НейроМиша

Материал для подготовки к ЕГЭ
neuromisha.ru

Python-заготовки для ЕГЭ

Десять заготовок, из которых собирается программная часть ЕГЭ по информатике. Каждая написана под свой образец условия, прогнана и снабжена настоящим выводом — числа в разделах не придуманы, а получены запуском. Меняете одну-две строки под своё условие и запускаете.

Информатика | Задание 12, 13, 15, 16, 17, 24, 27 | Обновлено 5 сентября 2026

Как устроен экзамен

ЕГЭ по информатике сдают в компьютерной форме. Каждый получает рабочее место со средой программирования, редактором электронных таблиц и текстовым процессором, но без доступа в сеть. Заданий 27, времени 235 минут.

Все ответы краткие: число или последовательность символов. Номера с первого по двадцать пятый стоят по одному баллу, двадцать шестой и двадцать седьмой — по два, максимум составляет 29 первичных баллов. Развёрнутых решений нет, программу никто не читает — в бланк идёт только результат её работы.

Одиннадцать заданий решаются на компьютере: 3, 9, 16, 17, 18, 22, 23, 24, 25, 26 и 27. К ним прилагаются файлы в открытых форматах — таблица, текстовый документ или обычный текстовый файл. Язык можно взять любой из разрешённых: Python, C++, C#, Java, Pascal.

По проекту спецификации 2027 года число заданий, баллы и продолжительность прежние, но переписаны формулировки трёх номеров: задание 10 теперь о маске подсети, 13 — об анализе хода исполнения алгоритма, 23 — об анализе графов. Ответ на задание 27 записывается одной строкой из двух чисел вместо двух строк. Проект обсуждается до 30 сентября 2026 года, формулировки ещё могут измениться.

Перебор слов и комбинаций

Образец. Сколько слов длины 5 можно составить из букв A, B, V, G, если буква A встречается ровно один раз и слово не начинается с B?

```
from itertools import product

alphabet = 'ABVG'
count = 0
for word in product(alphabet, repeat=5):
    w = ''.join(word)
    if w.count('A') == 1 and not w.startswith('B'):
        count += 1
print(count)
```

Выводит: 297.

Где менять: строку alphabet, число repeat и условие внутри if. Перебор идёт по всем словам подряд, поэтому любое условие описывается напрямую, без комбинаторных формул. При длине слова больше девяти и алфавите из четырёх букв перебор станет слишком долгим — тогда нужен подсчёт по шагам.

Системы счисления

Образец. Найдите сумму цифр числа $5^{12} + 5^7 - 30$ в пятеричной записи.

```
def to_base(n, base):
    digits = ''
    while n > 0:
        digits = str(n % base) + digits
        n //= base
    return digits or '0'

n = 5 ** 12 + 5 ** 7 - 30
s = to_base(n, 5)
print(s)
print(sum(int(d) for d in s))
```

Выводит: 1000004444340 и 24.

Где менять: выражение для n и основание в вызове to_base. Функция работает для оснований до десяти; для шестнадцатеричной записи есть встроенная hex. Печать самой записи оставьте: по ней сразу видно, правильно ли понято условие.

Логика с делимостью

Образец. Найдите наибольшее натуральное A, при котором выражение «x кратно 20 или x кратно 30 влечёт x кратно A» истинно для любого натурального x.

```
def formula(x, a):
    return (x % 20 != 0 and x % 30 != 0) or (x % a == 0)

answer = 0
for a in range(1, 101):
    if all(formula(x, a) for x in range(1, 10001)):
        answer = a
print(answer)
```

Выводит: 10.

Где менять: тело функции formula. Импликация «P влечёт Q» переписывается как «не P или Q» — именно это сделано в первой строке. Диапазон проверки берите с запасом: если ответ упирается в границу перебора, увеличьте её и запустите снова.

Рекуррентная функция

Образец. Функция задана так: F(n) = n при n не больше 3 и F(n) = F(n − 1) + 2·F(n − 3) при n больше 3. Найдите F(20).

```
from functools import lru_cache

@lru_cache(maxsize=None)
def f(n):
    if n <= 3:
        return n
    return f(n - 1) + 2 * f(n - 3)

print(f(20))
print([f(n) for n in range(1, 9)])
```

Выводит: 24227 и список [1, 2, 3, 5, 9, 15, 25, 43].

