

В JavaScript числа представлены либо как целые, либо как числа с плавающей точкой, а также поддерживают ряд встроенных методов и функций для работы с ними. Рассмотрим подробно.

---

## Типы чисел в JavaScript

1. **Целые числа (integer)** — числа без дробной части.
  2. `let int = 42;`
  3. **Числа с плавающей точкой (float)** — числа с дробной частью.
  4. `let float = 3.14;`
  5. **Специальные значения:**
    - o **Infinity** — результат деления на 0 или переполнения.
    - o `console.log(1 / 0); // Infinity`
    - o **-Infinity** — отрицательное переполнение.
    - o **NaN (Not a Number)** — результат некорректных операций с числами.
    - o `console.log("abc" / 2); // NaN`
- 

## Создание чисел

1. **Числовые литералы:**
  2. `let num = 123;`
  3. **Конструктор Number:**
  4. `let num = new Number(123);`
- 

## Свойства чисел

### 1. Константы Number

- **Number.MAX\_VALUE** — максимально возможное значение числа.
  - `console.log(Number.MAX_VALUE); // 1.7976931348623157e+308`
  - **Number.MIN\_VALUE** — минимально возможное положительное значение.
  - `console.log(Number.MIN_VALUE); // 5e-324`
  - **Number.MAX\_SAFE\_INTEGER** — максимально безопасное целое значение.
  - `console.log(Number.MAX_SAFE_INTEGER); // 9007199254740991`
  - **Number.MIN\_SAFE\_INTEGER** — минимально безопасное целое значение.
  - `console.log(Number.MIN_SAFE_INTEGER); // -9007199254740991`
  - **Number.POSITIVE\_INFINITY / Number.NEGATIVE\_INFINITY** — бесконечности.
  - **Number.NaN** — значение "не число".
- 

## Методы чисел

### 1. Преобразование чисел

- **toString([radix])** — преобразует число в строку в указанной системе счисления.
- `console.log((255).toString(16)); // "ff" (шестнадцатеричная)`

- **toFixed(digits)** — форматирует число с фиксированным количеством десятичных знаков.
  - `console.log((3.14159).toFixed(2)); // "3.14"`
  - **toExponential(fractionDigits)** — возвращает число в экспоненциальной форме.
  - `console.log((123456).toExponential(2)); // "1.23e+5"`
  - **toPrecision(precision)** — форматирует число с заданным общим количеством значащих цифр.
  - `console.log((123.456).toPrecision(4)); // "123.5"`
- 

## 2. Проверка чисел

- **Number.isFinite(value)** — проверяет, является ли число конечным.
  - `console.log(Number.isFinite(42)); // true`
  - `console.log(Number.isFinite(Infinity)); // false`
  - **Number.isInteger(value)** — проверяет, является ли число целым.
  - `console.log(Number.isInteger(42)); // true`
  - `console.log(Number.isInteger(3.14)); // false`
  - **Number.isSafeInteger(value)** — проверяет, является ли число безопасным целым.
  - `console.log(Number.isSafeInteger(42)); // true`
  - `console.log(Number.isSafeInteger(1e16)); // false`
  - **Number.isNaN(value)** — проверяет, является ли значение NaN.
  - `console.log(Number.isNaN(NaN)); // true`
  - `console.log(Number.isNaN("abc")); // false`
- 

## 3. Округление чисел

- **Math.round(value)** — округляет до ближайшего целого.
  - `console.log(Math.round(4.5)); // 5`
  - **Math.ceil(value)** — округляет вверх.
  - `console.log(Math.ceil(4.2)); // 5`
  - **Math.floor(value)** — округляет вниз.
  - `console.log(Math.floor(4.8)); // 4`
  - **Math.trunc(value)** — отбрасывает дробную часть.
  - `console.log(Math.trunc(4.9)); // 4`
- 

## 4. Арифметика

- **Math.abs(value)** — возвращает абсолютное значение.
  - `console.log(Math.abs(-42)); // 42`
  - **Math.pow(base, exponent)** — возводит в степень.
  - `console.log(Math.pow(2, 3)); // 8`
  - **Math.sqrt(value)** — извлекает квадратный корень.
  - `console.log(Math.sqrt(16)); // 4`
  - **Math.cbrt(value)** — извлекает кубический корень.
  - `console.log(Math.cbrt(27)); // 3`
  - **Math.hypot(...values)** — вычисляет квадратный корень суммы квадратов.
  - `console.log(Math.hypot(3, 4)); // 5`
- 

## 5. Тригонометрия

- **Math.sin(value) / Math.cos(value) / Math.tan(value)** — синус, косинус, тангенс.

- `console.log(Math.sin(Math.PI / 2)); // 1`
- `Math.asin(value) / Math.acos(value) / Math.atan(value)` — аркфункции.
- `Math.atan2(y, x)` — вычисляет угол между осью x и точкой (y, x).

---

## 6. Логарифмы и экспоненты

- `Math.exp(value)` — вычисляет  $e^{\text{value}}$ .
- `console.log(Math.exp(1)); // 2.718281828459045`
- `Math.log(value)` — вычисляет натуральный логарифм.
- `console.log(Math.log(Math.E)); // 1`
- `Math.log10(value) / Math.log2(value)` — логарифмы по основаниям 10 и 2.

