Общеобразовательная автономная некоммерческая организация
средняя общеобразовательная школа
«Пенаты»

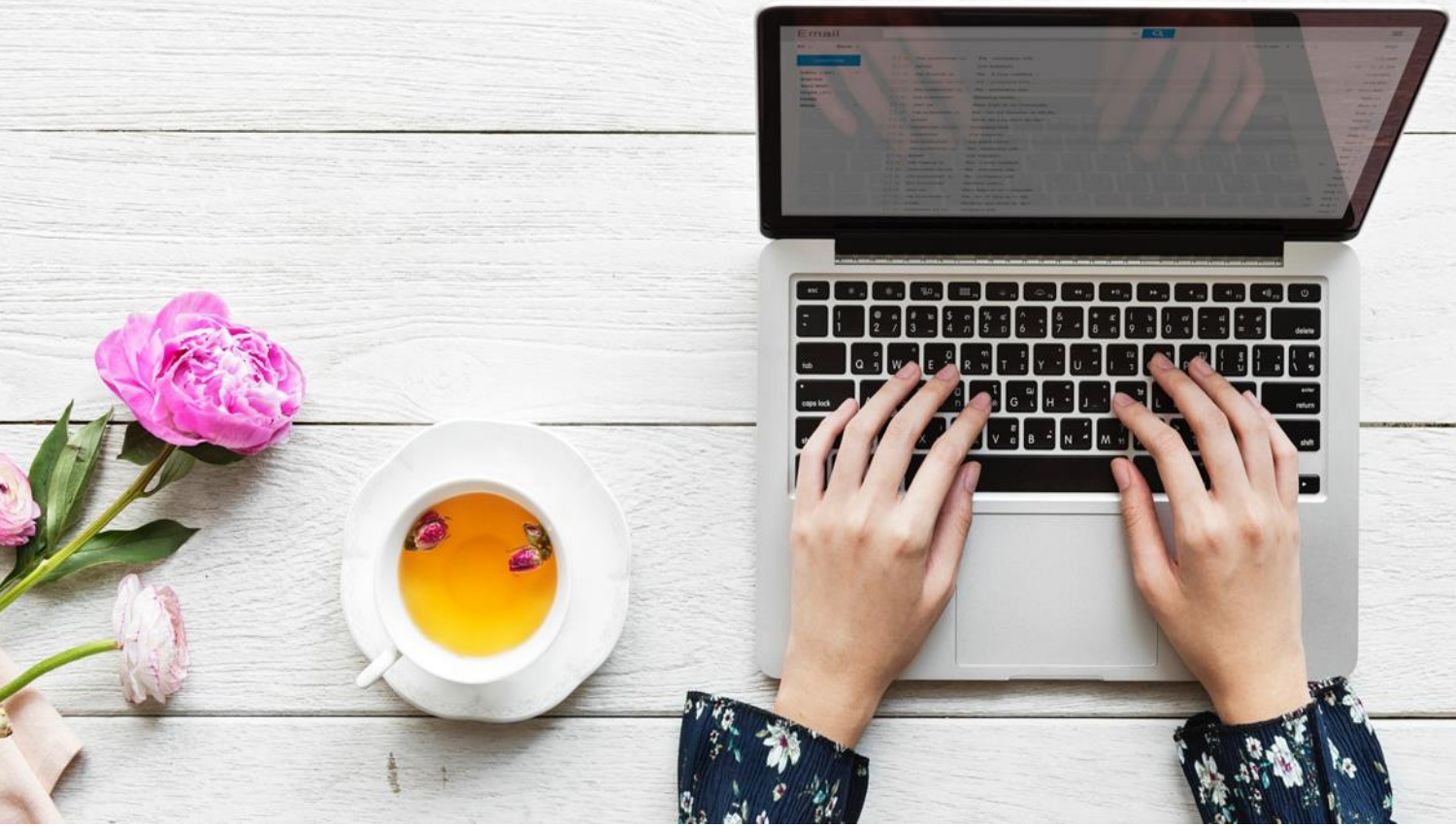
Возможности Python при решении задач ЕГЭ