Где менять: базовый случай и правило перехода. Строка с lru_cache обязательна: без неё вычисление одних и тех же значений повторяется и при больших аргументах программа зависает. Вторая печать — проверка: сверьте первые значения с посчитанными вручную.

Чтение файла с числами

Образец. В файле 17-sample.txt первая строка — количество чисел, дальше сами числа по одному в строке: 12 чисел 14, 3, −5, 21, 8, 7, 2, 40, 6, 35, 9, 11. Найдите количество пар соседних чисел, произведение которых кратно 7, и наибольшее из таких произведений.

```
with open('17-sample.txt', encoding='utf-8') as f:
    n = int(f.readline())
    data = [int(f.readline()) for _ in range(n)]

count = 0
best = None
for i in range(len(data) - 1):
    p = data[i] * data[i + 1]
    if p % 7 == 0:
        count += 1
        if best is None or p > best:
            best = p
print(count, best)
```

Выводит: 7 315.

Где менять: условие внутри if. Максимум специально начинается со значения None, а не с нуля: если все подходящие произведения окажутся отрицательными, нулевая заготовка даст неверный ответ. Цикл идёт до len(data) − 1, иначе последняя пара выйдет за границу списка.

Перебор диапазона с условием

Образец. Найдите три наибольших числа в диапазоне от 100 000 до 120 000, которые делятся на 19 и у которых сумма цифр кратна 7.

```
found = []
for x in range(100000, 120001):
    if x % 19 == 0 and sum(int(d) for d in str(x)) % 7 == 0:
        found.append(x)

found.sort(reverse=True)
print(len(found))
for x in found[:3]:
    print(x, sum(int(d) for d in str(x)))
```

Выводит: 146, затем три строки — 119890 28, 119719 28, 119548 28.

Где менять: границы range и условие в if. Печать длины списка нужна для проверки: пустой или подозрительно короткий список означает ошибку в условии, а не отсутствие ответа. Сортировка по убыванию задаётся одним аргументом reverse.

Строковый исполнитель

Образец. Исполнитель работает со строкой и умеет заменять первое вхождение подстроки. Начальная строка — 50 двоек подряд, затем три единицы. Программа: пока находится «22», заменять его на «1»; затем пока находится «111», заменять его на «2». Что получится?

```
s = '2' * 50 + '111'

def editor(s, v, w):
    while v in s:
        s = s.replace(v, w, 1)
    return s

s = editor(s, '22', '1')
s = editor(s, '111', '2')
print(s)
print(len(s), sum(int(c) for c in s))
```

Выводит: 2222222221, затем 10 19.

Где менять: начальную строку и пары замен. Третий аргумент replace, равный единице, обязателен: он заменяет только первое вхождение, как это делает исполнитель. Без него замена пойдёт по всей строке сразу и ответ разойдётся с эталоном.

Подсчёт числа программ

Образец. У исполнителя две команды: прибавить 1 и умножить на 2. Сколько программ переводят число 2 в число 30, проходя через 12 и не проходя через 8?

```
FORBIDDEN = {8}

def paths(start, finish):
    count = {start: 1}
    for x in range(start + 1, finish + 1):
        if x in FORBIDDEN:
            count[x] = 0
            continue
        total = count.get(x - 1, 0)
        if x % 2 == 0:
            total += count.get(x // 2, 0)
        count[x] = total
    return count[finish]

first = paths(2, 12)
second = paths(12, 30)
print(first, second, first * second)
```

Выводит: 5 5 25.

Где менять: множитель в двух местах (x % 2 и x // 2) и содержимое FORBIDDEN. Участки до обязательной точки и после неё считаются отдельно, а результаты перемножаются — это самая частая ошибка номера: результаты складывают. Метод get возвращает ноль для недостижимых чисел, поэтому проверок на границы не нужно.

