**Отдел образования администрации муниципального образования «Асекеевского район» Оренбургской области**

Асекеево 2025

«**Решение задач № 16 ОГЭ по информатике: разбор на Python»**

Содержание

1. Введение
2. Справочный материал
3. Практикум по решению задач:
4. Задачи для самостоятельного решения
5. Литература

**Введение**

**1. Актуальность темы**

Задание №16 в ОГЭ по информатике появилось в 2025 году. В этом году количество заданий в экзаменационной работе увеличилось с 15 до 16, а задание 16 стало обязательным (ранее существовало альтернативное задание 15.2, которое нужно было выбирать вместо 15.1).

До 2025 года структура экзамена включала 15 заданий, где задания 15.1 и 15.2 были альтернативными: ученик мог выбрать только одно из них. С 2025 года оба этих задания стали обязательными, что увеличило общее количество заданий до 16.

В доступных источниках не удалось найти прямых данных о проценте учащихся, приступивших к решению задания 16 ОГЭ по информатике. Однако есть информация о среднем проценте выполнения этого задания, которая косвенно указывает на уровень вовлечённости учащихся.

По данным аналитического отчёта за 2025 год, средний процент выполнения задания 16 составил **0,12%**. Это задание относится к высокому уровню сложности и требует умения создавать и выполнять программы на универсальном языке программирования. Низкий показатель выполнения связан с тем, что лишь небольшая часть участников приступила к его решению.

Некоторые дополнительные сведения:

* В группе учащихся, получивших отметку «4», процент выполнения задания 16 составил **1,91%**.
* Среди выпускников, получивших оценку «5», средний процент выполнения достиг **34,98%**.

В методических материалах и аналитических отчётах отмечается, что Python становится всё более популярным среди школьников, сдающих ОГЭ по информатике. Это связано с его простотой синтаксиса, лаконичностью кода и широким распространением в образовательной среде.

**2. Типичные условия задач**

В заданиях встречаются следующие типы обработки данных:

* поиск количества чисел, удовлетворяющих условию;
* вычисление суммы чисел по критерию;
* определение минимального/максимального значения среди чисел, подходящих под условие;
* расчёт среднего арифметического подходящих чисел;
* проверка наличия чисел с заданными свойствами.

**Справочный материал для решения задачи № 16 ОГЭ на языке программирования Python.**

1. Ввод и вывод данных

Для печати значений в Питоне есть функция print(). Внутри круглых скобок через запятую мы пишем то, что хотим вывести. Вот программа, которая делает несколько вычислений:

*print(5 + 10)*

*print(3 \* 7, (17 - 2) \* 8)*

*print(2 \*\* 16) # две звёздочки означают возведение в степень*

*print(37 / 3) # один слэш — это деление с ответом-дробью*

*print(37 // 3) # два слэша считают частное от деления нацело*

*# это как операция div в других языках*

*print(37 % 3) # процент считает остаток от деления нацело*

*# это как операция mod в других языках*

Для ввода данных в программу мы используем функцию input(). Она считывает одну строку.

*a = input()*

*b = input()*

*s = a + b*

*print(s)*

Программа выводит 57, хотя в реальной жизни 5 + 7 будет 12. Это произошло потому, что Питон в третьей строчке «сложил» две строки, а не два числа. В Питоне две строки складываются так: к первой строке приписывается вторая.

Каждый объект относится к какому-то типу. Строки хранятся в объектах типа str, целые числа хранятся в объектах типа int, дробные числа (вещественные числа) — в объектах типа float.

Вот пример правильной программы, которая считывает два числа и выводит их сумму:

*a = int(input())*

*b = int(input())*

*s = a + b*

*print(s)*

### 2. Условия

условная инструкция в Питоне имеет следующий синтаксис:

*if Условие:*

*Блок инструкций 1*

*else:*

*Блок инструкций 2*

Для выделения блока инструкций, относящихся к инструкции if или else, в языке Питон используются отступы. Все инструкции, которые относятся к одному блоку, должны иметь равную величину отступа, то есть одинаковое число пробелов в начале строки. Рекомендуется использовать отступ в 4 пробела и не рекомендуется использовать в качестве отступа символ табуляции.

