Конспект к стриму 1

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=40s

Система счисления — это способ записи числа. То есть, как люди могут записывать числа. За всю историю человечество придумывало самые разные способы записи чисел. Это какие-то клинышки, иероглифы, буквы, как, например, на Древней Руси. В Южной Америке число записывалось узелками. Но, все эти системы на сегодняшний день практически не используются. В современном мире используется десятичная система счисления, а также символы римской системы счисления для особого оформления в тексте.

Все системы делятся на два типа. Позиционные и непозиционные.

В качестве примера непозиционной системы исчисления рассмотрим Древнеримскую систему счисления. Она является непозиционной, потому что значение каждой цифры, символа, который обозначает количество, постоянно. Например, символ «I». Он обозначает 1 в любом случае, независимо от его положения в записи числа, символ «V» — это всегда 5, X — это всегда 10. В совокупности их суммарное значение может меняться, но значение каждой цифры, одинаково, несмотря на то, где она стоит в числе, сам символ имеет свой постоянный вес. Поэтому система называется непозиционной, т.к. в ней неважно, на каком месте, на какой позиции стоит цифра.

Можно привести много примеров непозиционных систем. В Древнем Египте, для записи использовали различные иероглифы, и они тоже были непозиционные. Каждый иероглиф обозначал какое-то конкретное число. В Древней Руси система счисления была представлена символами кириллицы, т.е. числа записывались с помощью букв.

И каждая цифра, обозначенная буквой, имела своё определённое значение. Поэтому это была непозиционная система счисления. Значения всех цифр записи числа складывались и то, что получалось, являлось самим числом.

Такие системы неудобны по нескольким причинам. Во-первых, потому что у они имеют определенный потолок в плане максимального значения числа. Во-вторых, с такими числами неудобно производить арифметические операции (складывать, вычитать, умножать, делить).

Сегодня пользуемся позиционными системами. Чем они отличаются? Тем, что в позиционных системах главную роль играет не сама цифра (символ), а то, где она расположен.

Рассмотрим пример. Возьмем числа 10 и 10000. Обратим внимание, что во всех этих числах у нас есть цифра 1, но эта цифра в этих двух числах имеет разное значение. Сама цифра 1 обозначает 1. В числе 10, на второй позиции справа – это десяток. Та же самая единица, но стоящая на пятой позиции, означает 10 тысяч. То есть от того, где расположена цифра, зависит то, что она означает. И эта позиция, место в числе, называется разрядом.

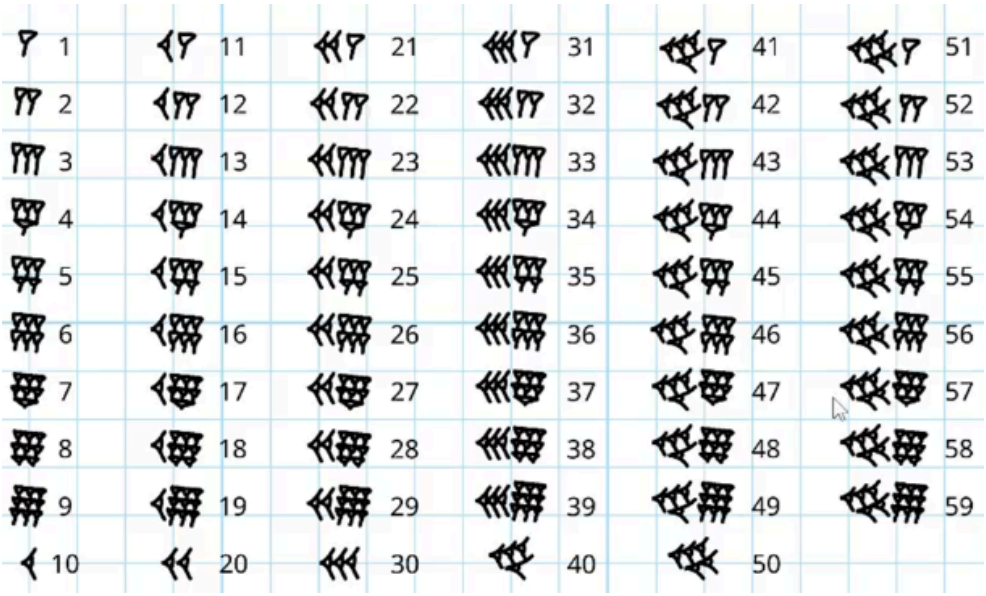
Различаются разряды единиц, десятков, сотен тысяч, десятков тысяч. И каждый разряд, имеет свой вес. Поэтому для записи сколь угодно больших чисел достаточно записать значение каждого разряда, использовав для этого всего 10 символов (цифр).