Автор:

Иванов Алексей, 11А класс

Руководитель проекта:

Лифанова Ольга Александровна



ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Изучение информатики, как и любого другого предмета, является сложным и долгим процессом. Требуется знать много терминов, понимать, что требуется найти и как решить поставленную задачу. При этом в информатике есть своя особенность – языки программирования, один из них, рекомендованный министерством просвещения для изучения в школе, это язык Python. К сожалению, не все возможности этого языка входят в школьную программу. Для того, чтобы решать сложные задачи, поставленные ЕГЭ, необходимо знать гораздо больше, ведь в этом языке есть громадная коллекция библиотек с разнообразными встроенными функциями, которые помогут упростить составление алгоритма и нахождения пути решения задач. К тому же в Python меня заинтересовали такие структуры данных как словари и множества, списки и кортежи, которые так же полезны для упрощения кода.

ИЗУЧЕНИЕ ЗАДАЧ ЕГЭ

Некоторые задачи из этого списка можно решить аналитически, но написание программного кода значительно упрощает этот процесс

Задача № 8

Эта задача на перебор слов и на системы исчисления.

Примеры:

«Шифр кодового замка представляет собой последовательность из пяти символов, каждый из которых является цифрой от 1 до 4. Сколько различных вариантов шифра можно задать, если известно, что цифра 1 встречается ровно два раза, а каждая из других допустимых цифр может встречаться в шифре любое количество раз или не встречаться совсем?»

«Определите количество шестизначных чисел, записанных в девятеричной системе счисления, в записи которых ровно одна цифра 4 и ровно две нечётные цифры.»

«Все 4-буквенные слова, составленные из букв Н, Р, Т, У, записаны в алфавитном порядке. Вот начало списка:

1. НННН
2. НННР
3. НННТ
4. НННУ
5. ННРН

Запишите слово, которое стоит на 215-м месте от начала списка.»

Задача № 12

Эта задача на выполнение алгоритмов для исполнителей. В этой задаче проверяется умение исполнить алгоритм с фиксированным набором команд

Пример:

12 Исполнитель Редактор получает на вход строку цифр и преобразует её. Редактор может выполнять две команды, в обеих командах *v* и *w* обозначают цепочки цифр.

А) **заменить** (*v*, *w*).

Эта команда заменяет в строке первое слева вхождение цепочки *v* на цепочку *w*. Например, выполнение команды

заменить (111, 27)

преобразует строку 05111150 в строку 0527150.

Если в строке нет вхождений цепочки *v*, то выполнение команды **заменить** (*v*, *w*) не меняет эту строку.

Б) **нашлось** (*v*).

Эта команда проверяет, встречается ли цепочка *v* в строке исполнителя Редактор. Если она встречается, то команда возвращает логическое значение «истина», в противном случае возвращает значение «ложь». Строка исполнителя при этом не изменяется.

Дана программа для Редактора:

```
НАЧАЛО
  ПОКА НЕ нашлось (00)
      заменить (02, 101)
      заменить (11, 2)
      заменить (12, 21)
      заменить (010, 00)
  КОНЕЦ ПОКА
КОНЕЦ
```

Известно, что исходная строка *A* содержала ровно два нуля – на первом и на последнем месте, а также поровну единиц и двоек, при этом всего в строке *A* было более 140 цифр. После выполнения данной программы получилась строка *B*, сумма цифр которой оказалась простым числом. Какое наименьшее количество единиц могло быть в строке *A*?

Ответ: _____.

Задача № 15

Эта задача на подбор параметра.

Пример:

«Обозначим через m&n поразрядную конъюнкцию неотрицательных целых чисел m и n. Например, 14&5 = 11102&01012 = 01002 = 4. Для какого наименьшего неотрицательного целого числа A формула (x&114 ≠ 0 ∨ x&94 ≠ 0) → (x&73 = 0 → x&A ≠ 0) тождественно истинна (т. е. принимает значение 1 при любом неотрицательном целом значении переменной x)?»