Операторы сравнения

Как правило, в качестве проверяемого условия используется результат вычисления одного из следующих операторов сравнения:

**<** Меньше.

**>** Больше.

**<=** Меньше или равно.

**>=** Больше или равно.

**==** Равенство.

**!=** Неравенство.

Операторы сравнения в Питоне можно объединять в цепочки (в отличии от большинства других языков программирования, где для этого нужно использовать логические связки), например, a == b == c или 1 <= x <= 10.

Иногда нужно проверить одновременно не одно, а несколько условий. Например, проверить, является ли данное число четным, можно при помощи условия (n % 2 == 0) (остаток от деления n на 2 равен 0), а если необходимо проверить, что два данных целых числа n и m являются четными, необходимо проверить справедливость обоих условий: n % 2 == 0и m % 2 == 0, для чего их необходимо объединить при помощи оператора and (логическое И): n % 2 == 0 and m % 2 == 0.

В Питоне существуют стандартные логические операторы: логическое И, логическое ИЛИ, логическое отрицание.

Каскадные условные инструкции. В такой конструкции условия if, ..., elif проверяются по очереди, выполняется блок, соответствующий первому из истинных условий. Если все проверяемые условия ложны, то выполняется блок else, если он присутствует.

### 3. Вычисления

Для целых чисел определены операции +, -, \* и \*\*. Операция деления / для целых чисел возвращает вещественное число (значение типа float). Также функция возведения в степень возвращает значение типа float, если показатель степени — отрицательное число.

Но есть и специальная операция целочисленного деления, выполняющегося с отбрасыванием дробной части, которая обозначается // (она соответствует операции div в Паскале). Она возвращает целое число: целую часть частного. Другая близкая ей операция − это операция взятия остатка от деления, обозначаемая % (она соответствует операции mod в Паскале).

### Преобразование действительных чисел к целому производится с округлением в сторону нуля, то есть int(1.7) == 1, int(-1.7) == -1.

Для повторения цикла некоторое заданное число раз n можно использовать цикл for вместе с функцией range:

*for i in range(4): # равносильно инструкции for i in 0, 1, 2, 3:*

*# здесь можно выполнять циклические действия*

*print(i)*

*print(i \*\* 2)*

*# цикл закончился, поскольку закончился блок с отступом*

*print('Конец цикла')*

Функция range может также принимать не один, а два параметра. Вызов range(a, b) означает, что индексная переменная будеть принимать значения от a до b - 1, то есть первый параметр функции range, вызываемой с двумя параметрами, задает начальное значение индексной переменной, а второй параметр — первое значение, которое индексная переменная принимать **не будет**. Если же a≥b, то цикл не будет выполнен ни разу. Например, для того, чтобы просуммировать значения чисел от 1 до n можно воспользоваться следующей программой:

*sum = 0*

*n = 5*

*for i in range(1, n + 1):*

*sum += i*

*print(sum)*

5. Строки

Узнать количество символов (длину строки) можно при помощи функции len.

Любой другой объект в Питоне можно перевести к строке, которая ему соответствует. Для этого нужно вызвать функцию str().Перевод строки в число осуществляется функцией int().Срез (slice) — извлечение из данной строки одного символа или некоторого фрагмента подстроки или подпоследовательности.

Есть три формы срезов. Самая простая форма среза: взятие одного символа строки, а именно, S[i] — это срез, состоящий из одного символа, который имеет номер i.

Срез с двумя параметрами: S[a:b] возвращает подстроку из b - a символов, начиная с символа c индексом a, то есть до символа с индексом b, не включая его. Если задать срез с тремя параметрами S[a:b:d], то третий параметр задает шаг, как в случае с функцией range, то есть будут взяты символы с индексами a, a + d, a + 2 \* d и т. д.

#### Метод count

Подсчитывает количество вхождений одной строки в другую строку. Простейшая форма вызова S.count(T)  возвращает число вхождений строки T внутри строки S. При этом подсчитываются только непересекающиеся вхождения, например:

*print('Abracadabra'.count('a'))*

*# вернёт 4*

*print(('a' \* 10).count('aa'))*

*# вернёт 5*

При указании трех параметров S.count(T, a, b), будет выполнен подсчет числа вхождений строки T в срезе S[a:b].