Когда в разряде накапливается десяток, то счет переходит в следующий разряд. Например, 10 единиц, это десяток. 10 десятков - это сотня, 10 сотен - это тысяча и так далее. Таким образом, используются цифры от 0 до 9, т.к. 10 это уже значение следующего разряда.

Рассмотрим пример разложения на разряды числа 2312. Эта запись означает 2 тысячи + 3 сотни + 1 десяток + 2 единицы. Фактически, число раскладывается по степеням десятки. Т.е., в десятичной системе каждый разряд имеет значение

кратное степени десяти (начиная с нулевой), соответствующее номеру позиции этого числа. Разряд единиц – нулевая степень. Разряд десятков – первая, сотен – вторая, тысяч – третья и т.д..

Но кроме десятичной системы счисления существовали и существуют и другие системы. Например, в древнем Вавилоне использовалась шестидесятиричную систему.



Это цифры от 1 до 60. 60 это максимальное значение каждого разряда, в позиционной системе древнего Вавилона.

Первый разряд – это количество единиц, второй разряд количество шестидесяток, третий разряд шестидесяток в квадрате, то есть по 3600, и так далее.

Таким образом, цифр в системе исчисления может быть и больше 10 и меньше 10. Следовательно и позиционных систем может быть любое количество и они будут отличаться друг от друга разным количеством цифр.

Рассмотрим двоичную систему исчисления. Двоичная система применяется для записи любой информации в компьютере, т.к. компьютер понимает только язык нулей единиц. И значит, числа должны быть записаны тоже нулями и единицами.

Число в двоичной системе счисления раскладывается по степеням двойки и выглядит так 101101_2. Цифра внизу числа обозначает основание системы счисления. Это двоичное число, и в нем используются только цифры 0 1, и каждый разряд означает определенную степень числа 2. Первый (правый) разряд – всегда, как и в любом числе – это разряд единиц, соответствует нулевой степени числа, основания системы, Вторая позиция накапливает значения числа до первой степени основания, т.е. до двух, третья позиция до второй степени, т.е. до четырех, четвертая – до третьей степени двойки, т.е. до 8, пятая до четвертой, т.е. до 16 и т.д. Мы видим, что вес каждого символа на своей позиции существенно меньше, чем вес символа в десятичной системе.

Рассмотрим пятеричную запись числа. В пятеричной записи используется 5 цифр от 0 до 4. Рассмотрим пятеричную запись числа 13243_5. Здесь значения разрядов, соответствует единицам, пятёркам, пятёркам в квадрате, пятёркам в кубе, пятёркам в четвёртой степени. И каждая цифра также означает вес каждого разряда.

Рассмотрим шестнадцатеричную запись. В шестнадцатеричной записи числа используется 16 цифр. И вес каждого разряда означает степень шестнадцати. Например, число 3F0E16. В первом разряде накапливается до 16 единиц, во втором разряде числа до первой степени числа 16, в третьем разряде до 16 в квадрате, в четвертом 16 в кубе и так далее. Т.к. кроме цифр от 0 до 9 нам необходимы еще 6 символов, в качестве символов добавляются используются латинские буквы. Это буквы a, b, c, d, e, f. Они означают то, что в десятичной системе записывается как 10, 11, 12, 13, 14 и 15, т. е. десятичное значение разряда в шестнадцатеричной системе записывается одним символом, соответствующим латинской букве. Очевидно, что в любой системе счисления, которая больше десятичной, будут использоваться дополнительные символы. На примере шестнадцатеричной системы счисления – это латинские буквы.

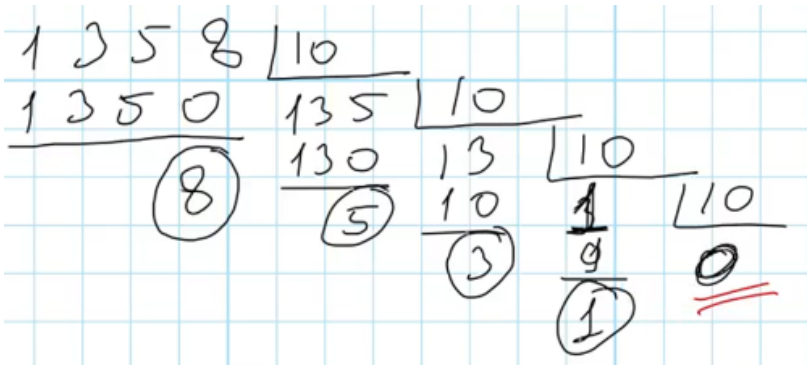
Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=21m55s

Рассмотрим два процесса работы с числами в различных системах счисления.

