуют строго возрастающую последовательность (например, 1234, 5789).
22. Блок-схема   
    
23. Код программы

def find\_inc\_num(n):  
 def generate\_numbers(current, last\_digit):  
 if len(current) == n:  
 result.append(int(current))  
 return  
 for digit in range(last\_digit + 1, 10):  
 generate\_numbers(current + str(digit), digit)  
  
 result = []  
 generate\_numbers("", 0)  
 return result  
  
n = int(input("введите n: "))  
numbers = find\_inc\_num(n)  
print(numbers)

1. Скриншоты выполнения программы  
   
2. Условие задачи  
   На части катушки с автобусными билетами номера шестизначные. Составить программу, определяющую количество счастливых билетов на катушке, если меньший номер билета — N, больший — M (билет является счастливым, если сумма первых трех его цифр равна сумме последних трех).
3. Блок-схема   
   
4. Код программы

def is\_lucky(ticket\_number):

str\_num = str(ticket\_number).zfill(6)

first\_half = sum(int(digit) for digit in str\_num[:3])

second\_half = sum(int(digit) for digit in str\_num[3:])

return first\_half == second\_half

def count\_lucky\_tickets(N, M):

count = 0

for ticket in range(N, M + 1):

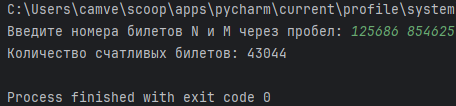
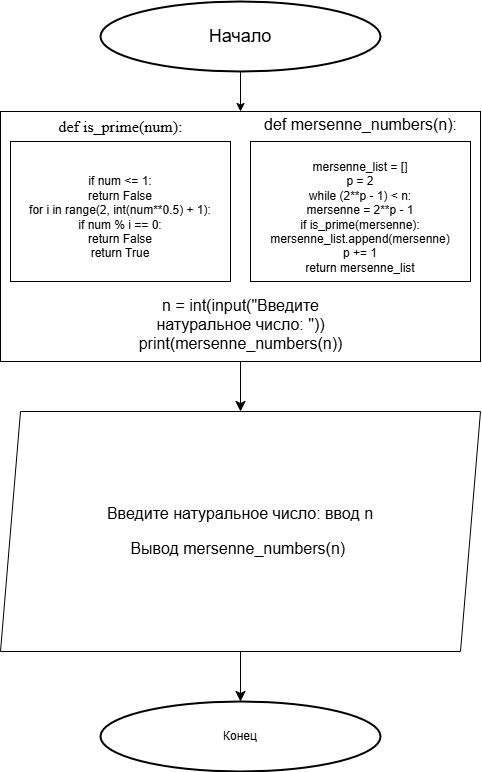
if is\_lucky(ticket):

count += 1

return count

N, M = map(int, input("Введите номера билетов N и M через пробел: ").split())

print(f"Количество счатливых билетов: {count\_lucky\_tickets(N, M)}")

1. Скриншоты выполнения программы  
   
2. Условие задачи  
   Дано натуральное число n. Найти все меньшие n числа Mepceна. (Простое число называется числом Мерсена, если оно может быть представлено в виде 2p - 1, где p – тоже простое число. Например, 31 = 25 - 1 – число Мерсена.)
3. Блок-схема   
   
4. Код программы  
     
   def is\_prime(num):

if num <= 1:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

def mersenne\_numbers(n):

mersenne\_list = []

p = 2

while (2\*\*p - 1) < n:

mersenne = 2\*\*p - 1

if is\_prime(mersenne):

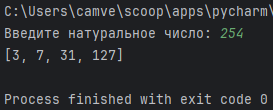
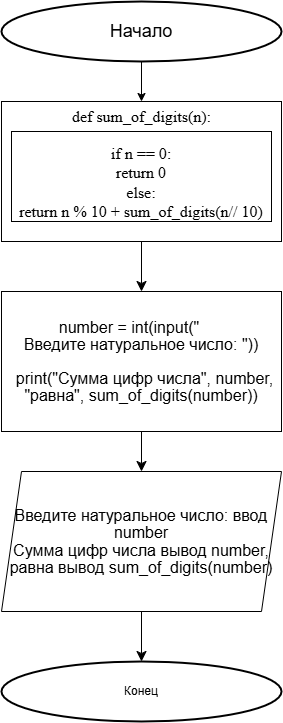
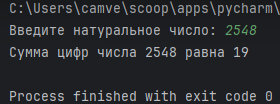
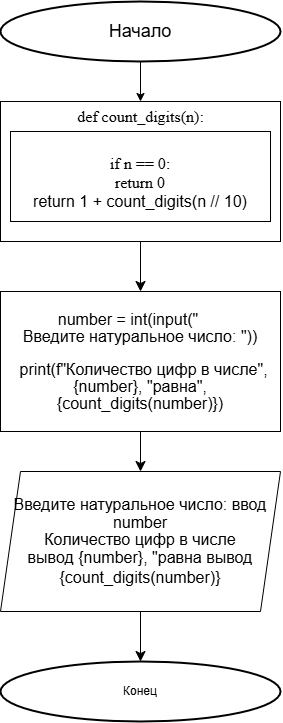
mersenne\_list.append(mersenne)

p += 1

return mersenne\_list

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

print(mersenne\_numbers(n))

1. Скриншоты выполнения программы  
   
2. Условие задачи  
   Найдите сумму цифр заданного натурального числа. Использовать рекурсию
3. Блок-схема   
   
4. Код программы  
     
   def sum\_of\_digits(n):  
    if n == 0:  
    return 0  
    else:  
    return n % 10 + sum\_of\_digits(n // 10)  
     
   number = int(input("Введите натуральное число: "))  
   print("Сумма цифр числа", number, "равна", sum\_of\_digits(number))
5. Скриншоты выполнения программы  
   
6. Условие задачи  
   Подсчитать количество цифр в заданном натуральном числе. Использовать рекурсию.
7. Блок-схема   
   
8. Код программы  
     
   def count\_digits(n):

if n == 0:

return 0

return 1 + count\_digits(n // 10)

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

print(f"Количество цифр в числе {number}: {count\_digits(number)}")

1. Скриншоты выполнения программы  
   