### 6. Цикл while

При выполнении цикла while сначала проверяется условие. Если оно ложно, то выполнение цикла прекращается и управление передается на следующую инструкцию после тела цикла. После тела цикла можно написать слово else: и после него блок операций, который будет выполнен *один раз* после окончания цикла, когда проверяемое условие станет неверно. Смысл появляется только вместе с инструкцией break. Если во время выполнения Питон встречает инструкцию break внутри цикла, то он сразу же прекращает выполнение этого цикла и выходит из него. При этом ветка else исполняться не будет. Разумеется, инструкцию break осмыленно вызывать только внутри инструкции if, то есть она должна выполняться только при выполнении какого-то особенного условия. Другая инструкция управления циклом — continue (продолжение цикла). Если эта инструкция встречается где-то посередине цикла, то пропускаются все оставшиеся инструкции до конца цикла, и исполнение цикла продолжается со следующей итерации.

### 7. Списки

Для хранения данных, заданных в виде последовательности, можно использовать структуру данных, называемую в Питоне список (в большинстве же языков программирования используется другой термин “массив”). Список представляет собой последовательность элементов, пронумерованных от 0, как символы в строке. Список можно задать перечислением элементов списка в квадратных скобках, например, список можно задать так:

*Primes = [2, 3, 5, 7, 11, 13]*

*Rainbow = ['Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Indigo', 'Violet']*

Рассмотрим несколько способов создания и считывания списков. Прежде всего, можно создать пустой список (не содержащий элементов, длины 0), а в конец списка можно добавлять элементы при помощи метода append.

*a = [] # заводим пустой список*

*n = int(input()) # считываем количество элемент в списке*

*for i in range(n):*

*new\_element = int(input()) # считываем очередной элемент*

*a.append(new\_element) # добавляем его в список*

*# последние две строки можно было заменить одной:*

*# a.append(int(input()))*

*print(a)*

### Методы split и join

Элементы списка могут вводиться по одному в строке, в этом случае строку целиком можно считать функцией input(). После этого можно использовать метод строки split(), возвращающий список строк, которые получатся, если исходную строку разрезать на части по пробелам. Пример:

*s = input() # s == '1 2 3'*

*a = s.split() # a == ['1', '2', '3']*

Если хочется получить список именно из чисел, то можно затем элементы списка по одному преобразовать в числа:

*a = input().split()*

*for i in range(len(a)):*

*a[i] = int(a[i])*

Используя специальную магию Питона — генераторы — то же самое можно сделать в одну строку:

*a = [int(s) for s in input().split()]*

## Операции со списками

Со списками можно легко делать много разных операций.

|  |  |
| --- | --- |
| x in A | Проверить, содержится ли элемент в списке. Возвращает True или False |
| x not in A | То же самое, что not(x in A) |
| min(A) | Наименьший элемент списка |
| max(A) | Наибольший элемент списка |
| A.index(x) | Индекс первого вхождения элемента x в список, при его отсутствии генерирует исключение ValueError |
| A.count(x) | Количество вхождений элемента x в список |

**Практикум «Как решить задание  ОГЭ по информатике: разбор на Python»**

**1. Основные конструкции Python для решения задачи**

**В** обобщенной форме алгоритм решения задачи можно представить задачи в виде блок схемы (слайд 2).

Как правило он содержит один цикл. В теле цикла одно условие. При истинности условий необходимо найти вычислить количество чисел удовлетворяющих условию, сумму или найти максимальный/минимальный элемент числового ряда .

Рассмотрим каждый из этих критериев по отдельности .

1. Если заранее известно количество повторений целесообразно использовать цикл со счетчиком. Основные формулировки таких задач

* Программа получает на вход количество чисел в последовательности, а затем сами числа
* Программа получает на вход число проехавших автомобилей N (1 < N < 30), затем указываются их скорости
* Введите с клавиатуры 5 положительных целых чисел
* Напишите программу, которая по двум данным натуральным числам *a* и *b*, не превосходящим 30000, подсчитывает количество четных натуральных чисел на отрезке [*a*, *b*] (включая концы отрезка). Программа получает на вход два натуральных числа *a* и *b*, при этом гарантируется, что 1 ≤ *a* ≤ *b* ≤ 30000

Основная алгоритмическая конструкция выглядит следующим образом.