Первый процесс – это получение цифр числа, то есть как можно получить каждую отдельную цифру числа. И как это может быть использовано для перевода исходного числа в другую систему счисления.

Рассмотрим алгоритм получения цифр числа на примере десятичного числа 1358. Задача состоит в том, чтобы получить все цифры числа по порядку, т. е. разбить число на отдельные цифры.

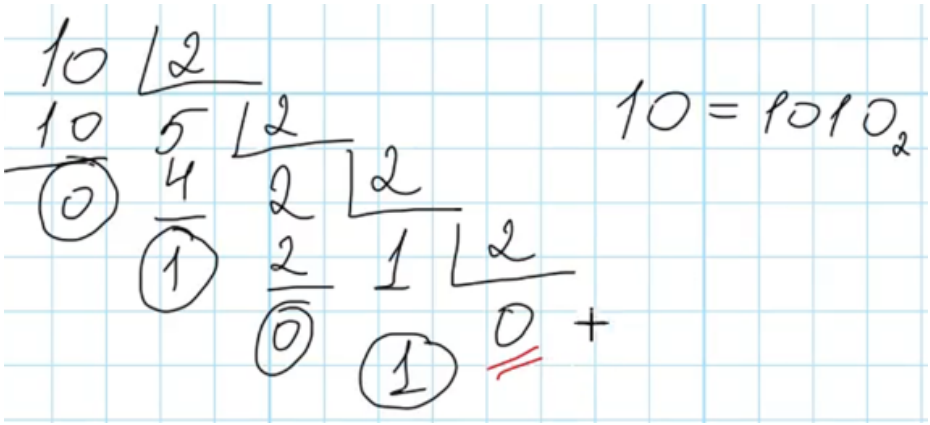
Для этого выполним деление числа на 10 с остатком. Первая операция деления даёт нам в остатке последнюю цифру числа. Повторим это действие для полученного целого результата, следующий остаток даст нам вторую справа цифру числа. Продолжим эти действия, пока результат от деления получаемых числе не станет нулем



На этом процесс деления завершается, т.к. далее он не будет иметь смысла.

Проанализировав полученные результаты, мы видим, что в ходе выполнения четырех итераций действия деления мы получили цифры 8, 5, 3, 1. Это и есть цифры нашего искомого числа в обратном порядке, от младшего разряда к старшему.

По аналогии с этим примером разберём способ получения из десятичного двоичного числа. Для более компактной записи в качестве примере рассмотрим перевод десятичного числа 10 в двоичное число.



Используя тот же принцип, что и в предыдущем примере, мы последовательно будем делить число на 2, т.е. на число соответствующие основанию системы счисления цифры которой мы хотим получить.

Дойдя до 0 при получении целой части деления, остановим процесс. В этом примере мы получили 4 цифры. Т.е. наше число 10, в двоичной записи будет представлено как 1010_2

Аналогично можно получать числа в троичной, и в пятеричной, и в восьмеричной, и в шестнадцатеричной и любых других системах счисления. Рассмотрим перевод число 173 в шестнадцатеричною систему счисления.

Один остаток соответствует одной цифре. Мы получаем два остатка, значит наше шестнадцатеричное число будет состоять из двух цифр, и эти цифры в шестнадцатеричной системе счисления имеют свои соответствующие символы латинскими буквами. Цифра 10 записывается символом «a», а цифра 13 символом «d». Написание символов допустимо, как с заглавной, так и строчной

буквы. Таким образом число 173_{10} в шестнадцатеричной системе счисления будет выглядеть как ad_{16}, или AD_{16}

Последний ноль, который получается в остатке не записывается в начало числа, т.к. не является значимым.

Рассмотрим, как выполнить эти действия с помощью программирования.

Для этого используется две операции. Первая операция «//»– деление нацело.

При выполнении этой операции получается целое частное, и отбрасывается любая дробная часть.

Вторая операция, которая используется для решения подобных задач это – «%», операция для получения остатка от отделения.