Один проход по длинной строке

Образец. В файле 24-sample.txt лежит строка ABCABBACBCCABACABABCBBACA. Найдите длину самой длинной подстроки, в которой нет двух одинаковых соседних символов.

```
with open('24-sample.txt', encoding='utf-8') as f:
    s = f.read().strip()

best = 1
current = 1
for i in range(1, len(s)):
    if s[i] != s[i - 1]:
        current += 1
    else:
        current = 1
    if current > best:
        best = current
print(len(s), best)
```

Выводит: 25 11.

Где менять: условие сравнения соседних символов. Вызов strip обязателен: без него символ перевода строки в конце файла попадёт в подсчёт. Максимум обновляется на каждом шаге, а не после цикла, — иначе теряется участок, которым строка заканчивается. Проход по строке ровно один, поэтому файл на миллион символов обрабатывается за секунды.

Анализ данных: два числа одной строкой

Образец. В файле 27-sample.txt каждая строка — номер дня и выручка за него: 1 120, 2 340, 3 90, 4 410, 5 400, 6 75, 7 260, 8 300. Найдите наибольшую суммарную выручку за два подряд идущих дня и номер первого дня этой пары.

```
days = []
with open('27-sample.txt', encoding='utf-8') as f:
    for line in f:
        line = line.strip()
        if not line:
            continue
        day, money = line.split()
        days.append((int(day), int(money)))

best_day = 0
best_sum = -1
for i in range(len(days) - 1):
    total = days[i][1] + days[i + 1][1]
    if total > best_sum:
        best_sum = total
        best_day = days[i][0]
print(best_day, best_sum)
```

Выводит: 4 810.

Где менять: разбор строки в split и правило отбора. Пропуск пустых строк через continue снимает главную проблему чтения файлов: последняя строка часто пустая. Начальная сумма равна минус единице, а не нулю, чтобы отрицательные величины тоже находились. Ответ печатается одной строкой из двух чисел — так его требует записывать формулировка 2027 года.

Как готовить заготовки к экзамену

1. Заготовку нельзя принести с собой: на экзамене выдают чистое рабочее место. Тренируйте набор по памяти, а не копирование — цель в том, чтобы любая из этих программ писалась за две-три минуты.
2. Всегда проверяйте программу на маленьком файле, где ответ считается вручную. Если она не воспроизводит известный результат, на большом файле она тем более ошибётся.

3. Держите один скелет чтения файла на все задания: открыть, убрать перевод строки, превратить в числа. Разница между номерами тогда сводится к одной строке условия.
4. Печатайте промежуточные величины — длину списка, первые значения функции, саму строку. Лишняя печать стоит секунды, а ошибку в понимании условия показывает сразу.
5. Помните про два ограничения по скорости: в номере 24 нужен один проход по строке, в номере 25 — перебор делителей до квадратного корня. Всё остальное решается перебором в лоб.

Типичные ошибки

- Инициализируют максимум нулём, когда подходящие значения могут быть отрицательными. Берите первое найденное значение или None.
- Считают, что ноль не кратен заданному числу. Ноль кратен любому, и это меняет ответ в заданиях на делимость.
- Забывают убрать перевод строки в конце файла и получают лишний символ в подсчёте.
- Выходят за границу списка на последней паре: цикл должен идти до len(data) − 1, а не до len(data).
- Перебирают все подстроки вложенными циклами. На файле в миллион символов такая программа не досчитает за отведённое время.
- Складывают количества путей на двух участках вместо перемножения в задании о числе программ.
- Пишут рекурсию без базового случая или без запоминания значений и получают либо бесконечный спуск, либо зависание.
- Меняют местами два числа в ответе на задания 26 и 27: за перестановку ставят один балл вместо двух.

Источники

- ФИПИ: демоверсии, спецификации, кодификаторы — проект КИМ ЕГЭ 2027 по информатике: число заданий, баллы, продолжительность, перечень заданий с использованием компьютера
- ФИПИ: планируемые изменения в КИМ ЕГЭ 2027 года — новые формулировки заданий 10, 13 и 23 и форма записи ответа на задание 27
- Рособрнадзор: методические рекомендации по проведению ЕГЭ в ППЭ — состав программного обеспечения на рабочем месте и запрет собственных носителей
- ФИПИ: открытый банк заданий ЕГЭ — форматы прилагаемых файлов и требования к записи ответа