---

## 7. Генерация случайных чисел

- `Math.random()` — возвращает случайное число от 0 (включительно) до 1 (исключительно).
- `console.log(Math.random()); // Например: 0.654321`

---

## Примеры для практики

Если нужно описание дополнительных функций или реализация задач с числами, обращайтесь!

# братный порядок 2

Дано натуральное число. Напишите программу, которая меняет порядок цифр числа на обратный.

### Формат входных данных

На вход программе подаётся одно натуральное число.

### Формат выходных данных

Программа должна вывести число, записанное в обратном порядке.

Тестовые данные ☐

---

### Sample Input 1:

5086334

---

### Sample Output 1:

4336805

```
let str = "12345"
let str2 = ""
for (let i=4 ; i > -1 ; i--){
    str2 += str[i]
}
```

```
console.log(str2)
```

```
x =int(input())
while x != 0:
    a= x % 10
    print(a,end="")
    x = x // 10
```

Дано натуральное число  $n(n \geq 10)$ . Напишите программу, которая определяет его максимальную и минимальную цифры и выводит текст в следующем формате:

Максимальная цифра равна <максимальная цифра>

Минимальная цифра равна <минимальная цифра>

#### Формат входных данных

На вход программе подаётся одно натуральное число  $n(n \geq 10)$ .

#### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

Тестовые данные ☐

---

#### Sample Input 1:

26670

---

#### Sample Output 1:

Максимальная цифра равна 7

Минимальная цифра равна 0

```
const num = 194730
let numStr = String(num)
let maxOne = numStr[0]
let minOne = numStr[1]
for (let i =0 ;i < numStr.length ; i++){
    numStr[i] > maxOne ? maxOne =numStr[i] : maxOne = maxOne
    numStr[i] < minOne ? minOne =numStr[i] : minOne = minOne
}
console.log(maxOne,minOne)
```

```
x = input()
```

```
d=str(x)
```

```
print("Максимальная цифра равна",max(d))
```

```
print("Минимальная цифра равна",min(d))
```

---

Сумма чисел и произведение

```
const num = 194730
let numStr = String(num)
let sum = 0
let ymn = 1

for (let i = 0; i < numStr.length; i++){
    sum += Number(numStr[i])
    ymn *= Number(numStr[i])
}
console.log(sum, ymn)
```

```
const num = 11111
let numStr = String(num)
let first = numStr[0]
let flag = true
for (i = 0; i < numStr.length; i++){
    if (numStr[i] !== first){
        flag = false
    }
}
flag ? console.log("=") : console.log("!=")
```

Дано натуральное число  $n$  ( $n \leq 9$ ). Напишите программу, которая печатает таблицу сложения для всех чисел от 1 до  $n$  (включительно) в соответствии с примером.

#### Формат входных данных

На вход программе подаётся одно натуральное число.

#### Формат выходных данных

Программа должна вывести таблицу сложения для всех чисел от 1 до  $n$ .

**Примечание 1.** Таблицу сложения подразумеваем от 1 до 99 (включительно).

```
const num = 3
for (let i = 1; i <= 3; i++){
    for (let j = 1; j < 13; j++){
        console.log(`${i} x ${j} = ${i*j}`)
    }
    console.log()
}
```

## Численный треугольник 2

Дано натуральное число  $n$ . Напишите программу, которая печатает численный треугольник с высотой, равной  $n$ , в соответствии с примером:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
...
```

#### Формат входных данных

На вход программе подаётся одно натуральное число.

#### Формат выходных данных

Программа должна вывести треугольник в соответствии с условием.

**Примечание.** Используйте вложенный цикл `for`.

Тестовые данные ☐

---

#### Sample Input 1:

3

---

#### Sample Output 1:

```
1
2 3
4 5 6
```

---

#### Sample Input 2:

6

---

#### Sample Output 2:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
```

```

counter = 1;
for (i = 1; i <= 21; i++){
    let row = ""
    for (j = 1 ;j <= i && counter < 22 ; j++){
        row = row + " " + counter
        counter +=1
    }
    console.log(row)
}

```

```

counter =1
for (i = 1;i < 22; i++){
    row = ""
    for (j =1;j < i && counter < 22;j++){
        row = row +""+ counter
        counter +=1
    }
    console.log(...row)
}

```

```

counter = 1
for i in range(1, 22):
    for j in range(i):
        if counter == 22:
            break
        print(counter, end=" ")
        counter += 1
    print()

```

## Численный треугольник 3

Дано натуральное число  $n$ . Напишите программу, которая печатает численный треугольник с высотой, равной  $n$ , в соответствии с примером:

```

1
121
12321
1234321
123454321
...

```

### Формат входных данных

На вход программе подаётся одно натуральное число

```
const n = 5
for (i=1;i<n;i++){
  row = ""
  for (j=1 ;j < i; j++){
    row = row + j
  }
  for (k=j ;k >= 1;k--){
    row = row + k
  }
  console.log(row)
}
```

```
n=5
for i in range(5+1):
    for j in range(1,i):
        print(j,end=" ")
    for k in range(i,0,-1):
        print(k,end=" ")
    print()
```

## Делители-1 🐦☐

На вход программе подаются два натуральных числа  $aa$  и  $bb$  ( $a < b$ ).

Напишите программу, которая находит натуральное число из

отрезка  $[a;b]$  (от  $aa$  до  $bb$  включительно) с максимальной суммой делителей. Если чисел с максимальной суммой делителей несколько, то искомым числом является наибольшее из них. Ваша программа должна выводить ответ в следующем формате:

<число с максимальной суммой делителей> <сумма делителей этого числа>

### Формат входных данных

На вход программе подаются два числа, каждое на отдельной строке.