Используя эти две операции, правильным образом мы можем перебирать отдельные цифры числа в разных системах счисления.

Например:

1358 // 10 = 135

10 // 2 = 5

1358 % 10 = 8

10 % 2 = 0

Рассмотрим пример получения десятичных цифр числа 1358 в программе

```
#для получения цифр числа нам необходимо несколько раз поделить это число на 10
# перед каждым шагом получая остаток от деления
x=1358
print(x%10)
x=x//10
print(x%10)
x=x//10
print(x%10)
x=x//10
print(x%10)
x=x//10
```

Получаем остатки

```
8
5
3
1
```

Но этот код плохой. Т.к. он не является универсальных. Мы действовали зная, что у нас в числе всего четыре цифры. Если же в числе окажется больше цифр, то мы не получим правильного результата. Т.е. нам нужно предусмотреть такой код, который будет столько раз повторять наши действия, сколько цифр в числе.

При этом цикл «for» мы не можем использовать, т.к. он рассчитан на конкретное количество повторов, здесь количество повторяющихся действий может быть различным. Но у нас есть условие, до какого момента выполнять повторы. Мы выполняем действия до тех пор, пока число, точнее то, что от него

осталось, больше нуля. Как только делимое число стало равно нулю нам нужно прекратить эти действия. Для этого в питоне используется цикл «while», т.е. «пока» - это условный цикл, который выполняет действия, повторяет их до тех пор, пока условие выполняется выглядит

```
#для получения цифр числа нам необходимо несколько раз поделить это число на 10
# перед каждым шагом получая остаток от деления
x=1358456789
while x>0:
  print(x%10)
  x=x//10
```

```
9
8
7
6
5
4
8
5
3
1
```

С помощью этого цикла мы действительно получим все цифры числа в обратном порядке сколько бы их ни было в исходном числе.

Этот алгоритм работает в любых системах счисления.

```
x=10
while x>0:
  print(x%2)
  x=x//2
```

```
0
1
0
1
```

Важно после операции получения остатка выполнять операцию «//», т.к., если не выполнить эту операцию цикл будет повторяться бесконечное количество раз. Принципиально важно, что сначала мы печатаем остаток от деления числа, затем вычисляя целое частное отбрасываем последнюю цифру. Если это сделать наоборот последняя цифра числа будет потеряна и ответ будет неверный.

В Python, используя встроенные методы, есть возможность переводить в двоичную, восьмеричную, шестнадцатеричную систему автоматически.

Для этого используются следующие функции Первая функция bin().

```
bin(10)
```

Она возвращает двоичную запись целого числа

```
0b1010
```

0b - это префикс, который говорит о том, что число двоичное, 10102.

Обычно, когда мы используем эту функцию, этот префикс мешает, поэтому его удаляют. Для этого используется специальный синтаксис срезов.

```
bin(10)[2:]
```

с помощью которого, убирается этот префикс, оставляя только 1 0 1 0. Есть также еще метод. Это метод f-строка.

```
f'{10:b}'
```

после буквы f в фигурных скобках указывается число или переменная, затем двоеточие, а после двоеточия указывается буква b

```
После запуска мы получим
 1010
```

Сразу без префикса. Но по смыслу это такое же действие, как и срез.

Итак, в Python есть три встроенные функции bin, oct, hex для перевода чисел в двоичную, восьмеричную и шестнадцатеричную системы счисления. oct() переводит десятичные числа в восьмеричную систему, hex() – в шестнадцатеричную систему счисления. Чтобы убрать ненужные первые два символа используются срезы, но нужно помнить, что результат этих функций не числа, а строки.

```
>>> bin(10)[2:]
'1010'
>>> oct(32)[2:]
'40'
>>> hex(173)[2:]
'ad'
```

Задача № 1 (2175)

Значение выражения $81^{17} + 3^{24}$ - 45 записали в системе счисления с основанием 9. Сколько цифр 8 содержится в этой записи?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=45m30s

Решение

Нужно получить все цифры числа в системе 9 и сосчитать, сколько из них цифр 8.

Создадим переменную x, в которой сохраним значение заданного выражения. Можно вывести его на экран. Мы получаем число 278128389443693511257568205768197. Эта запись десятичная, а нам нужно девятеричное число, т.к. мы должны получить цифры в девятеричной системе счисления. Для решения этой задачи выполним эти действия с помощью цикла « while». Получившийся результат мы также можем вывести на экран. Это будут цифры нашего девятеричного числа в обратной последовательности.

