# Урок 06.02. Дополнительные возможности функции print

С самых первый уроков мы используем функцию print, мы её используем в каждой программе. Давайте немного расширим нашими знаниями.

И традиционно, познакомимся с новыми функциями на практике. Вспомним нашу программу, где мы выводили символы фразы «Python!» в каждой отдельной строчке. А если мы захотим выводить символы через пробел, а не на каждой строчке. Давайте попробуем решить эту задачку через новую переменную, в которую мы будем добавлять очередной символ из фразы и пробел, и после формирования этой переменной будем её выводить. Такая программа будет выглядеть вот так:

phrase = ‘Python!’

new\_phrase = ‘’

for symbol in phrase:

new\_phrase += symbol + ‘ ’

print(new\_phrase)

Как-то громоздко и из-за второй переменной памяти много «жрётся». Получается не очень красивый код, хотя красота вещь субъективная. Чтобы такого избежать у функции print есть некоторые настройки. Одна из таких – end. Этот параметр определяет для print чем закончить строку после вывода, по умолчанию это знак перевода на новую строку. И для нашей задачи

phrase = ‘Python!’

for symbol in phrase:

print(symbol, end = ‘ ’)

Вот так просто, лаконично и красиво. На самом деле заканчивать можно любым количеством символов. Тут стоит знать еще несколько специальных символов, которые называются «литералами» (это специальные символы). Основными литералами можно считать ‘\n’ – литерал перевода на новую строчку; ‘\t’ – литерал табуляции.

Давайте решим ещё одну задачу. У Никиты дома есть много компьютеров, которые он хочет подключить к одной локальной сети. Для этого ему нужно сгенерировать ip-адрес (айпи-адрес) для каждого компьютера. IP-адрес записывается как 4 числа, которые отделяются точкой. Не долго думая, для генерации Никита решил использовать арифметическую прогрессию, причём первые 3 числа в адресе - это члены прогрессии, а последнее число - это её сумма.

Напишите программу, где пользователь вводит первый член прогрессии и разность и в результате получает ip-адрес.

Арифметическая прогрессия это такой ряд чисел, которые идут одно число за другим и каждый последующий член прогрессии больше предыдущего на заданное число. Например, «1, 2, 3, 4…», или «10, 20, 30…», или «3, 6, 9, 12, 15…» и др.

first\_number = int(input(‘Введите первое число: ’))

step = int(input(‘Введите первое число: ’))

number = 0

sum = 0

print(‘\nIP-адрес: ’, end=‘ ’)

for count in range(3):

print(number, end = ‘.’)

summ += number

number += step

print(summ)

И всё. Запустим программу. Наша программа работает.

## Типовые алгоритмы работы со строками.

Давайте рассмотрим ещё одну типовую задачу на строки и числа.

Всем нам знакомо слово фильтр. Это такое устройство, которое что-то пропускает (чистую воду в кран), а что-то задерживает (всякие примеси). Именно это нам и надо будет сделать. Необходимо написать программу, которая будет из введенной строчки удалять все цифры и сумму этих цифр. Представим, что данная программа будет запускаться на очень слабом компьютере, и одним из ограничений на программу будет – использование всего двух переменных,

text = input(‘Введите текст: ’)

summ = 0

print(‘\nОтфильтрованный текст:’, end=‘ ’)

for symbol in text:

if ‘1’ < symbol < ‘9’

summ += int(symbol)

else:

print(symbol, end=‘’)

print(‘\nСумма: ’, summ)

Запустим. Всё работает. Отлично.

Следующая задача. Нужно определить имеется ли в строке две одинаковые буквы подряд. Простая задача... Возможно. Давайте не будем спешить и подумаем, как решить эту задачу.

Первое, это сделать ввод строки. Дальше нам нужно перебирать все символы строки. Теперь нам нужно этот (очередной) символ symbol сравнивать с предыдущим символом строки. И если они совпали, то установить флаг, что мы нашли такую пару символов.

text = input(‘Введите строку: ’)

prevSym = ‘’

equalSym = False

for symbol in text:

if prevSym == symbol:

equalSym = True

break

else:

prevSym = symbol

if equalSym:

print(‘Есть две одинаковые подряд идущие буквы’)

else:

print(‘Нет двух одинаковых подряд идущих букв’)