«Для какого наибольшего целого числа A формула ((x ≤ 9) →(x · x ≤ A)) ∧ ((y · y ≤ A) → (y ≤ 9)) тождественно истинна, то есть принимает значение 1 при любых целых неотрицательных x и y?»

Задача № 16

Эта задача на рекурсию.

Пример:

«Алгоритм вычисления значения функции F(n), где n — целое неотрицательное число, задан следующими соотношениями:

F(0) = 0;

F(n) = F(n − 1) + 1, если n нечётно;

F(n) = F(n / 2), если n > 0 и при этом n чётно.

Укажите количество таких значений n < 1 000 000 000, для которых F(n) = 3.»

«(№ 6241) (PRO100 ЕГЭ) Алгоритм вычисления функции F(n), где n – натуральное число, задан следующими соотношениями:

F(n) = 1, при n = 1,

F(n) = n + F(n - 1), если n > 1.

Чему равно значение выражения F(2023) - F(2019)»

Задача № 19–21

Эта задача на теорию игр.

Пример:

«(№ 4725) (И. Осипов) Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат три кучи камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в одну из куч (по своему выбору) три камня или увеличить количество камней в куче в два раза. Например, пусть в первой куче 10 камней, во второй 7, а в третьей 4 камня; такую позицию в игре будем обозначать (10, 7, 4). Тогда за один ход можно получить любую из шести позиций: (13, 7, 4), (20, 7, 4), (10, 10, 4), (10, 14, 4), (10, 7, 7), (10, 7, 8). Для того чтобы делать ходы, у каждого игрока есть неограниченное количество камней. Игра завершается в тот момент, когда суммарное количество камней в кучах становится не менее 71. Победителем считается игрок, сделавший последний ход, т. е. первым получивший такую позицию, что в кучах всего будет 71 или больше камней. В начальный момент в первой куче было семь камней, во второй куче пять камней, в третьей куче – S камней; $1 \leq S \leq 58$.

Ответьте на следующие вопросы:

Вопрос 1. При некотором значении S Ваня одержал победу свои первым ходом после неудачного хода Пети. Укажите минимальное значение S, при котором это возможно.

Вопрос 2. Найдите минимальное и максимальное значения S, при которых Петя выигрывает вторым ходом при любом ходе Вани.

Вопрос 3. Найдите значение S, при котором одновременно выполняются два условия: а) у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре

Пети; б) у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.»

Задача № 23

Эта задача на динамическое программирование.

Пример:

«(№ 6171) (М. Шагитов) У исполнителя Калькулятор имеются три команды, которым присвоены номера:

1. Прибавь 3

2. Умножь на 4

3. Умножь на 5

Выполняя первую из них, исполнитель увеличивает число на экране на 3, выполняя вторую – умножает на 4, выполняя третью – умножает на 5. У исполнителя есть запас энергии, который в начальный момент равен 1000. При выполнении каждой команды над текущим числом на экране исполнитель расходует энергию равную количеству цифр этого числа. Сколько существует различных программ, преобразующих число 1 в число 5500, после выполнения которых запас энергии точно равен 0?»

Задача № 25

Эта задача на обработку целочисленной информации.

Пример:

«Маска числа – это последовательность цифр, в которой могут встречаться специальные символы «?» и «*». Символ «?» означает ровно одну произвольную цифру, символ «*» означает произвольную (в том числе пустую) последовательность цифр. Пример: Маске 123*4?5 соответствуют числа 123405 и 12376415. Найдите все натуральные числа, не

превышающие 1010, которые соответствуют маске 1?3948*5 и при этом без остатка делятся на 3013. В ответе запишите все найденные числа в порядке возрастания.»

Задача № 27

Эта задача на обработку большого количества данных.

Пример:

«В текстовом файле записан набор пар натуральных чисел, не превышающих 10 000. Необходимо выбрать из набора некоторые пары так, чтобы второе число в каждой выбранной паре было нечётным, сумма бóльших чисел во всех выбранных парах была чётной, а сумма меньших — нечётной. Какую наибольшую сумму чисел во всех выбранных парах можно при этом получить?