Для подсчета искомого количества цифр 8 создадим переменную-счетчик.

Дополним цикл «while» проверкой с помощью условного оператора «if» Если полученная на этапе данной итерации мы получили остаток 8, соответственно выделенная из числа цифра равна 8, счётчик k в этот момент увеличивается на 1. Используя оператор «print» выводим на экран полученный ответ.

```
#способ через счётчик
# создаем переменную x
```

```
x = 81**17 + 3**24 - 45
k = 0
while x>0:
  if x%9==8:
    k = k + 1
  x = x//9
print(k)
```

Ответ: 10

Это только первый путь.

Рассмотрим второй вариант решения задачи.

Он заключается в том, что полученные цифры числа мы будем складывать в определенное место. Для этого мы будем использовать структуру Список. Создадим пустой список.

Список — это просто какая-то последовательность элементов, чисел, букв, строк и т.д.

В данном случае список у нас будет в последовательности цифр, которые мы будем получать. В этот список, используя метод append мы будем добавлять цифры девятеричного числа. Это будет выполняться до тех пор, пока преобразуемое число будет больше 0

В результате выполнения программы мы получим список чисел, являющихся цифрами нашего девятеричного числа, записанных, опять же, в обратном порядке.

```
[0, 4, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
```

Работа со списком удобна, так как у списков есть встроенные методы, которые могут помочь ответить на вопрос задачи быстрее. При решении задач №8 мы изучили метод «count» у строк. У списков тоже есть такой метод. И мы можем сосчитать количество чисел 8 в нем.

Обратите внимание, что 8 в данном случае будет будут записываться без кавычек, потому что это в список записаны именно числа записаны. Использовав метод «count» , мы получаем тот же ответ.

```
#способ через список
x = 81**17 + 3**24 - 45
#создаём пустой список
a = []
while x>0:
#добавляем в список значения удовлетворяющие условию
    a.append(x%9)
    x = x//9
print(a.count(8))
```

Ответ: 10.

Задача №2 (3560)

(Е.А. Мирончик) Сколько цифр в восьмеричной записи числа $2^{299}+2^{298}+2^{297}+2^{296}$?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=58m25s

Решение

Первый вариант решения – эти использовать встроенную функцию oct(). Количество элементов списка мы сосчитаем с помощью функции len, которая возващает длину строки или списка.

```
#способ через oct
x = 2**299 + 2**298 + 2**297 + 2**296
#в переменную a запишем восьмеричную строковую запись данного числа
#и с помощью среза удалим первые два лишних символа, указвающих на то, #что число является восьмеричным
a = oct(x)[2:]
print(len(a))
```

Ответ: 100

Второй способ.

```
#способ через счётчик
x = 2**299 + 2**298 + 2**297 + 2**296
k = 0
while x>0:
    k = k + 1
    x = x//8
print(k)
```

Ответ: 100

Третий способ.

```
#способ через список
x = 2**299 + 2**298 + 2**297 + 2**296
a = []
while x>0:
    a.append(x%8)
    x = x//8
print(len(a))
```

Ответ: 100

Задача № 3 (5352)

(ЕГЭ-2022) Значение выражения $13 \cdot 625^{1320} + 12 \cdot 125^{1230} - 14 \cdot 25^{1140} - 13 \cdot 5^{1050} - 2500$ записали в системе счисления с основанием 25. Определите количество значащих нулей в этой записи.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=1h6m15s

Решение

При использовании нашего алгоритм незначащие нули никогда не получаются, поэтому это условие учитывать особым образом не нужно, оно выполняется автоматически само по себе.

Первый способ

```
#способ через счётчик
x = 13*625**1320 + 12*125**1230 - 14*25**1140 - 13*5**1050 - 2500
k = 0
while x>0:
    if x%25==0:
        k = k + 1
    x = x//25
print(k)
```

Ответ:796

Второй способ

```
#способ через список
x = 13*625**1320 + 12*125**1230 - 14*25**1140 - 13*5**1050 - 2500
a = []
while x>0:
    a.append(x%25)
    x = x//25
print(a.count(0))
```

Ответ:796

Задача № 4 (4072)

(В. Шелудько) Значение выражения $4^{1103} + 3 \cdot 4^{1444} - 2 \cdot 4^{144} + 66$ записали в системе счисления с основанием 4. Найдите сумму цифр получившегося числа и запишите её в ответе в десятичной системе счисления

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=1h12m10s

Решение

В этой задаче нам нужно найти не количество, а сумму цифр. Для этого создадим переменную s, которая будет накапливать в себе сумму всех остатков.