### Формат выходных данных

Программа должна вывести два числа на одной строке, разделенных пробелом: число с максимальной суммой делителей и сумму его делителей.

Тестовые данные ☐

---

#### Sample Input 1:

1  
10

---

#### Sample Output 1:

10 18



---

**Sample Input 2:**

1  
100

---

**Sample Output 2:**

96 252

```
lst=[]
sum_max = 0
i_max= 0
for i in range(61,71):
    sum = 0
    for j in range(1,i+1):
        if i % j == 0:
            # print(j)
            sum =sum + j
    lst.append([i,sum])
    if sum >= sum_max:
        sum_max = sum
        i_max =i
print(i_max,sum_max)
```

```
let sumaDelite = 0;
let numWithMaxSum = 0;
let maxSumma = 0;
const a = 10;
const b = 16;

for (let i = a; i <= b; i++) { // Включаем b
    sumaDelite = 0; // Сбрасываем сумму делителей для каждого нового i
    for (let j = 1; j <= i; j++) { // Цикл для нахождения делителей
        if (i % j === 0) {
            sumaDelite += j; // Добавляем делитель к сумме
        }
    }
    // Проверяем, является ли текущая сумма больше максимальной
    if (sumaDelite >= maxSumma) {
        maxSumma = sumaDelite;
        numWithMaxSum = i;
    } else if (sumaDelite === maxSumma && i > numWithMaxSum) {
        numWithMaxSum = i; // Если суммы равны, выбираем большее число
    }
}

console.log(numWithMaxSum, maxSumma);
```

## Делители-2

На вход программе подаётся натуральное число  $n$ . Напишите программу, выводящую графическое изображение делимости чисел от 1 до  $n$  включительно. В каждой строке надо напечатать очередное число и столько символов `+`, сколько делителей у этого числа.

#### Формат входных данных

На вход программе подаётся одно натуральное число.

#### Формат выходных данных

Программа должна вывести графическое изображение чисел от 1 до  $n$ , каждое на отдельной строке.

Тестовые данные ☐

---

#### Sample Input:

5

---

#### Sample Output:

1+  
2++  
3++  
4+++  
5++

```
for i in range(1,5+1):  
    str = ""  
    for j in range(1,i+1):  
        if i % j ==0:  
            str = str+"+"  
    print(i,str,sep="")
```

```
for (i =1; i <= 5;i++){  
    let row = ""  
    for (j =1; j <=i;j++){  
        if (i % j == 0){  
            row = row + "+"  
        }  
    }  
    console.log(`${i}${row}`)  
}
```

## Цифровой корень 🐦 ☐

Цифровым корнем числа  $n$  называется число, получающееся следующим образом: вычисляется сумма цифр числа  $n$ , затем сумма цифр у получившегося числа и так далее, пока не получится однозначное число. Например, цифровой корень числа 98759875 равен 22:

- $9+8+7+5=29$   
 $9+8+7+5=29$

- $2+9=11$   $2+9=11$
- $1+1=2$   $1+1=2$

На вход программе подаётся натуральное число  $n$ . Напишите программу, которая находит цифровой корень данного числа.

#### Формат входных данных

На вход программе подаётся одно натуральное число.

#### Формат выходных данных

Программа должна вывести цифровой корень введённого числа.

**Примечание.** Используйте вложенные циклы `while`.

Тестовые данные ☐

#### Sample Input 1:

192

#### Sample Output 1:

3

#### Sample Input 2:

6

#### Sample Output 2:

6

```
n = str(129)
while len(n) > 1:
    n = sum([int(i) for i in n])
    n = str(n)
print(n)
```

```
let n = 129
n = String(n)
while (n.length > 1){
    array = n.split("")
    let summa = 0
    for (i in array){
        summa = summa + Number(i)
    }
    n = String(summa)
}
console.log(n)
```

## Сумма факториалов !

Дано натуральное число  $n$ . Напишите программу, которая выводит значение суммы:

$$1!+2!+3!+\dots+n!1!+2!+3!+\dots+n!$$

#### Формат входных данных

На вход программе подаётся одно натуральное число.

#### Формат выходных данных

Программа должна вывести значение суммы  $1!+2!+3!+\dots+n!1!+2!+3!+\dots+n!$ .

**Примечание 1.** Факториалом натурального числа  $n$  называется произведение всех натуральных чисел от 1 до  $n$ , то есть:  $n!=1\cdot 2\cdot 3\cdot \dots \cdot n$

**Примечание 2.** Задачу можно решить без вложенного цикла. Напишите две версии программы. 😊

```
let suma = 0

for (i=1; i <= 10; i++){
    let ymn = 1
    for (j = 1 ; j <= i ; j++){
        ymn *= j
    }
    suma = suma + ymn
}

console.log(suma)
```

```
summa = 0
for i in range(1,11):
    ymn = 1
    for j in range(1,i+1):
        ymn = ymn * j
    summa += ymn
print(summa)
```

## Простые числа 🐼

На вход программе подается два натуральных числа  $a$  и  $b$  ( $a < b$ ).

Напишите программу, которая находит все простые числа от  $a$  до  $b$  включительно.

#### Формат входных данных

На вход программе подаются два числа, каждое на отдельной строке.