Входные данные:

Файл A

Файл B

Первая строка входного файла содержит целое число N — общее количество пар в наборе. Каждая из следующих N строк содержит пару чисел.

Пример входного файла:

4

5 3

7 15

7 14

12 9

В данном случае есть три подходящие пары: (5, 3), (7, 15) и (12, 9). Пара (7, 14) не подходит, так как в ней второе число чётное. Чтобы удовлетворить требования, надо взять пары (5, 3), (7, 15) и (12, 9). Сумма бóльших чисел в этом случае равна 32, сумма меньших равна 19. Общая сумма равна 51. В ответе надо указать число 51.

Вам даны два входных файла (A и B), каждый из которых имеет описанную выше структуру. В ответе укажите два числа: сначала значение искомой суммы для файла A, затем для файла B.»

ТИПЫ ДАННЫХ И МЕТОДЫ РАБОТЫ С НИМИ

all()

Возвращает True в случае того, если при всех заданных значениях переменной данная функция выполняется.

```
all(x > 50 for x in range(100, 151))

# True
```

any()

Возвращает True в случае того, если при хотя бы одном заданном значении переменной данная функция выполняется.

```
any(x == 100 for x in range(1001))

# True
```

Списки

Списки в Python – упорядоченный изменяемый набор объектов произвольных типов, пронумерованных от 0. Нумерация элемента в списки называется его индексом. Они используются для хранения и работы с данными. Длину списка можно использовать len().

Задать список можно различными способами:

a = [] – пустой список, с переменой a можно работать как со списком

a = [15, 6, 89] ; a[0] = 15 ; a[1] = 6 ; a[2] = 89

a = [0] * 4 – список длинной 4, где все значения – 0

a = [8] + [5]*3 – список длинной 4, где первое значение 8, остальные – 5

Множества

Множества в Python – список, где нет повторяющихся переменных. Список можно превратить в множество при помощи set().

.sort()

Сортирует список по возрастанию. Если написать .sort(reverse), то список будет отсортирован по убыванию.

```
a = [15, 6, 89]

a.sort()

print(a)

# [6, 15, 89]
```

.append()

Позволяет расширить список, добавив на конец списка переменную.

```
a = []

a.append(600)

a.append(89)

print(a)

# [600, 89]
```

.remove()

Позволяет удалить первое найденное вхождение заданной переменной.

```
a = [6, 89, 15, 89]

a.remove(89)

print(a)
```

```
# [6, 15, 89]
```

.count()

Возвращает количество вхождений данного значение.

```
a = [7, 97, 7, 4, 5, 7]

b = a.count(7)

print(b)

# 3
```

.index()

Возвращает индекс первого вхождения данной значения.

```
a = [4, 5, 4, 78]

b = a.find(4)

print(b)

# 0
```

БИБЛИОТЕКИ

Библиотеки имеют заранее созданные функции и методы.

Functools

Эта библиотека используется для улучшения производительности программ, в частности в функциях, которые вызывают сами себя.

.lru_cash

functools.lru_cash(maxsize) – функция, которая сохраняет в памяти первые maxsize уникальных входов в созданную функцию и потом использует эту информацию чтобы сразу выдать ответ при тех же вводных данных.

Itertools

Эта библиотека используется для создания различных комбинаций при заданных условиях. (Далее r – натуральное число, iterable – набор символов)

.product

product(iterable, repeat = r) — функция, создающая всевозможные перестановки из iterable с r количеством элементов.

Пример:

```
import itertools

for i in itertools.product('12', repeat=2):

    print(i)
```

Вывод: ('1', '1'); ('1', '2'); ('2', '1'); ('2', '2')

.permutations

permutations(iterable, r) — функция, которая возвращает перестановки без повторений длиной r из iterable.

Пример:

```
import itertools

for i in itertools.permutations('12', 2):

    print(i)
```

Вывод: ('1', '2'); ('2', '1')

.combinations

combinations(iterable, r) — функция, которая возвращает комбинации длиной r из iterable без повторяющихся элементов.