Далее мы будем в цикле каждый остаток отделения на 4 добавлять к s, то есть увеличивать s на значение остатка

```
#суммирование в переменную
x = 4**1103 + 3*4**1444 - 2*4**144 + 66
s = 0
while x>0:
    s = s + x%4
    x = x//4
print(s)
```

Ответ: 2882

Второй способ решения:

Сумму списка мы получим используя функцию sum, которая возвращает сумму элементов переданной последовательности.

```
#суммирование списка
x = 4**1103 + 3*4**1444 - 2*4**144 + 66
a = []
while x>0:
    a.append(x%4)
    x = x // 4
print(sum(a))
```

Ответ: 2882

Задача №5 (4619)

Значение выражения $11 \cdot 15^{65} + 18 \cdot 15^{38} - 14 \cdot 15^{17} + 19 \cdot 15^{11} + 18338$ записали в системе счисления с основанием 15. Сколько различных цифр содержится в этой записи?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=1h17m35s

Решение

По условию задачи нужно определить количество уникальных, без повторений, цифр числа.

Если использовать цикл «while», то нужно будет считать, сколько раз встречается каждая цифра числа. Фактически, сколько нулей, сколько единиц, двоек, троек, четверок, пятерок, шестерок, семерок, восьмерок, девяток, десяток, одиннадцать, двенадцать, тринадцать, четырнадцать. Сколько раз встречается, внимание, каждая цифра. И это очень длинный неудобный, хотя и рабочий способ.

При решении подобных задач правильно будет использовать списки.

В списке полученном для решения этой задачи будут находиться все цифры числа, хотя их запись может быть в виде двухзначных цифр до 14 (например,

11,14)

При выводе этого списка на экран мы видим, что в нем есть повторы. Убрать эти повторы можно с помощью структуры «множество» set, которую мы уже использовали при решении 8 задания. set убирает все повторы. Преобразовав список в множество получим структуру, в которой все значения уникальные. Соответственно, len(set(a)) вернет количество уникальных элементов.

```
x = 11*15**65 + 18*15**38 - 14*15**17 + 19*15**11 + 18338
a = []
while x>0:
    a.append(x%15)
    x = x//15
print(set(a), len( set(a) ))
```

Ответ: 10

Задание №6 (3672)

(П.М. Волгин) Значение арифметического выражения 7^2+49^4-21 записали в системе счисления с основанием 14. В этой записи помимо цифр от 0 до 9 могут встречаться цифры из списка: A, B, C, D, которые имеют числовые значения от 10 до 13 соответственно. Сколько цифр A и цифр 0 встречается в этой записи?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=1h25m50s

Решение

По условию задачи нам нужно проверить, имеет ли цифра значение A или значение 0. Для этого будем использовать для каждой цифры свой счетчик, а в цикле два условия. Обратим внимание, что в четырнадцатеричной системе число A будет равняться 10. Последовательно проверим оба этих условия. После того, как мы перебрали в цикле все цифры выводим сумму значений двух этих счетчиков.

Способ 1

```
#способ через счётчик
x = 7**2 + 49**4 - 21
#используем два счетчика
#счетчик цифр оканчивающихся на a
ka = 0
#счетчик цифр оканчивающихся на 0
k0 = 0
while x>0:
#используем два условия
#остаток A - это число 10

    if x%14 == 10:
        ka = ka + 1
```

```
    if x%14 == 0:
        k0 = k0 + 1
    x = x // 14
print(ka+k0)
```

Ответ: 3

Способ 2

```
#способ через список
x = 7**2 + 49**4 - 21
a = []
while x>0:
    a.append(x%14)
    x = x//14
print(a.count(0)+a.count(10))
```

Ответ:3

Задание № 7 (7474)

(ЕГЭ-2024) Значение арифметического выражения $3^{100}-x$, где x – целое положительное число, не превышающее 2030, записали в троичной системе счисления. Определите наибольшее значение x, при котором в троичной записи числа, являющегося значением данного арифметического выражения, содержится ровно пять нулей. В ответе запишите число в десятичной системе счисления

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=1h31m30s

Решение

Для решения этой задачи нужно перебрать все значения выражения для числа x в заданном диапазоне, записать эти значения в троичной системе счисления и посмотреть сколько в этой записи нулей, если их ровно 5, записать это число для того, что бы в дальнейшем выбрать наибольшее из таких чисел и дать ответ.

