Умскул

ЕГЭ
Информатика

★ Интенсивный курс ★

Готовься к экзаменам с Умскул

Виктория Ланская

Москва

УДК 373.5:004
ББК 32.81я721
Л22

Ланская, Виктория.

Л22 ЕГЭ. Информатика / Виктория Ланская. — Москва : Эксмо, 2026. — 128 с. — (Готовься к экзаменам с Умскул).

ISBN 978-5-04-222326-6

В справочнике от популярной онлайн-школы «Умскул» ты найдёшь всё, что необходимо для успешной сдачи ЕГЭ по информатике!

Книга разложит по полочкам все темы школьного курса за 5—11 классы: ты сможешь запросто повторить уже изученный материал и получить новые знания. Только действительно нужная для экзамена теория по разделам информатики преподносится наглядно и понятно, а также сопровождается разными типами экзаменационных заданий с ответами и пояснениями.

Также пособие будет полезно учителям и репетиторам при планировании и проведении занятий.

УДК 373.5:004
ББК 32.81я721

ISBN 978-5-04-222326-6

СОДЕРЖАНИЕ

ОТ АВТОРА

В сборнике собраны типовые модели заданий ЕГЭ по информатике вместе с подробными решениями программированием. К каждому типу задания приводится необходимая теоретическая база — от основ программирования на Python до конкретных способов решений.

Пособие можно использовать как универсальный справочник при подготовке к ЕГЭ. Чтобы извлечь максимум пользы, установите всё требуемое программное обеспечение (интерпретатор Python и др.) и воспроизводите решения самостоятельно, шаг за шагом.

Наборы входных данных для заданий № 9 и № 17 размещены по ссылке (см. ниже).

Каждый тип экзаменационного задания разобран последовательно и системно. К нему приводится:

1) краткое введение в теорию;
2) базовый пример;
3) ключевые экзаменационные прототипы;
4) подробный комментарий к алгоритму.

Не ограничивайтесь чтением — запускайте код, проверяйте гипотезы, экспериментируйте. Глубокое понимание, а не заученные шаблоны, приносит высокий результат. Придумывайте альтернативные решения и оптимизируйте приведённые — так вы быстрее отточите навыки и почувствуете уверенность.

Помните: программирование кажется сложным лишь вначале. С каждым решённым примером барьер будет снижаться, а скорость — расти!

Материалы для выполнения заданий № 9 и № 17 вы можете скачать по ссылке

https://addons.eksmo.ru/it/EGE_Inf.7z

или скачать по QR-коду

СТРУКТУРА ЭКЗАМЕНА И СИСТЕМА ОЦЕНИВАНИЯ

- ✓ **Формат.** ЕГЭ по информатике проходит исключительно в компьютерной форме. На рабочей станции уже установлены текстовый редактор, электронные таблицы и несколько сред разработки для языков программирования Python 3.x, PascalABC.NET, C++, Java, C#.
- ✓ **Время.** На выполнение всей работы отводится **3 ч 55 мин (235 минут)**.
- ✓ **Объём.** Всего требуется решить **27 заданий**; номера в КИМ не отражают уровень трудности.
- ✓ **Баллы.** Задания № 1–25 оцениваются в **1 первичный балл** каждое. Задания № 26–27 приносят по **2 первичных балла**.
- ✓ **Форма ответа.** Все решения вводятся в виде одного или нескольких чисел либо строк.
- ✓ **Проверка.** Ответы проверяются автоматически, поэтому критически важно соблюдать требуемый формат — лишние пробелы и разный регистр могут привести к потере баллов.

Следуя рекомендациям и регулярно тренируясь на предоставленных материалах, вы сможете уверенно освоить большую часть экзаменационных заданий и показать высокий результат на ЕГЭ.

Раздел 1

ОСНОВЫ ПРОГРАММИРОВАНИЯ

Переменные и типы данных в Python

В Python есть несколько базовых типов данных:

Тип	Назначение	Пример
`int`	Целые числа	5, −10, 0
`float`	Вещественные числа (дробные)	3.14, −2.5
`str`	Строки (текст)	“Привет”, ‘Python’
`bool`	Логические значения	True, False

Что такое переменная

Переменная — это коробка с подписью, в которую можно положить какое-то значение.

Представьте, что у вас есть коробка с надписью x. Вы положили в неё число 5. Теперь x — это 5.

Переменная — это имя, которое ссылается на значение. В Python переменные не требуют объявления типа данных.

Как создать переменную в Python

Чтобы создать переменную, нужно написать её имя, знак = и значение:

```
x = 5
```

- ✓ `x` — это имя переменной (можете придумать любое осмысленное название);
- ✓ `=` — это знак **присваивания** (он **не** означает «равно», как в математике!);
- ✓ `5` — значение, которое сохраняется в переменной.

Знак = в программировании означает «положить значение справа в переменную слева».

Примеры переменных:

```
age = 18 # возраст (целое число)
pi = 3.14159 # число Пи (дробное)
name = "EGE_INF" # название предмета (строка)
is_student = True # логический флаг (да/нет)
```

Основные правила именования переменных

- ✓ Название переменной **должно начинаться** с буквы или подчёркивания _.
- ✓ Можно использовать буквы, цифры и подчёркивания.
- ✓ Название **не может начинаться с цифры**.
- ✓ Переменные должны иметь **понятные имена** (не `a, b,` а `age, price, user_name`).

Так нельзя создавать переменную:

```
2user = "Неверно"
```

А вот так можно:

```
user2 = "Верно"
_user = "Тоже верно"
```

Ввод и вывод данных

Что такое ввод и вывод

Ввод — это получение данных от пользователя.
Вывод — это отображение информации пользователю.