Пример:

```
import itertools

for i in itertools.combinations('12', 2):

    print(i)
```

Вывод: ('1', '2')

Math

Эта библиотека содержит функции, которые выполняют различные математические операции

.ceil

Эта функция возвращает округление до ближайшего большего числа.

.sqrt

Эта функция возвращает квадратный корень из числа в формате float.

Fnmatch

Эта библиотека вводит понятие масок и даёт возможность с ними работать:

? – один любой символ

* – любое количество любых символов

ПРИМЕРЫ РЕШЕНИЯ ЗАДАЧ

Я выяснил, что компьютерного ЕГЭ по информатике предлагаются десять типов заданий на следующие темы.

1. Вычисления
2. Решение уравнений численными методами
3. Перебор целых чисел. (Разбиение числа на цифры)
4. Перебор чисел. Проверка делимости
5. Перебор целых чисел. Количество делителей
6. Символьные строки. Цепочки символов
7. Функции двух аргументов. Таблицы значений
8. Электронные таблицы. Встроенные функции (не решается средства Python)
9. Рекурсия. Рекурсивные функции
10. Исследование моделей. Оптимизация

1. Пример задания на вычисление

С помощью программы Калькулятор или электронных таблиц вычислите значение выражения $\sqrt{1+\cos(3{,}53\pi)\cdot 10}\cdot 310$. В ответе запишите только целую часть результата. Можно также написать программу.

Программа на языке Python

```
from math import sqrt, cos, pi
```

```
print( sqrt(1 + cos(3.53*pi)*10)*310 )
```

Ответ: 431.

Для решения данного задания, нужно знать правила записи математических функций на языке Python. В связи с невозможностью записи некоторых стандартных математических функций с клавиатуры персонального компьютера в языке Python существуют так называемые встроенные функции, с помощью которых пользователь записывает арифметические выражения.

Основные математические функции языка Python представлены в таблице 1. Прежде чем использовать математические функции, необходимо в начале программы написать инструкцию import math, однако тогда перед упоминанием каждой функции необходимо будет добавлять имя модуля - math, например, y=math.sin(x). Другой способ, который позволит избежать многократного вызова модуля math, - сделать следующую запись в начале программы: from math import *.

Таблица 1. Общие математические функции модуля Math

Запись на Python	**Действие**
math.sin (x)	Возвращает значение функции Sin от числа x
math.cos (x)	Возвращает значение функции Cos от числа x
math.tan (x) или math.sin (x) / math.cos (x)	Возвращает значение функции Tg от числа x
math.cos (x) / math.sin (x)	Возвращает значение функции Ctg от числа x
math.abs (x)	Возвращает абсолютную величину числа x
math.exp (x)	Возвращает результат возведения числа e в степень X
math.Log lp (x)	Возвращает натуральный логарифм от x+1
math.sqrt (x)	Возвращает результат извлечения квадратного корня числа x
math.log (x)	Возвращает логарифм числа x по основанию 10
math.cos (x) * math.cos (x)	Возвращает результат возведения функции Cos x в квадрат
math.acos (x)	Возвращает значение функции арккосинус от числа x

math.asin (x)	Возвращает значение функции арксинус от числа x
math.atan (x)	Возвращает значение функции арктангенс от числа x
Pi	Возвращает 3.141592653589793
math.degrees(x)	Преобразует радианы в градусы
math.radians(x)	Преобразует градусы в радианы
math.floor(x)	Возвращает значение, округленное до ближайшего меньшего целого
math.ccil(x)	Возвращает значение, округленное до ближайшего большего целого
math.factorial(x)	Возвращает факториал числа. 3 != 1 *2*3

В таблице 2 представлены некоторые встроенные функции для работы с числами, не требующие подключения модуля math.