Т.к. нам нужно рассмотреть большое количество чисел x много мы будем использовать специальный метод «range» , который генерирует последовательности чисел.

```
range(100)      # [0...100)
range(1, 70)    # [1...70)
range(1, 30, 5) # [1, 6, 11, 16, ..., 30)
```

Вариантов задания последовательностей может быть несколько и они указаны на рисунке. В третьем случае третьим параметром задаётся шаг последовательности. Обратите внимание, что число заданное в качестве правой границы в последовательность никогда не включается. Т.к. нам нужно перебрать разные значения x от 1 до 2030 включительно, мы используем цикл «for»,

который последовательно будет перебирать эти значения. Чтобы число 2030 вошло в последовательность чисел зададим в скобках метода «range» запишем диапазон (1, 2031) Для удобства оформления переименуем переменную x в исходном выражении в переменную «a». И напишем для этой переменной цикл перебора, внутри которого известный нам алгоритм работы с цифрами числа в различных системах счисления и определим числа, в записи которых ровно 5 нулей. Выведем эти числа на экран. Нам нужно наибольшее число, это будет число, которое будет выведено на экран последним

```
for a in range(1,2031):
    x = 3**100 - a
    k = 0
    while x>0:
        if x%3==0:
            k = k + 1
        x = x//3
    if k == 5:
        print(a)
```

Ответ: 2024

Задание №8 (7670)

(К. Багдасарян) Значение арифметического выражения $6^{900}+6^{10}-x$, где x – натуральное число, не превышающее 10000, записали в системе счисления с основанием 6. Определите максимальное значение x, при котором данная запись содержит одинаковое количество цифр «3» и «5».

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=1h44m10s

Решение

В этой задаче нужно перебрать всевозможные значения данного числа от 1 до 10 тысяч. Для каждого из них заданное выражение перевести в шестеричный вид и проверить, что цифр 3 и 5 одинаковое количество, и там, где оно действительно одинаковое, вывести это значение.

Для перебора чисел опять, как и в предыдущей задаче будем использовать цикл «for» с методом «range()» Т.к. нам нужен диапазон от 1 до 10000 включительно, запишем в скобках (1, 10001). Значение выражения, которое у нас будет получаться для заданного x на этом диапазоне назовем переменной «chislo». Количество цифр в каждом последующем числе будет меняться, и для каждого нам нужно сосчитать количество троек и пятерок. Для этого создадим два счетчика k3 и k5 и далее используя цикл «while» будем считать количество цифр для каждого числа. После перебора мы проверим, если количество троек равно количеству пятерок, то тогда для этого значения x условие задачи выполняется и мы выведем это значение на экран. В ходе работы программы

мы получим два числа, именно в этих двух числах совпадает количество троек и пятерок. В ответ нам требуется записать максимальное число.

```
for x in range(1,10001):
    chislo = 6**900 + 6**10 - x
    k3 = 0
    k5 = 0
    while chislo>0:
        if chislo%6==3:
            k3 = k3 + 1
        if chislo%6==5:
            k5 = k5 + 1
        chislo = chislo//6
    if k3 == k5:
        print(x)
```

```
>>>
3111
9591
```

Ответ: 9591

Задание №9(7669)

(К. Багдасарян) Значение арифметического выражения $9^{250}+9^{150}-x$, где x – натуральное число, не превышающее 2000, записали в системе счисления с основанием 9. Определите максимальное значение x, при котором данная запись содержит наибольшее количество цифр «1».

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=1h50m50s

Решение

Первым шагом нужно найти наибольшее количество цифр 1. В предыдущих задачах мы научились определить длину и сумму списка. Сейчас мы научимся находить максимум и минимум списка. Первым шагом мы переберем разные значения x от 1 до 2000, включительно. Для каждого из них сосчитаем значение числа. После перебора всех цифр получаем какое-то количество единиц, которое будет в данном числе. Мы получим 2000 разных значений k, среди которых нужно найти наибольшее. Самый простой путь найти максимум или минимум чего-то – это создать список. Список нужно создать до начала работы цикла, а в цикле мы будем добавлять туда полученные значения k – количество единиц для разных x. Выведем на экран результат работы программы. Список состоящий из 2000 тысяч количеств единиц и использовав метод max() его максимальное значение, которое будет равно 4.

```
a = []
for x in range(1,2001):
    c = 9**250 + 9**150 - x
    k = 0
    while c>0:
        if c%9==1:
```

```
            k = k + 1
        c = c//9
    a.append(k) #добавляем в список количество единиц для разных x
print(a)
print(max(a))
```