* ……

n= int(input())

for i in range(n):

a = int(input())

if <условие>:

<тело ветвления при истинности условия>

else:

<тело ветвления при истинности условия>

print(ответ)

Для последней формулировки целесообразно использовать конструкцию

* ……

a= int(input())

b= int(input())

for i in range(a,b+1):

a = int(input())

if <условие>:

<тело ветвления при истинности условия>

else:

<тело ветвления при истинности условия>

print(ответ)

Если не известно количество повторений, но известно условие остановки целесообразно воспользоваться циклом с условием . Примеры формулировок задач:

* Программа получает на вход натуральные числа, количество введенных чисел неизвестно, последовательность чисел заканчивается числом 0 (0  — признак окончания ввода, не входит в последовательность)
* ……

a= int(input())

while a!=0:

if <условие>:

<тело ветвления при истинности условия>

else:

<тело ветвления при истинности условия>

a = int(input())

print(ответ)

Можно выделить следующие условия и их комбинации

Рассмотрим каждый из условий по отдельности

* 1. Числа кратные…
* If a % b == 0:
* <>
* 2. Числа оканчивающиеся на цифру…
* If a % 10==b:
* <>
* 3. Двухзначные, трехзначные числа и.т.д
* If a>9 and a<100:
* <>
* 4. Четные и нечетные числа
* Четные числа
* If a%2 == 0:
* <>
* Нечетные числа
* If a%2 != 0:
* <>
* 5. Максимальные и минимальные числа
* Минимальное число
* Min = минимальное число -1
* ….
* If a< min:
* <>
* Максимальное число
* Max = максимальное число +1
* If a< min:
* <>
* 6. запись которых в системе счисления с основанием …. Оканчивается….
* B = основание системы счисления
* C = цифра на которое оканчивается число
* …….
* If a//b == c

**2. Типовые задачи**

* **1 Напишите программу, которая в последовательности натуральных чисел определяет максимальное число, кратное 5. Программа получает на вход количество чисел в последовательности, а затем сами числа. В последовательности всегда имеется число, кратное 5. Количество чисел не превышает 1000. Введенные числа не превышают 30 000. Программа должна вывести одно число  — максимальное число, кратное 5**

Проанализируем условие задачи

1. Программа получает на вход количество чисел в последовательности, а затем сами числа Это соответствует циклу for

n = int(input())

for i in range(n):

<>

2 последовательности натуральных чисел определяет максимальное число, кратное 5 Т.е число должно быть одновременно и максимальным и кратным 5 – это соответствует логической операции «и», также необходимо инициализировать минимальное число

maxi=0

n = int(input())

for i in range(n):

k = int(input())

if maxi< k and k%5==0:

maxi=k

print(maxi)

* **2 Напишите программу, которая в последовательности натуральных чисел определяет сумму чисел кратных 6. Программа получает на вход количество чисел в последовательности, а затем сами числа. В последовательности всегда имеется число, кратное 6. Количество чисел не превышает 100. Введенные числа не превышают 300. Программа должна вывести одно число  — сумму чисел, кратных 6.**

1. Программа получает на вход количество чисел в последовательности, а затем сами числа Это соответствует циклу for

* n = int(input())
* for i in range(n):
* <>

2. чисел кратных 6

* n = int(input())
* for i in range(n):
* k=int(input(())
* if k//6==0:
* <>

3. последовательности натуральных чисел определяет сумму чисел

* summi=0
* n = int(input())
* for i in range(n):
* k=int(input(())
* if k//6==0:
* sum=sum+k

print(sum)

* **Напишите программу для решения следующей задачи. Камера наблюдения регистрирует в автоматическом режиме скорость проезжающих мимо нее автомобилей, округляя значения скорости до целых чисел. Необходимо определить максимальную зарегистрированную скорость автомобиля. Если скорость хотя бы одного автомобиля была меньше 30 км/ч, выведите «YES», иначе выведите «N0».**

**Программа получает на вход число проехавших автомобилей N (1 < N < 30), затем указываются их скорости. Значение скорости не может быть меньше 1 и больше 300.Программа должна сначала вывести максимальную скорость, затем YES или NO**

1. Программа получает на вход число проехавших автомобилей N (1 < N < 30), затем указываются их скорости

* n = int(input())
* for i in range(n):
* a = int(input())

2. Необходимо определить максимальную зарегистрированную скорость автомобиля

* maxi=0
* n = int(input())
* for i in range(n):
* a = int(input())
* if a>maxi:
* maxi=a
* 3. Если скорость хотя бы одного автомобиля была меньше 30 км/ч, выведите «YES», иначе выведите «N0».
* d = False
* maxi=0
* n = int(input())
* for i in range(n):
* a = int(input())
* if a>maxi:
* maxi=a
* if a<30:
* b=True