#### Формат выходных данных

Программа должна вывести все простые числа от  $a$  до  $b$  включительно, каждое на отдельной строке.

**Примечание 1.** Простое число – это натуральное число, единственными делителями которого являются только оно само и 11.

**Примечание 2.** Число 11 простым не является.

Тестовые данные ☐

---

**Sample Input 1:**

2  
15

---

**Sample Output 1:**

2  
3  
5  
7  
11  
13

```
for (i =2; i <= 15 ;i++){  
    flag = true  
    for (j =2; j < i; j++){  
        if (i % j == 0){  
            flag = false  
        }  
    }  
    if (flag){  
        console.log(i)  
    }  
}
```

```
for i in range(2,15):  
    flag = True  
    for j in range(2,i):  
        if i % j == 0:  
            flag =False  
    if flag:  
        print(i)
```

## Decimal to Binary 10

На вход программе подаётся натуральное число, записанное в десятичной системе счисления. Напишите программу, которая переводит данное число в [двоичную систему счисления](#).

```
num = int(input())
print(bin(num).replace("0b", ""))
```

```
num = 45
console.log(num.toString(2))
```

В JavaScript строки представляют собой неизменяемые последовательности символов. Они имеют множество методов и функций для работы с текстом. Рассмотрим их подробно.

---

## Создание строк

Строки можно создавать двумя способами:

1. **Литерал строки:**
  2. `let str = "Hello, World!";`
  3. **Конструктор String:**
  4. `let str = new String("Hello, World!");`
- 

## Основные свойства строки

- **length** — возвращает длину строки.
  - `let str = "Hello";`
  - `console.log(str.length); // 5`
- 

## Методы строк

### 1. Извлечение подстроки

- **charAt(index)** — возвращает символ по индексу.
  - `let str = "Hello";`
  - `console.log(str.charAt(1)); // "e"`
  - **charCodeAt(index)** — возвращает код символа в Юникоде.
  - `console.log(str.charCodeAt(1)); // 101 (код 'e')`
  - **slice(start, end)** — извлекает часть строки (не включая end).
  - `console.log(str.slice(1, 4)); // "ell"`
  - **substring(start, end)** — аналогично slice, но не принимает отрицательные индексы.
  - `console.log(str.substring(1, 4)); // "ell"`
  - **substr(start, length)** — извлекает подстроку заданной длины (устаревший метод, лучше избегать).
  - `console.log(str.substr(1, 3)); // "ell"`
-

## 2. Изменение регистра

- **toUpperCase()** — переводит строку в верхний регистр.
  - `console.log("hello".toUpperCase()); // "HELLO"`
  - **toLowerCase()** — переводит строку в нижний регистр.
  - `console.log("HELLO".toLowerCase()); // "hello"`
- 

## 3. Поиск подстроки

- **indexOf(substring, start)** — возвращает индекс первого вхождения подстроки.
  - `console.log("Hello".indexOf("l")); // 2`
  - **lastIndexOf(substring, start)** — возвращает индекс последнего вхождения подстроки.
  - `console.log("Hello".lastIndexOf("l")); // 3`
  - **includes(substring, start)** — проверяет, содержит ли строка подстроку.
  - `console.log("Hello".includes("ell")); // true`
  - **startsWith(substring, start)** — проверяет, начинается ли строка с подстроки.
  - `console.log("Hello".startsWith("He")); // true`
  - **endsWith(substring, length)** — проверяет, заканчивается ли строка на подстроку.
  - `console.log("Hello".endsWith("lo")); // true`
- 

## 4. Изменение строки

- **replace(searchValue, newValue)** — заменяет первое вхождение подстроки.
  - `console.log("Hello, World!".replace("World", "JS")); // "Hello, JS!"`
  - **replaceAll(searchValue, newValue)** — заменяет все вхождения подстроки.
  - `console.log("Hello, World, World!".replaceAll("World", "JS")); // "Hello, JS, JS!"`
  - **trim()** — удаляет пробелы с начала и конца строки.
  - `console.log(" Hello ".trim()); // "Hello"`
  - **trimStart()** / **trimEnd()** — удаляют пробелы только с начала или конца строки.
  - `console.log(" Hello".trimStart()); // "Hello"`
  - `console.log("Hello ".trimEnd()); // "Hello"`
  - **padStart(targetLength, padString)** — дополняет строку с начала до заданной длины.
  - `console.log("5".padStart(3, "0")); // "005"`
  - **padEnd(targetLength, padString)** — дополняет строку с конца.
  - `console.log("5".padEnd(3, "0")); // "500"`
- 

## 5. Разделение и объединение

- **split(separator, limit)** — разбивает строку на массив подстрок.
  - `let words = "Hello, World!".split(", "); // ["Hello", "World!"]`
  - **concat(...strings)** — объединяет строки (чаще используют оператор +).
  - `console.log("Hello".concat(", ", "World!")); // "Hello, World!"`
- 

## 6. Повторение строки

- **repeat(count)** — повторяет строку указанное количество раз.
  - `console.log("Hello".repeat(3)); // "HelloHelloHello"`
-

## 7. Работа с символами и кодировкой

- **codePointAt(pos)** — возвращает код символа в Юникоде (учитывает суррогатные пары).
  - `console.log("ㄱ".codePointAt(0)); // 134071`
  - **fromCodePoint(...codes)** — создает строку из кодов символов.
  - `console.log(String.fromCodePoint(134071)); // "ㄱ"`
- 

## 8. Сравнение строк

- **localeCompare(compareString, locales, options)** — сравнивает строки с учетом локализации.
  - `console.log("a".localeCompare("b")); // -1`
  - `console.log("b".localeCompare("a")); // 1`
  - `console.log("a".localeCompare("a")); // 0`
- 

### Примеры для практики

Если нужна помощь с реализацией или примерами, дайте знать!