Таблица 2. Функции для работы с числами

Запись на Python	**Описание**
round(x)	Возвращает результат округления числа x до ближайшего меньшего целого значения для чисел с дробной частью меньше 0.5 или результат округления числа x до ближайшего большего целого значения для чисел с дробной частью больше 0.5
pow(x,y) другой вариант x**y	Возвращает результат возведения числа x в степень y
max(список чисел через запятую)	Возвращает большее значение из списка чисел
min(список чисел через запятую)	Возвращает меньшее значение из списка чисел
sum(список K чисел через запятую)	Возвращает сумму значений элементов последовательности
float(число)	Преобразует объект (например, строковое значение, целое значение) в вещественное число

2. Пример задания на решение уравнения численным методом

Известно, что уравнение $0{,}01e^x = \cos(3x)$ на отрезке [0; 1,5] имеет единственный корень. Найдите его приблизительное значение с точностью не менее 0,00001 и запишите в ответе найденное значение ровно с пятью значащими цифрами после запятой.

Программа на языке Python:

```
from math import cos, exp          # подключить функции cos, exp
def f(x):                          # это функция f(x)
return 0.01*exp(x) - cos(3*x)

a, b = 0, 1.5                      # границы отрезка

while b-a > 1e-6:                   # пока ширина отрезка >= 10^(-6)
  c = (a + b) / 2                  # середина отрезка
  if f(a)*f(c) <= 0:               # сдвигаем правую или левую границу
        b = c
  else: a = c

# вывод с 5 знаками в дробной части
print( "{:.5f}".format((a + b) / 2) )
```

Ответ: 0.51800

3. Пример задания на перебор целых чисел. Разбиение числа на цифры

Назовём натуральное четырёхзначное число N (1000 ≤ N ≤ 9999) счастливым, если суммы двух его первых и двух последних цифр различаются не более, чем на 3. Найдите количество таких чисел.

Программа на языке Python

```
count = 0
```

```
for n in range(1000, 10000):
  d0 = n % 10; n //= 10
  d1 = n % 10; n //= 10
  d2 = n % 10
  d3 = n // 10
  if abs(d3+d2-d1-d0) <= 3:
    count += 1
print(count)
```

Поскольку заданный отрезок [1000; 9999] содержит всего 9000 чисел, можно решать задачу простым перебором. Для этого сначала нужно разбить число на цифры с помощью операций деления нацело и остатка от деления; цифры помещаем в переменные d0, d1, d2, d3. Затем проверяем «счастливость» числа: число счастливое при выполнении условия в этом случае увеличиваем счётчик найденных счастливых чисел.

Ответ: 4071.

4. Пример задания на перебор целых чисел. Проверка делимости

Рассматривается множество целых чисел, принадлежащих отрезку [1033; 7737], которые делятся на 5 и не делятся на 11, 17, 19 и 23. *Найдите количество таких чисел и максимальное из них. В ответе запишите два числа через пробел: сначала количество, затем максимальное число.*

Программа на языке Python

```
count = 0
maxGood = 0
for n in range(1033, 7737+1):
  if (n % 5 == 0) and (n % 11 != 0) and \
     (n % 17 != 0) and (n % 19 != 0) and (n % 23 != 0):
    maxGood = n
    count += 1
print(count, maxGood)
```

Поскольку заданный отрезок [1033; 7737] содержит не так много чисел, можно решать задачу простым перебором. Условие будем понимать так: интересующие нас числа делятся на 5 и не делятся ни на одно из чисел 11, 17, 19 и 23. Нам выгоднее перебирать числа в порядке возрастания, тогда

последнее найдённое число – это и есть искомое максимальное подходящее число (если требуется найти наименьшее подходящее число, удобнее перебирать числа в порядке убывания)

Ответ: 1040 7730

Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [194455; 194500], *числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.*

Программа на языке Python

```
for n in range(194455, 194500+1):
  divs = []
  for d in range(1,n+1):
    if n % d == 0:
      divs.append(d)
  if len(divs) == 4:
    print( *divs )
```

При написании программы на языке Python можно поступить так

```
for для всех чисел n в интервале:
  divs = массив всех делителей n
  if len(divs) == 4:
    вывести массив делителей
```

