Информатика ЕГЭ

100БАЛЛЬНЫЙ РЕПЕТИТОР

ШПАРГАЛКА ПО PYTHON

ЧАСТЬ 1

Артем Имаев

ПОДПИШИСЬ НА СОЦ. СЕТИ

Telegram
Связь со мной, закрытые занятия, презентации

ВКонтакте
Гайды и полезные подборки

YouTube
Теория и дополнительные материалы

ВЫВОД НА ЭКРАН

print(12) – функция (оператор) выводит на экран данные. Внутри круглых скобок записываются выражения, которые нужно вывести на экран.

print('a') – выведет **a**. При вводе слов/букв будем заключать текст в кавычки. Если этого не сделать, компьютер посчитает, что нам необходимо вывести переменную и выдаст ошибку.

print(5, 'privet') – выведет **5 privet**. При вводе каждый аргументы будем отделять между собой запятой. При выводе получаем аргументы, отделённые пробелом.

print(5 + 3) – выведет **8**

ВВОД ДАННЫХ С КЛАВИАТУРЫ

Для получения информации с клавиатуры в Python есть функция **input()**. Когда **input()** вызывается, поток программы останавливается до тех пор, пока пользователь не введет данные через командную строку. Для ввода нужно нажать Enter после завершения набора текста. **ВАЖНО: По умолчанию функция input() конвертирует всю получаемую информацию в строку.**

```
a = input()
print(a + 5)
```

Если мы введём в командную строку 7, компьютер выведет ошибку, так как в этом случае a = '7'. Строку и число нельзя складывать

```
a = int(input())
print(a + 5)
```

Если мы введём в командную строку 7, компьютер выведет 12, так как a преобразовано в числовой тип данных

ПЕРЕМЕННЫЕ

Переменные предназначены для хранения данных.

- первый символ должен быть заглавной или строчной латинской буквой или нижним подчёркиванием _;
- остальные символы могут быть заглавными или строчными латинскими буквами, нижними подчёркиваниями и цифрами;
- нельзя использовать пробелы;
- имя переменной не должно совпадать ни с каким из зарезервированных в Python ключевых слов.

Допустимые имена: x, X, xyz, _x_y_z, XYZ, xyz_123, _123, x1Y2z2.

Как **нельзя** называть переменные: 1, 1x, x y z, x&y.

Регистр в Python имеет значение: name и Name будут считаться разными переменными

ТИПЫ ДАННЫХ

Целые числа

Тип **int** представляет целое число, например, 1, 4, 8, 50.

Дробные числа

Тип **float** представляет число с плавающей точкой, например, 1.2 или 34.76. В качестве разделителя целой и дробной частей используется точка.

Логические значения

Тип **bool** представляет два логических значения: True (верно, истина) или False (неверно, ложь).

```
count = 15
print(count)          # 15

height = 1.68
weight = 68.
print(height)         # 1.68
print(weight)         # 68.0

A = False
print(A)              # False
B = True
print(B)              # True
```

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ И ВЫРАЖЕНИЯ

ОПЕРАЦИЯ	ВЫВОД	РЕЗУЛЬТАТ
+ — сложение	print(6 - 2)	# 4
- — вычитание	print(6 + 2)	# 8
* — умножение	print(6 * 2)	# 12
** — возведение в степень	print(6 ** 2)	# 36 (Возводим число 6 в степень 2)
/ — деление	print(6 / 2)	# 3.0
	print(7 / 2)	# 3.5
// — целочисленное деление	print(7 // 2)	# 3 (целое число от деления 7 на 2)
% - остаток от деления	print(7 % 2)	# 1 (остаток от целочисленного деления числа 7 на 2)

ОПЕРАЦИЯ	ВЫВОД	РЕЗУЛЬТАТ
+= Присвоение результата сложения	number = 10	
-= Присвоение результата вычитания	number += 5	
*= Присвоение результата умножения	print(number)	# 15
/= Присвоение результата от деления	number -= 3	
//= Присвоение результата целочисленного деления	print(number)	# 12
**= Присвоение степени числа	number *= 4	
%= Присвоение остатка от деления	print(number)	# 48

ПЕРЕВОД В СИСТЕМЫ СЧИСЛЕНИЯ

int(‘10000000’, 2)

В языке программирования Python встроенная функция int() возвращает **целое число** (экземпляр класса int) **в десятичной системе счисления.** В случае, когда указывается второй аргумент для функции int(), первый всегда должен быть строкой. С помощью второго аргумента сообщается, в какой системе счисления находится число, указанное в строке первого аргумента.

bin(12) – из 10сс переводит в 2сс

oct(12) – из 10сс переводит в 8сс

hex(12) – из 10сс переводит в 16сс

Первые 2 символа – технические. На них не обращаем внимание. Если хотим избавиться от них, можем написать [2:] после функции (например bin(12)[2:])

ОПЕРАТОР УСЛОВИЯ IF

Сначала записывается условие (“>”,”<”,”>=”,”<=”,”==”)

Затем на следующей строке с отступом записывается действие, которое выполняется при выполнении условия.