Используя срезы, дополните приведённый ниже код так, чтобы он вывел последние

```
s = "In 2010, someone paid 10k Bitcoin for two pizzas."
print(s[-9:])
```

```
let str = "In 2010, someone paid 10k Bitcoin for two pizzas."
console.log(str.slice(str.length-9))
```

Используя срезы, дополните приведённый ниже код так, чтобы он вывел каждый 7-й символ строки `s` (начиная с 0-го индекса).

```
let str = "In 2010, someone paid 10k Bitcoin for two pizzas."
let row = ""
for (i=0; i <= str.length;i+=7){
    if (i < str.length ){
        row = row + str[i]
    }
}
console.log(row)
```

Используя срезы, дополните приведённый ниже код так, чтобы он вывел строку `s` в обратном порядке.

```
s = "In 2010, someone paid 10k Bitcoin for two pizzas."
print(s[::-1])
```

```
let str = "In 2010, someone paid 10k Bitcoin for two pizzas."
let array = Array.from(str)
```



```
console.log(array.reverse().join(""))
```

## Палиндром

На вход программе подаётся одно слово, записанное в нижнем регистре. Напишите программу, которая определяет, является ли оно палиндромом.

```
str = input()
if str == str[::-1]:
    print("Yes")
else:
    print("No")
```

```
let str = "assa"
let array = Array.from(str)
let array2 = array.reverse().join("")
if (str == array2){
    console.log("Yes")
}else{
    console.log("No")
}
```

## Делаем срезы 1

На вход программе подаётся одна строка. Напишите программу, которая выводит:

1. общее количество символов в строке;
2. исходную строку, повторённую 32 раза;
3. первый символ строки;
4. первые три символа строки;
5. последние три символа строки;
6. строку в обратном порядке;
7. строку с убойными первым и последним символами.

```
8. str = "abc"
str32 = str*32
print(len(str))
print(str32)
print(str[0])
print(str[:3])
print(str[-3:])
print(str[::-1])
print(str[0]+str+str[-1])
```

```
let str = "absd"

console.log(str.length)
console.log(str.repeat(32))
console.log(str[0])
```

```
console.log(str.slice(0,3))
console.log(str.slice(-3))
console.log(Array.from(str).reverse().join(""))
console.log(str[0]+str+str[str.length-1])
```

## Делаем срезы 2

На вход программе подаётся одна строка. Напишите программу, которая выводит:

1. третий символ этой строки;
2. предпоследний символ этой строки;
3. первые пять символов этой строки;
4. всю строку, кроме последних двух символов;
5. все символы с чётными индексами;
6. все символы с нечётными индексами;
7. все символы в обратном порядке;
8. все символы строки через один в обратном порядке, начиная с последнего.

```
str = "1234567890"
print(str[2])
print(str[-1])
print((str[:5]))
print((str[0:-2]))
print(x[0::2])
print(x[1::2])
print()
print(str[::-1])
print(str[::-1][::2])
```

```
/* 1. третий символ этой строки;
2. предпоследний символ этой строки;
3. первые пять символов этой строки;
4. всю строку, кроме последних двух символов;
5. все символы с чётными индексами;
6. все символы с нечётными индексами;
7. все символы в обратном порядке;
8. все символы строки через один в обратном порядке, начиная с последнего. */

let str = "12345678"
console.log(str[2])
console.log(str[str.length - 1])
console.log(str.slice(0,5))
console.log(str.slice(0,(str.length)-2))
let row = ""
for (i=0 ; i < str.length;i++){
    if ( i % 2 === 0){
        row = row + str[i]
    }console.log(row)
}
row = ""
```

```

for (let i = 0; i < str.length; i++) {
    if (i % 2 === 0) {
        row += str[i];
    }
}
console.log(row)

row = ""
for (i=0 ; i < str.length;i++){
    if ( i % 2 !== 0){
        row = row + str[i]
    }
}
console.log(row)

```

## Две половинки

На вход программе подаётся строка текста. Напишите программу, которая разрежет её на две равные части, переставит их местами и выведет на экран.

### Формат входных данных

На вход программе подаётся строка текста.

```

let str = prompt()
if (str.length > 1){
    let razdielitel = Math.floor(str.length / 2);
    if (str.length % 2 == 0){
        str = str.slice(-1*razdielitel)+str.slice(0,razdielitel)

    }else{
        str = str.slice(-1*razdielitel)+str.slice(0,razdielitel+1)
    }
    console.log(str)
}else{
    console.log(str)
}

```

```

str = input()
if len(str) == 1:
    print(str)
if len(str) % 2 == 0:
    d=len(str)//2
    print(str[d:]+str[:d])
else:
    d=len(str)//2
    print(str[d+1:]+str[:d+1])

```

## Заглавные буквы <sup>AB</sup><sub>CD</sub>

На вход программе подаётся строка, состоящая из имени и фамилии человека, разделённых одним пробелом. Напишите программу, которая проверяет, что имя и фамилия начинаются с заглавной буквы.

```
s = input()
if s == s.title():
    print("YES")
else:
    print("NO")
```

```
str = "Asdd Huu"
array = str.split(" ")
console.log(array)
if (array[0][0].toUpperCase() === array[0][0] && array[1][0].toUpperCase() === array[1][0]){
    console.log("Yes")
}else{
    console.log("No")
}
```

```
s = input().split()
if s[0][0].isupper() and s[1][0].isupper():
    print("YES")
else:
    print("NO")
```

## sWAP cASE ↻

На вход программе подаётся строка. Напишите программу, которая меняет регистр символов – заменяет все строчные символы заглавными и наоборот.