5. Пример задания на работу с простыми числами

Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [3532000; 3532160], *простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.*

Программа на языке Python

```
from math import sqrt
count = 0
for n in range(3532000, 3532160+1):
```

```
    prime = True
    for d in range(2, round(sqrt(n))):
      if n % d == 0:
        prime = False
        break
    if prime:
      count += 1
      print( count, n )
```

6. Пример задания на работу с символьными строками

В текстовом файле k7.txt находится цепочка из символов латинского алфавита A, B, C, D, E. Найдите количество цепочек длины 3, удовлетворяющих следующим условиям:

- 1-й символ – один из символов B, C или D;
- 2-й символ – один из символов B, D, E, который не совпадает с первым;
- 3-й символ – один из символов B, C, E, который не совпадает со вторым.

Программа на языке Python

```
s = open('k7.txt').read()
count = 0
for i in range(len(s)-2):
    if s[i] in 'BCD' and s[i+1] in 'BDE' \
      and s[i+2] in 'BCE' and s[i]!=s[i+1] \
      and s[i+1]!=s[i+2]:
      count += 1
print(count)
```

Решение:

1) Считываем из файла и перебираем символы.

2) Перебираем все тройки символов. Примем, что переменная i будет хранить номер первого элемента в тройке, то есть, будем рассматривать тройки (**s[i]**, **s[i+1]**, **s[i+2]**).

3) Организуем цикл который перебирает значения **i** от 1 до **len(s)-2**

```
for i in range(len(s)-2):
  ...
```

4) Проверяем символы в каждой тройке на соответствие условию. Проверка принадлежности символов набору аналогична заданию 1. Дополнительно необходимо указать условия неравенства символов, указанных в условии задачи. Если условия выполняются, то к переменной количества прибавляется единица.

7. Пример задания на вычисление значения функции от двух переменных

С помощью редактора электронных таблиц создайте таблицу вещественных значений выражения $F(x,y)=2x^3/(y+1)$ для следующих вещественных значений x и y:

x = 5,5; 6,0; …; 8,5; y = 10,0; 10,3; …; 13,0.

Вычислите сумму получившихся значений и запишите её целую часть в ответе.

Для выполнения этого заданий также можно написать программу.

Решение:

1) Чтобы написать программу, нужно использовать вложенный цикл: в одном цикле будем перебирать значения *x*, а во втором (вложенном) – значения *y*; учитывая, что цикл с переменной (**for ... in ...**) работает только с целыми последовательностями чисел, придётся использовать циклы с условием:

```
s = 0                  # это неправильная программа
x = 5.5                # это неправильная программа
while x <= 8.5:        # это неправильная программа
  y = 10
  while y <= 13:
    # print(x, y) # отладочная печать, см. обсуждение ниже
    s += 2*x**3/(y+1)
```

```
    y += 0.3
  x += 0.5
print( s )
```

сумма значений функции накапливается в переменной **s**

2) однако эта программа выводит неверный ответ.

3) Дело в том, что вещественные числа, которые нельзя представить в виде суммы целых (в том числе и отрицательных) степеней числа 2, в двоичной системе счисления представляют собой бесконечную дробь и поэтому не могут быть точно записаны в памяти двоичного компьютера; при выполнении вычислений с такими числами ошибка накапливается, и к последнему шагу (это можно проверить с помощью отладочной печати) значение *y* равно не 12,7, а чуть больше:

12.700000000000006

из-за этого следующее значение, равное 13,000000000000006, уже больше, чем 13, и не удовлетворяет условию работы цикла; таким образом, на каждом шаге цикла по *x* мы теряем одно значение *y*, и соответствующее значение функции не включается в сумму.

