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

**Докладчик**

**Камалиев Рустам Насыхович**

**учитель информатики**

**МБОУ Кутлуевской СОШ**

**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.

**3. 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

**4. 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)

**5. 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).

**6. 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).

**7. 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