### Формат входных данных

На вход программе подаётся строка.

### Формат выходных данных

Программа должна вывести строку в соответствии с условием задачи.

```
s = input()
swapped_case = s.swapcase()
print(swapped_case)
```

```
let s = "Hello"
res = ""
for (let i = 0 ; i < s.length ; i++){
  if (s[i] === s[i].toUpperCase()){
    res = res + s[i].toLowerCase()
  }else{
    res = res + s[i].toUpperCase()
  }
}

console.log(res);
```

## Хороший оттенок 👍

На вход программе подаётся строка текста. Напишите программу, которая определяет, является ли оттенок текста хорошим или нет. Текст имеет хороший оттенок, если содержит подстроку «хорош» (без кавычек) во всех возможных регистрах.

### Формат входных данных

На вход программе подаётся строка текста

```
let str = "я очень хооший текст =)"
str = str.toLowerCase()
if (str.includes("хорош")){
  console.log("YES")
}else{
  console.log("NO")
}
```

## Нижний регистр ▼

На вход программе подаётся строка. Напишите программу, которая подсчитывает количество буквенных символов в нижнем регистре.

### Формат входных данных

На вход программе подаётся строка.

```
s = "ASD455ccccUIO";
counter = 0;
for (let i = 0 ; i < s.length ; i++){
  if ( 'abcdefghijklmnopqrstuvwxyz'.includes(s[i])){
```

```

        counter +=1
    }
}
console.log(counter)

```

```

s = input()
counter=0
for i in range(len(s)):
    if s[i] in 'abcdefghijklmnopqrstuvwxyz':
        counter=counter+1
print(counter)

```

## Количество цифр

На вход программе подаётся строка текста. Напишите программу, которая подсчитывает количество цифр в данной строке.

### Формат входных данных

На вход программе подаётся строка текста.

```

t = input()
s = 0
for n in range(10):
    s = s + t.count(str(n))
print(s)

```

```

let str = "as34fg78hn9"
let counter = 0
for (let i = 0 ; i < str.length ; i++){
    if ("0123456789".includes(str[i])){
        counter +=1
    }
}
console.log(counter)

```

## .com or .ru

На вход программе подаётся строка текста. Напишите программу, которая проверяет, что строка заканчивается подстрокой `.com` или `.ru`.

### Формат входных данных

```

let str = "www.google.r"
if (str.endsWith("com") || str.includes("ru")){
    console.log("YES")
}
console.log("NO")

```

## Самый частотный символ

На вход программе подаётся строка текста. Напишите программу, которая выводит на экран символ, который появляется наиболее часто.

### Формат входных данных

На вход программе подаётся строка текста. Текст может содержать строчные и заглавные буквы английского и русского алфавита, а также цифры.

```
s = "aaavvbadddddffa"
rep_s = 0
rep_letter = ""
for i in s:
    if s.count(i) > rep_s:
        rep_letter = i
        rep_s = s.count(i)
print(rep_letter)
```

```
s = "aaavvbadddddffa"
rep_s = 0
rep_letter = ""
for i in s:
    if s.count(i) > rep_s:
        rep_letter = i
        rep_s = s.count(i)
print(rep_letter)
```

## Первое и последнее вхождение

На вход программе подаётся строка текста. Если в этой строке буква «f» встречается только один раз, выведите её индекс. Если она встречается два и более раз, выведите индексы её первого и последнего вхождения на одной строке, разделённые символом пробела. Если буква «f» в данной строке не встречается, следует вывести «NO» (без кавычек).

```
let str = "afsdfigs sfg";
let firstIndex = str.indexOf('f');
let lastIndex = str.lastIndexOf('f');
if (firstIndex === -1) {
    console.log("NO");
} else if (firstIndex === lastIndex) {
    console.log(firstIndex);
} else {
    console.log(firstIndex, lastIndex);
}
```

```
t = "df ref ttfg"
if t.count("f") == 1 :
    print (t.find("f"))
elif t.count("f") > 0:
    print(t.find("f"),t.rfind("f"))
elif t.count("f") == 0:
    print("NO")
```

## Самое тяжёлое слово

Под "тяжестью" слова будем понимать сумму кодов по таблице Unicode всех символов этого слова. Напишите программу, которая принимает 44 слова и находит среди них **самое тяжёлое** слово. Если самых тяжёлых слов будет несколько, то программа должна вывести первое из них.

### Формат входных данных

На вход программе подаются 44 слова, каждое на отдельной строке.

### Формат выходных данных

Программа должна вывести самое тяжёлое слово в строке.