4) с переменной *x* подобных проблем нет, так как шаг изменения *x* равен $0{,}5 = 2^{-1}$

5) исправить ситуацию можно так: организовать перебор только целых значений, используя вспомогательные целочисленные переменные $x_{10} = x \cdot 10$ и $y_{10} = y \cdot 10$:

```
s = 0
for x10 in range(55, 86, 5):
  for y10 in range(100, 131, 3):
    s += 2*(x10/10)**3/(y10/10+1)
print(s)
```

8. Пример задания на вычисление значения рекурсивной функции

Определите наименьшее значение n, при котором сумма чисел, которые будут выведены при вызове F(n), будет больше 500000. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующую сумму выведенных чисел.

```
def F( n ):
  print(2*n)
  if n > 1:
    print(n-5)
    F(n-1)
    F(n-2)
```

Решение:

Первое, что может прийти в голову – вызывать приведённую процедуру при разных значениях параметра и увеличивать это значение до тех пор, пока сумма выведенных чисел не превысит заданное значение 500000; это тупиковый подход, поскольку чисел очень много и сложение займет очень много времени при низкой вероятности правильного ответа

Можно попробовать изменить программу так, чтобы сумма выводимых чисел считалась автоматически: добавим в программу глобальную переменную **s** и будем увеличивать её при выводе каждого числа на значение этого числа; при этом для ускорения (значительного!) работы программы сразу закомментируем вывод чисел на экран:

```
def F( n ):
  global s    # если не объявить s глобальной – ошибка!
  # print(2*n)
  s += 2*n
  if n > 1:
```

```
    # print(n-5)
    s += n - 5
    F(n-1)
    F(n-2)
```

Дальше можно написать такую программу и запускать её при различных значениях переменной **n**:

```
n = 15
s = 0
F(n)
print( n, s )
```

Увеличивая каждый раз значение *n* на 1, мы в конце концов найдём первое (минимальное) значение *n*, при котором сумма чисел, которые будут выведены при вызове *F*(*n*), будет больше 500000 – это *F*(24) = 531864

Ответ: 24 531864.

9. Пример задания на оптимизацию

На покупку мебели выделено 500 тыс. рублей. Стоимость одного комплекта составляет 18 тыс. рублей. Запишите наборы вариантов покупки максимального количества комплектов мебели, при условии, что производитель M продает мебель упаковками по 6 комплектов в упаковке, а производитель N – по 4 комплекта в упаковке.

Запишите в ответ пары чисел: количество упаковок производителя M далее через пробел количество упаковок производителя N. Каждую пару записывайте с новой строки. Пары должны быть отсортированы по возрастанию значений в первом столбце.

Решение:

В простейшем варианте можно просто вывести на экран все варианты сочетаний *a* и *b* с соответствующими значениями *K*, в конце программы вывести максимальное значение *K*; затем вручную найти все строки, где значение *K* равно максимальному.

```
S0 = 500000 # доступная сумма
cost1 = 18000  # стоимость одного комплекта
packM = 6      # количество комплектов в упаковке M
packN = 4      # количество комплектов в упаковке N
```

```
   # максимальное значение a
aMax = int(S0 / (packM*cost1))
   # поиск максимального K по всем вариантам
maxK = 0
for a in range(aMax+1):
  Sb = S0 - a*packM*cost1  # сумма на закупку у N
  b = int(Sb / (packN*cost1))
  K = packM*a + packN*b    # общее количество
  print(a, b, K)
  if K > maxK:
    maxK = K               # новый максимум
print(maxK, maxK*cost1)
```

ВЫВОДЫ

Задача № 8

Для данного типа задач стоит использовать библиотеку itertools для создания всевозможных комбинаций.

Задача № 12

Для данного типа задач стоит использовать библиотеку itertools для создания строк для обработки.

Задача № 15

Использование all() сильно упрощает написание программы и нахождения ответа.

Задача № 16

В данном типе задач можно использовать functools.lru_cache() для сокращения время работы программы.

Задача № 19–21

В некоторых случаях можно использовать библиотеки itertools или math.

Задача № 23

В данном типе задач можно использовать functools.lru_cache() для сокращения время работы программы.

Задача № 25

Для данного типа задач стоит использовать библиотеку fnmatch, чтобы ввести в Python понятие маски.

Задача № 27

В этих задачах надо использовать много списков и уметь с ними по-разному работать.