4. Программа должна сначала вывести максимальную скорость, затем YES или NO

* d = False
* maxi=0
* n = int(input())
* for i in range(n):
* a = int(input())
* if a>maxi:
* maxi=a
* if a<30:
* d=True

pint(maxi)

if d:

print(‘Yes’)

else:

print(‘No’)

* **Напишите программу, которая в последовательности натуральных чисел определяет сумму всех чисел, кратных 6 и оканчивающихся на 4. Программа получает на вход натуральные числа, количество введенных чисел неизвестно, последовательность чисел заканчивается числом 0 (0  — признак окончания ввода, не входит в последовательность). Количество чисел не превышает 1000. Введенные числа не превышают 30 000. Программа должна вывести одно число: сумму всех чисел, кратных 6 и оканчивающихся на 4.**

1. Программа получает на вход натуральные числа, количество введенных чисел неизвестно, последовательность чисел заканчивается числом 0 (0  — признак окончания ввода, не входит в последовательность). Это соответствует циклу с условием.

* a = int(input())
* while a != 0:

<>

2. **всех чисел, кратных 6 и оканчивающихся на 4.**

* a = int(input())
* while a != 0:
* if a % 6 == 0 and a % 10 == 4:

<>

3. **определяет сумму**

* sumi = 0
* a = int(input())
* while a != 0:
* if a % 6 == 0 and a % 10 == 4:
* sumi += a
* a = int(input())
* print(sumi)

**3. Типичные ошибки и способы их избежать**

1. **Неправильная инициализация переменных**
   * Для поиска минимума: инициализировать меньшим числом (например, min\_num = 0).
   * Для поиска максимума: инициализировать большим числом (например, max\_num = 30001).
2. **Деление на ноль при вычислении среднего**
   * Всегда проверять, что количество чисел (q) больше нуля перед делением.
3. **Неучёт границ диапазона**
   * Внимательно читать условия (например, «числа не превышают 30 000»).
4. **Ошибки в условиях проверки**
   * Использовать правильные операторы: % для остатка от деления, == для сравнения.
   * Проверять логические связки (and, or, not).

**4. Рекомендации по подготовке**

1. **Изучите базовые конструкции Python:**
   * ввод/вывод данных;
   * условные операторы;
   * циклы for и while.
2. **Практикуйтесь на типовых задачах:**
   * решайте задания с разными условиями (поиск суммы, максимума, количества);
   * отрабатывайте ввод данных как с известным, так и с неизвестным количеством элементов.
3. **Анализируйте решения:**
   * разбирайте примеры кода, понимайте логику каждого шага;
   * проверяйте работу программы на разных тестовых данных.
4. **Следите за форматом вывода:**
   * выводите ровно то, что требуется в условии (целое число, дробное с точностью, текстовое сообщение).
5. **Используйте онлайн‑среды для тестирования:**
   * проверяйте код в интерактивных интерпретаторах Python (например, на сайтах типа repl.it).

**5. Заключение**

Использование Python для решения 16‑й задачи ОГЭ позволяет:

* наглядно реализовать алгоритмы обработки данных;
* эффективно применять циклы и условные операторы;
* получать точный результат при соблюдении условий задачи.

Успешное выполнение задания требует:

* понимания условий задачи;
* знания базовых конструкций языка;
* внимательности при написании кода и тестировании.

Регулярная практика и анализ типовых решений помогут уверенно справляться с этим заданием на экзамене

**Задачи для самостоятельного решения**

**1.** Напишите программу, которая в последовательности натуральных чисел определяет количество трехзначных чисел, кратных 4. Программа получает на вход натуральные числа, количество введенных чисел неизвестно, последовательность чисел заканчивается числом 0 (0  — признак окончания ввода, не входит в последовательность).

Количество чисел не превышает 1000. Введенные числа не превышают 30 000. Программа должна вывести одно число: количество трехзначных чисел, кратных 4.

**Пример работы программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 120  9  365  4  0 | 1 |

**Решение.** Решением является программа, записанная на любом языке программирования.

**решение на языке Python.**

count = 0

a = 1

while a != 0:

a = int(input())

if a == 0:

break

if a % 4 == 0 and 99 < a < 1000:

count += 1

print(count)

Возможны и другие варианты решения.