Тестовые данные ☐

### Sample Input 1:

строки  
списки  
кортежи  
множества

### Sample Output 1:

Множества

```
def weight(word):
    return sum([chr(i) for i in word])

#lst =[input() for _ in range(4)]
lst = ['строки', 'списки', 'кортежи', 'множества']
print(lst)
max_weight = 0
word_weight = ""
for i in lst:
    if weight(i) > max_weight:
        max_weight = weight(i)
        word_weight = i
print(word_weight)
```

```
let sumOrd= function(word){
```



```



    let suma = 0
    for (char of word){
        suma += char.charCodeAt(0)
    }
    return suma
}
let maxWeight = 0
let wordMaxWeight = ""
const lst = ['строки', 'списки', 'кортежи', 'множества']
for (word of lst){
    weight = sumOrd(word)
    if (weight > maxWeight){
        console.log(weight, word)
        maxWeight = weight
        wordMaxWeight = word
    }
}
console.log(maxWeight, wordMaxWeight)


```

`String.fromCharCode(code)` обратная функция

## Стоимость ответа


Модератору Сэму за **каждый символ** его сообщений в комментариях Тимур платит в  (пчёлках-coin) по следующему тарифу:

<код символа в таблице Unicode>×3.<код символа в таблице Unicode>×3.

А стоимость всего сообщения складывается из суммы стоимостей всех символов. Сэму захотелось подсчитать, сколько  он зарабатывает за свои ответы в комментариях, и просит вас помочь ему.

На вход программе подаётся строка текста. Требуется написать программу, которая находит стоимость сообщения Сэма в  и выводит текст в следующем формате:

Текст сообщения: '<текст сообщения Сэма>'


Стоимость сообщения: <стоимость сообщения Сэма>

### Формат входных данных

На вход программе подаётся строка текста – очередной ответ Сэма в комментариях.

### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

**Примечание.**  (пчёлка-coin) – виртуальная валюта команды BEEGEEK, которой Тимур расплачивается со своими сотрудниками.

### Sample Input 1:

@кодер 666, пишите в комментариях по делу, не засоряйте чат бредом

```
let str = "@кодер 666, пишите в комментариях по делу, не засоряйте чат бредом"
let price = 0
for (let char of str){
    price += char.charCodeAt()
}
console.log(price * 3)
```

```
str = "@кодер 666, пишите в комментариях по делу, не засоряйте чат бредом"
price = 0
for char in str:
    price += ord(char)*3
print(price)
```

В JavaScript массивы представляют собой упорядоченные коллекции данных. Они обладают рядом встроенных методов и функций, которые упрощают работу с элементами массива. Ниже приведен подробный обзор методов и функций, которые можно использовать с массивами.

---

## Создание массива

Примеры:

```
let arr = [1, 2, 3, 4]; // Литерал массива
let arr2 = new Array(5); // Создание массива длиной 5
let arr3 = Array.of(1, 2, 3); // Создание массива из значений
let arr4 = Array.from('hello'); // Создание массива из объекта-подобного массива
const lst = Array.from({ length: 6 }, (_, i) => i);
console.log(lst); // [0, 1, 2, 3, 4, 5]
```

---

## Основные методы массивов

### 1. Добавление и удаление элементов

- **push()** — добавляет элементы в конец массива.  
`arr.push(5); // [1, 2, 3, 4, 5]`
- **pop()** — удаляет последний элемент массива и возвращает его.  
`let last = arr.pop(); // last = 4, arr = [1, 2, 3]`
- **unshift()** — добавляет элементы в начало массива.  
`arr.unshift(0); // [0, 1, 2, 3]`

- **shift()** — удаляет первый элемент массива и возвращает его.
  - `let first = arr.shift(); // first = 1, arr = [2, 3]`
  - `Array(2).fill(numbers1).flat()`
- 

## 2. Поиск и работа с элементами

- **indexOf()** — возвращает индекс первого вхождения элемента.
  - `arr.indexOf(2); // 1`
  - **lastIndexOf()** — возвращает индекс последнего вхождения элемента.
  - `arr.lastIndexOf(2);`
  - **includes()** — проверяет, содержит ли массив элемент.
  - `arr.includes(3); // true`
- 

## 3. Итерации и преобразования

- **forEach()** — выполняет указанную функцию для каждого элемента массива.
  - `arr.forEach((item) => console.log(item));`
  - **map()** — создает новый массив, преобразуя элементы исходного.
  - `let squares = arr.map(x => x * x);`
  - **filter()** — создает новый массив из элементов, прошедших проверку.
  - `let even = arr.filter(x => x % 2 === 0);`
  - **reduce()** — сводит массив к одному значению.
  - `let sum = arr.reduce((acc, cur) => acc + cur, 0);`
  - **reduceRight()** — то же, что `reduce`, но работает справа налево.
- 

## 4. Сортировка и изменение массива

- **sort()** — сортирует массив (по умолчанию — строки).
  - `arr.sort((a, b) => a - b); // Для чисел: [1, 2, 3]`
  - **reverse()** — переворачивает массив.
  - `arr.reverse(); // [3, 2, 1]`
  - **splice()** — добавляет, удаляет или заменяет элементы массива.
  - `arr.splice(1, 1, 'new');` // Удаляет 1 элемент с индекса 1 и добавляет 'new'
  - **slice()** — возвращает новый массив, содержащий часть исходного.
  - `let subArray = arr.slice(1, 3); // [2, 3]`
- 