Если хотим чтобы программа после **if** выполнила что – то еще, то записываем это на следующей строке без отступа:

```
a = 5
b = 10
if a==b:
   print(‘*’)
print(‘----------‘)
```

Здесь, a и b равны не равны, поэтому программа выведет ‘----------’, а если a и b были бы равны, то программа выведет ‘*’ и ‘----------’

Также существует **“else”**, в него программа входит, если она не вошла в if, например:

```
a = 5
b = 10
if a > b:
    print(‘*’)
else:
    print(‘+’)
print(‘----------‘)
```

Здесь программа выводит ‘+’ и “----------’, так как a < b

Существует условие **‘elif’**, которое проверяется в том случае, если не выполняется условие ‘if’:

```
a = 10
b = 10
if a > b:
    print(‘*’)
elif a == b:
    print(‘$’)
else:
    print(‘+’)
print(‘----------‘)
```

Здесь программа выводит ‘$’ и “----------’, так как a = b. ‘elif’ в программе может встречаться по несколько раз.

ОПЕРАТОРЫ ЦИКЛОВ

ЦИКЛ WHILE

Цикл с условием **while** выполняется, пока истинно задающее его условие. Поэтому этот цикл также иногда называют циклом "пока".

Часто цикл **while** используется, когда невозможно заранее предсказать, сколько раз необходимо выполнить тело цикла: **while (условие): ....**

```
a = 0
while a < 5:
a += 1
print(a)
```

Такая программа выведет числа от 1 до 5

```
a = 0
while a < 5:
print(a)
a += 1
```

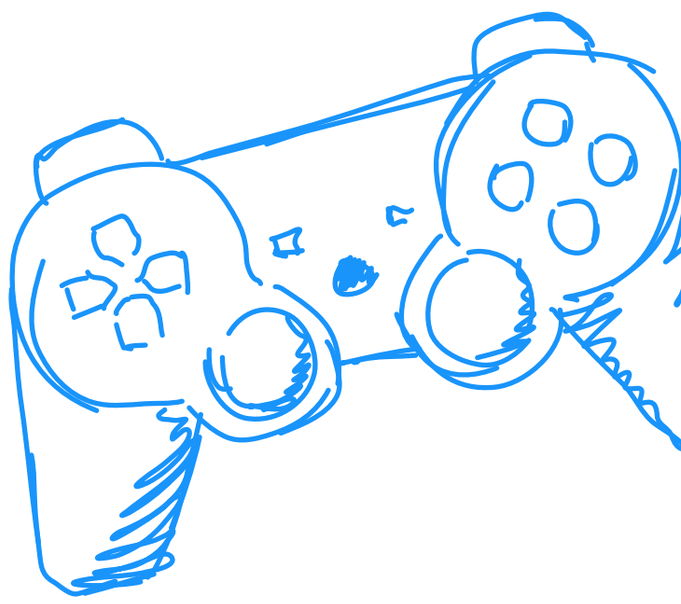
Но если поменять 2 последние строчки, то выведет числа от 1 до 4

Вводится число N. Выведите количество четных цифр этого числа

```
x = int(input())
k = 0
while x > 0:
    if (x % 10) % 2 == 0:
        k += 1
    x = x // 10
print(k)
```

Дано целое число, не меньшее 2. Выведите количество его натуральных делителей

```
x = int(input())
i = 2
print(1, x, end = ' ')
while i <= x //2 :
    if x % i == 0:
        print(i, end = ' ')
    i += 1
```



Вводится число N. Выведите сумму цифр этого числа

```
x = int(input())
s = 0
while x > 0:
    s = s + (x % 10)
    x = x//10
print(k)
```

ЦИКЛ FOR

Цикл **for** используется в двух случаях:

- Если нужно выполнить одну операцию или набор из нескольких различных действий определенное **количество раз.**
- Если необходимо провести перебор элементов коллекции – списка, строки, словаря, кортежа, множества – одновременно выполняя какиелибо операции с этими элементами.

for i in range(a,b,k): где i = переменная (на её месте может быть любое другое значение),

a – начальное значение (включительно), b - конечное значение (не включительно), d – шаг

КАК ИЗМЕНЯЕТСЯ ПЕРЕМЕННАЯ В ЦИКЛЕ

for i in range(1, 10)	1, 2,3,4,5,6,7,8 9
for i in range(1, 10, 2)	1, 3, 5, 7, 9
for i in range(10, 7, -1)	10, 9, 8
for i in range(5, 2, -1)	5, 4, 3
for i in range(5, 0, -1)	5, 4, 3, 2, 1

s=0 – **вводим сумму. В начальный момент она равна 0**
for i in range (10): – **перебираем 10 предметов**
    x=int(input()) – **считываем стоимость предмета**
    s += x – **добавляем его стоимость к сумме**
print(s) – **выводим ответ**

Дано N чисел: сначала вводится число N, затем вводится ровно N целых чисел. Подсчитайте количество нулей среди введенных чисел и выведите это количество

n=int(input()) – **вводим n**
k=0 – **вводим счётчик**
for i in range (n): – **перебираем каждое число**
    x=int(input()) – **считываем каждое число**
    if x == 0:
        k+=1 – **если число равно 0, добавляем 1 к счётчику**
print(k)) – **выводим ответ**

Дано число x, перевести в троичную систему счисления

```
x = int(input())
while x > 0:
    print(x % 3, end = '')
    x = x // 3
```