Для проверки правильности работы программы необходимо использовать следующие тесты:

|  |  |  |
| --- | --- | --- |
| **№** | **Входные данные** | **Выходные данные** |
| 1 | 300  16  17  0 | 1 |
| 2 | 1000  8  0 | 0 |
| 3 | 400  420  0 | 2 |

**2.**Напишите программу, которая в последовательности натуральных чисел определяет количество чисел, кратных 3. Программа получает на вход количество чисел в последовательности, а затем сами числа. В последовательности всегда имеется число, кратное 3. Количество чисел не превышает 100. Введенные числа не превышают 300. Программа должна вывести одно число  — количество чисел, кратных 3.

**Пример работы программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3  12  26  24 | 2 |

**Решение. на языке Python.**

n = int(input())

kol = 0

for i in range(n):

a = int(input())

if a % 3 == 0:

kol += 1

print(kol)

**3**Напишите программу, которая в последовательности натуральных чисел определяет сумму чисел, оканчивающихся на 4. Программа получает на вход количество чисел в последовательности, а затем сами числа. В последовательности всегда имеется число, оканчивающееся на 4. Количество чисел не превышает 1000. Введенные числа не превышают 30 000. Программа должна вывести одно число  — сумму чисел, оканчивающихся на 4.

**Пример работы программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3  14  25  24 | 38 |

**Решение.**

**на языке Python.**

n = int(input())

sumi = 0

for i in range(n):

a = int(input())

if a % 10 == 4:

sumi += a

print(sumi)

**4.**Напишите программу для решения следующей задачи. Камера наблюдения регистрирует в автоматическом режиме скорость проезжающих мимо нее автомобилей, округляя значения скорости до целых чисел. Необходимо определить:

1)  разность максимальной и минимальной скоростей автомобилей;

2)  количество автомобилей, скорость которых не превышала 30 км/ч.

Программа получает на вход число проехавших автомобилей N (1 ≤ N ≤ 30), затем указываются их скорости. Значение скорости не может быть меньше 1 и больше 300. Программа должна сначала вывести разность максимальной и минимальной скоростей автомобилей, затем количество автомобилей, скорость которых не превышала 30 км/ч.

**Пример работы программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 474696396 | 330 |

**Решение.**

**на языке Python.**

max = 0

min = 301

num = 0

n = int(input())

for i in range(n):

a = int(input())

if a > max:

max = a

if a < min:

min = a

if a <= 30:

num += 1

print(max - min)

print(num)

Для проверки правильности работы программы необходимо использовать следующие тесты:

|  |  |  |
| --- | --- | --- |
|  | **Входные данные** | **Выходные данные** |
| 1 | 3  50  81  65 | 31  0 |
| 2 | 1  30 | 0  1 |
| 3 | 2  20  52 | 32  1 |
| 4 | 3  15  25  140 | 125  2 |

**5.**Напишите программу для решения следующей задачи. На контрольной работе по алгебре ученикам 9 класса было предложено 10 примеров. Неудовлетворительная оценка выставляется, если правильно решено менее половины примеров. Сколько неудовлетворительных оценок было получено учениками? Если хотя бы один из учеников правильно решил все задачи, выведите YES, иначе выведите NO. Программа получает на вход количество учеников в классе N (1 ≤ N ≤ 30), затем для каждого ученика вводится количество правильно решенных примеров.

**Пример работы программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 4  3  9  2  8 | 2  NO |

**Решение на языке Python.**

count = 0

top = 0

n = int(input())

for i in range(1, n + 1):

a = int(input())

if a < 5:

count += 1

if a == 10:

top = 1

print(count)

if top == 1:

print('YES')

else:

print('NO')

**6.**Напишите программу для решения следующей задачи. Камера наблюдения регистрирует в автоматическом режиме скорость проезжающих мимо нее автомобилей, округляя значения скорости до целых чисел. Необходимо определить максимальную зарегистрированную скорость автомобиля. Если скорость хотя бы одного автомобиля была меньше 30 км/ч, выведите «YES», иначе выведите «N0».

Программа получает на вход число проехавших автомобилей N (1 < N < 30), затем указываются их скорости. Значение скорости не может быть меньше 1 и больше 300.Программа должна сначала вывести максимальную скорость, затем YES или NO.