- ✓ Ввод: программа спрашивает у человека данные.
- ✓ Вывод: программа показывает человеку результаты работы.

Как реализовать ввод данных в Python

Для ввода используется команда `input()`. Она может выводить сообщение и ждёт, пока пользователь введёт текст.

```
name = input("Введите ваше имя: ")
```

Что происходит здесь?

- ✓ На экране появится фраза: `Введите ваше имя:`
- ✓ Программа **ставится на паузу** и будет ждать ответа.
- ✓ Введённый текст сохраняется в переменной `name`.

Важно знать!

Всё, что вводится через `input()`, всегда сохраняется как строка `(str)`, даже если пользователь набрал число.

Как реализовать вывод данных в Python

Для вывода информации используется команда `print()`

```
print("Привет,", name)
```

- ✓ `print()` выводит на экран то, что вы ему передадите.
- ✓ Можно передавать несколько значений через запятую — они будут разделены пробелами.

Разберёмся на простом **примере.**

Если пользователь введёт `18`, то программа напечатает:

```
age = input("Сколько вам лет? ")
print("На данный момент вам", age, "лет!")
```

В данном случае `age` — это **строка**, а не число. Поэтому, если вы захотите прибавить 1, нужно сначала превратить строку в число.

Преобразование типов данных

Чтобы сделать из строки число, используем функцию `int()`:

```
age = int(input("Сколько вам лет? "))
print("Через год вам будет", age + 1)
```

Теперь:

- ✓ пользователь вводит, например, 18;
- ✓ `age` становится числом `18 (int)`;
- ✓ `age + 1` будет `19`.

Арифметические операции

Что такое арифметические операции

Арифметические операции позволяют выполнять вычисления с числами: сложение, вычитание, умножение, деление, нахождение остатка при делении, возведение в степень и другие действия.

Базовые арифметические операции в Python

Операция	Символ	Пример	Результат
Сложение	+	2 + 3	5
Вычитание	–	5 – 2	3
Умножение	*	3 * 4	12
Деление	/	10 / 2	5.0
Целочисленное деление	//	10 // 3	3
Остаток от деления	%	10 % 3	1
Возведение в степень	**	2 ** 3	8

Примеры операций:

```
a = 9
b = 4

print(a + b) # сложение: 13
print(a - b) # вычитание: 5
print(a * b) # умножение: 36
print(a / b) # деление: 2.25
print(a // b) # целочисленное деление: 2
print(a % b) # остаток от деления: 1
print(a ** b) # возведение в степень: 6561
```

Условные операторы и логические операции

Операторы сравнения

Операторы сравнения используются для проверки различных условий. Вот самые распространённые:

- ✓ == — равно
- ✓ != — не равно

- ✔ > — больше
- ✔ < — меньше
- ✔ >= — больше или равно
- ✔ <= — меньше или равно

Условные операторы позволяют программе принимать решения в зависимости от выполнения некоторого условия.

В Python для этого используются операторы `if`, `elif` и `else`.

Общий синтаксис условных операторов

```
if <условие>:           # Если условие истинно, выполняется этот блок кода
    <операции, если условие истинно>
elif <другое условие>: # Если предыдущее условие ложно, но это истинно
    <операции, если это условие истинно>
else:                   # Если все предыдущие условия ложны
    <операции, если все условия ложны>
```

- ✔ `if` — выполняет блок кода, если условие истинно;
- ✔ `elif` — проверяет следующее условие, если предыдущее ложное (этот блок является необязательным);
- ✔ `else` — выполняет блок кода, если все условия (до этого) ложные. Данный блок также является необязательным.

Пример:

```
age = 20
if age >= 18:
    print("Вы уже совершеннолетний!")
else:
    print("Вы ещё не совершеннолетний!")
```

Логические операторы

Логические операторы позволяют комбинировать несколько условий в одно:

- ✓ `and` — условие истинно, если все операнды истинны;
- ✓ `or` — условие истинно, если хотя бы один из операндов истинный;
- ✓ `not` — инвертирует логическое значение, то есть делает ложное истинным и наоборот.

Пример с `and`:

```
age = 25
is_student = False
if age >= 18 and is_student:
    print("Вы совершеннолетний студент!")
else:
    print("Вы не совершеннолетний студент!")
```

Пример с `or`:

```
age = 15
is_student = True
if age < 18 or is_student:
    print("Вам младше 18 или вы студент!")
else:
    print("Вы не студент, и вы не младше
    18 лет!")
```

Пример с `not`:

```
is_raining = False
if not is_raining:
    print("Сегодня не дождливо!")
```

Вложенные условия. Иногда внутри условного оператора нужно проверять дополнительные условия. Это называется вложенными условиями.

Пример:

```
age = 20
is_student = True

if age >= 18:
    if is_student:
        print("Вы совершеннолетний студент!")
    else:
        print("Вы совершеннолетний, но не
        студент!")
else:
    print("Вы не совершеннолетний!")
```

Вначале мы проверяем, является ли человек взрослым, а затем, если это так, проверяем, является ли он студентом.

Циклы

Циклы позволяют повторять блок кода несколько раз.
В Python есть два основных типа циклов: `for` и `while`.

Цикл `for`

Цикл `for` используется для перебора элементов в последовательностях, таких как списки, строки, диапазоны и другие итерируемые объекты.

Синтаксис:

```
for <переменная> in <последовательность>:
    <операции, которые выполняются с каждым
    элементом>
```

Пример:

```
for i in range(5): # range(5)создаёт
последовательность чисел от 0 до 4
    print(i)
```