Используя найденное значение максимального количества единиц в искомом числе найдем ответ задачи. Мы получим два числа 638 и 1367. Т.к. нам нужно максимальное число выбираем в окончательный ответ 1367

```
#a = []
for x in range(1,2001):
    c = 9**250 + 9**150 - x
    k = 0
    while c>0:
        if c%9==1:
            k = k + 1
        c = c//9
    #a.append(k)
    if k == 4: #максимальное число единиц определили предварительно
        print(x)
#print(max(a))
```

Ответ: 1367

Задание №10 (325)

Укажите наименьшее основание системы счисления, в которой запись числа 71 оканчивается на 13.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=2h3m20s

Решение

В предыдущих задачах мы перебирали все цифры в цикле, или собирали список из цифр. Но иногда бывает необходимо получить непосредственно запись самого числа. Для этого можно написать функцию перевода числа в нужную нам, произвольную систему счисления. В данном случае функция – это часть кода, который выделен от основной программы. Он имеет свое имя, он имеет свои входные параметры, то есть ему можно передавать определенные значения, которые он будет принимать и что-то с ними делать

В нашем случае, например, переводить в нужную нам систему счисления и возвращать результат.

Функция создается с помощью ключевого слова def. def означает, мы определяем функцию, даем ей какое-то имя. Например, давайте назовем ее cc,

то есть система исчислений. В скобках мы указываем, что мы ей будем передать, когда будем её вызывать. Нашей функции будут передаваться два параметра. Первый параметр — это число x и второй параметр n это система исчисления, в которую требуется перевести число. Далее, опишем, что происходит в функции. В данном случае, для того, чтобы получить число в указанной системе счисления. Это будут уже знакомые нам шаги, до момента, пока мы не получим все цифры числа. На этом этапе нам нужно вернуть значение функции и это мы делаем с помощью служебного слова «return». Но мы помним, что в результате работы знакомого нам уже алгоритма мы получаем цифры в обратном порядке, и для вывода числа нужно их развернуть, чтобы полученный список цифр шел в обратном порядке. Для этого мы используем опять инструмент срезов и в данном случае он будет выглядеть так a[::-1]

По условию задачи нам нужна система счисления, в которой запись числа 71 будет оканчиваться на 13. Используя цикл «for» для систем счисления с основанием от 2 до 9, передавая в параметры функции само число и значение n из нашего цикла, запустив нашу программы мы получим числа 71 с записью в различных системах счисления. Для ответа на вопрос нашей задачи подходит список оканчивающийся цифрами 1,3 и он соответствует четверичной системе счисления

```
# def - ключевое слово перед именем функции
# cc - имя, по которому будем обращаться к функции
# x, n - значение, которые функция будет обрабатывать
def cc(x,n):
    a = []
    while x>0:
        a.append(x%n)
        x = x//n
    # что функция возвращает после работы
    # в нашем случае список остатков в обратном порядке
    return a[::-1]

for n in range(2,10):
    print(n, cc(71,n))
```

Ответ : 4

Задача № 11(327)

Запись числа 381 в системе счисления с основанием N оканчивается на 3 и содержит 3 цифры. Укажите наибольшее возможное основание этой системы счисления N

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241182?t=2h11m30s

Решение

Решение и функция для решения этой задачи будут аналогичны функции и решению предыдущей задачи. Запустив программу с преданными функции перевода числа параметрами из условия задачи на диапазоне до от 2 до 30 мы получим списки цифр из которых выберем список соответствующий максимальному значению n, содержащий три цифры и оканчивающийся на цифру 3. Ответ будет соответствующее число n.

Примечание Джобса: немного изменил программу с веба, чтобы выводились только те значения, которые нужны по условию.

```
def cc(x,n):
    a = []
    while x>0:
        a.append(x%n)
        x = x//n
    return a[::-1] #переворот списка

for n in range(2,30):
    digits = cc(381, n)
    # условие на количество цифр и значение последней
    if len(digits) == 3 and digits[-1] == 3:
      print(n, digits)
```

```
>>>
9 [4, 6, 3]
14 [1, 13, 3]
18 [1, 3, 3]
```

Ответ: 18