**Пример работы программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 474696366 | 74NO |

**на языке Python.**

max = 0

b = False

n = int(input())

for i in range(n):

a = int(input())

if a > max:

max = a

if a < 30:

b = True

print(max)

if b == True:

print('YES')

else:

print('NO')

Для проверки правильности работы программы необходимо использовать следующие тесты:

|  |  |  |
| --- | --- | --- |
|  | **Входные данные** | **Выходные данные** |
| 1 | 3  50  81  65 | 81  NO |
| 2 | 2  13  52 | 52  YES |
| 4 | 3  150  140  25 | 150  YES |

**7.** Напишите программу, которая в последовательности натуральных чисел определяет сумму всех чисел, кратных 6 и оканчивающихся на 6. Программа получает на вход натуральные числа, количество введенных чисел неизвестно, последовательность чисел заканчивается числом 0 (0 – признак окончания ввода, не входит в последовательность). Количество чисел не превышает 100. Введенные числа не превышают 300. Программа должна вывести одно число: сумму всех чисел, кратных 6 и оканчивающихся на 6.

**Пример работы программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 36  12  16  66  11  0 | 102 |

**решение на языке Python.**

summi = 0

a = 1

while a != 0:

a = int(input())

if a == 0:

break

if a % 6 == 0 and a % 10 == 6:

summi += a

print(summi)

Возможны и другие варианты решения. Для проверки правильности работы программы необходимо использовать следующие тесты:

|  |  |  |
| --- | --- | --- |
| **№** | **Входные данные** | **Выходные данные** |
| 1 | 66  36  0 | 102 |
| 2 | 60  0 | 0 |
| 3 | 26  0 | 0 |
| 4 | 20  96  0 | 96 |

**8.** Введите с клавиатуры 5 положительных целых чисел. Вычислите сумму тех из них, которые делятся на 4 и при этом заканчиваются на 6. Программа должна вывести одно число: сумму чисел, введенных с клавиатуры, кратных 4 и оканчивающихся на 6.

**Пример работы программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1216362630 | 52 |

**Решение. на языке Python.**

sumi = 0

for i in range(5):

a = int(input())

if (a % 4 == 0) and (a % 10 == 6):

sumi += a

print(sumi)

Для проверки правильности работы программы можно использовать следующие тесты:

|  |  |  |
| --- | --- | --- |
|  | **Входные данные** | **Выходные данные** |
| 1 | 2  3  4  5  6 | 0 |
| 2 | 16  14  15  13  12 | 16 |
| 3 | 16  26  36  56  66 | 108 |

**9.** Ученики 4 класса вели дневники наблюдения за погодой и ежедневно записывали дневную температуру. Найдите среднюю температуру для дней, когда температура поднималась выше нуля градусов. Определите количество таких дней. Гарантируется, что за время наблюдения хотя бы в один из дней температура поднималась выше нуля градусов. Программа получает на вход количество дней, в течение которых проводилось наблюдение N (1 ≤ N ≤ 31), затем для каждого дня вводится температура.

**Пример работы программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 4  –5  12  –2  8 | 10.0  2 |

**Приведем решение на языке Python.**

n = int(input())

s = 0

cnt = 0

for i in range(n):

x = int(input())

if x > 0:

s += x

cnt +=1

print(s / cnt)

print(cnt)

**10.**Напишите программу, которая в последовательности натуральных чисел определяет количество чисел, кратных 4. Программа получает на вход количество чисел в последовательности, а затем сами числа. В последовательности всегда имеется число, кратное 4. Количество чисел не превышает 1000. Введенные числа не превышают 30 000. Программа должна вывести одно число  — количество чисел, кратных 4.

**Пример работы программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3162624 | 2 |

**решение на языке Python.**

n = int(input())

kol = 0

for i in range(n):

a = int(input())

if a % 4 == 0:

kol += 1

print(kol)

Для проверки правильности работы программы необходимо использовать следующие тесты:

|  |  |  |
| --- | --- | --- |
|  | **Входные данные** | **Выходные данные** |
| 1 | 2  4  7 | 1 |
| 2 | 3  16  4  24 | 3 |
| 3 | 4  24  4  44  2 | 3 |

1. Литература

* 1. Открытый банк ЕГЭ и ОГЭ.
* 2. Материалы с сайта ФИПИ <http://www.fipi.ru>
* 3 <https://ege.sdamgia.ru/>