## 5. Объединение массивов

- **concat()** — объединяет массивы и возвращает новый массив.
  - `let combined = arr.concat([4, 5]); // [1, 2, 3, 4, 5]`
  - **flat()** — разворачивает вложенные массивы до указанной глубины.
  - `let nested = [1, [2, [3]]];`
  - `let flat = nested.flat(2); // [1, 2, 3]`
  - **flatMap()** — объединяет `map()` и `flat()`.
  - `arr.flatMap(x => [x, x * 2]); // [1, 2, 2, 4, 3, 6]`
-

## 6. Проверка массива

- **Array.isArray()** — проверяет, является ли объект массивом.
  - `Array.isArray(arr); // true`
- 

## 7. Объединение в строку

- **join()** — создает строку из элементов массива.
  - `let str = arr.join(', '); // "1, 2, 3"`
  - **toString()** — возвращает строковое представление массива.
  - `arr.toString(); // "1,2,3"`
- 

## 8. Поиск и проверка элементов

- **find()** — возвращает первый элемент, соответствующий условию.
  - `let found = arr.find(x => x > 2); // 3`
  - **findIndex()** — возвращает индекс первого элемента, соответствующего условию.
  - `let index = arr.findIndex(x => x > 2); // 2`
  - **every()** — проверяет, удовлетворяют ли все элементы условию.
  - `arr.every(x => x > 0); // true`
  - **some()** — проверяет, удовлетворяет ли хотя бы один элемент условию.
  - `arr.some(x => x > 2); // true`
- 

## 9. Копирование и заполнение

- **fill()** — заполняет массив указанным значением.
  - `arr.fill(0, 1, 3); // [1, 0, 0, 4]`
  - **copyWithin()** — копирует элементы внутри массива.
  - `arr.copyWithin(1, 2); // [1, 3, 4, 4]`
- 

# Список чисел

На вход программе подаётся одно число  $n$ . Напишите программу, которая выводит список `[1, 2, 3, ..., n]`.

### Формат входных данных

На вход программе подаётся одно натуральное число.

### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

Тестовые данные ☐

---

### Sample Input 1:

1

---

### Sample Output 1:

[1]

---

**Sample Input 2:**

5

---

**Sample Output 2:**

[1, 2, 3, 4, 5]

```
let lst = Array.from({length : 6} ,(_,i) =>i)
console.log(lst)
```

```
let lst = ""
for (let i =1 ;i <= 5 ; i++){
    lst = lst + i
}
lst = Array.from(lst)
console.log(lst,typeof lst)
```

```
lst = [ i for i in range(1,6)]
print(lst)
```

## Список букв

На вход программе подаётся натуральное число  $n$ . Напишите программу, которая выводит список, состоящий из  $n$  букв английского алфавита `['a', 'b', 'c', ...]` в нижнем регистре.

**Формат входных данных**

На вход программе подаётся натуральное число  $n$  ( $1 \leq n \leq 26$ ).

**Формат выходных данных**

Программа должна вывести текст в соответствии с условием задачи.

Тестовые данные ☐

---

**Sample Input 1:**

1

---

**Sample Output 1:**

['a']

---

**Sample Input 2:**

5

---

**Sample Output 2:**

['a', 'b', 'c', 'd', 'e']

```
let lst = Array.from({length : 5} ,(_,i) => String.fromCharCode(97 + i))
console.log(lst)
```

```
lst = [ chr(i+97) for i in range(1,6) ]
print(lst)
```

Дополните приведённый ниже код так, чтобы он вывел **сумму** минимального и максимального элементов списка `numbers`.

```
let numbers = [12.5, 3.1415, 2.718, 9.8, 1.414, 1.1618, 1.324]
console.log(Math.max(...numbers)+Math.min(...numbers))
```

```
numbers = [12.5, 3.1415, 2.718, 9.8, 1.414, 1.1618, 1.324]
print(max(numbers)+min(numbers))
```

Дополните приведённый ниже код так, чтобы он вывел среднее арифметическое элементов списка `evens`.

```
let evens = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
console.log((evens.reduce((a,b) => a + b ,0)/evens.length))
```

Используя операторы конкатенации (`+`) и умножения списка на число (`*`), дополните приведённый ниже код так, чтобы он вывел следующий список:

```
[1, 2, 3, 1, 2, 3, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 8, 9, 10, 11, 12, 13]
```

```
numbers1 = [1, 2, 3]
numbers2 = [6]
numbers3 = [7, 8, 9, 10, 11, 12, 13]
res = [...Array(2).fill(numbers1).flat()
      ,...Array(9).fill(numbers2).flat()
      ,...numbers3]
console.log(res)
```

```
numbers1 = [1, 2, 3]
numbers2 = [6]
numbers3 = [7, 8, 9, 10, 11, 12, 13]
print(numbers1*2+numbers2*9+numbers3)
```

## Список строк

На вход программе подаются натуральное число  $n$ , а затем  $n$  строк. Напишите программу, которая создаёт из указанных строк список, а затем выводит его.

### Формат входных данных

На вход программе подаются натуральное число  $n$ , а затем  $n$  строк, каждая на отдельной строке.

```
let lst = Array.from({length : 4}, (_, i) => prompt())
```

```
lst = [input() for i in range(4)]  
print(lst)
```

## Алфавит

Напишите программу, выводящую следующий список:

```
['a', 'bb', 'ccc', 'dddd', 'eeee', 'ffffff', ...]
```

### Формат выходных данных

Программа должна вывести указанный список.

**Примечание.** Последний элемент списка должен состоять из 2626 символов `z`.

```
lst = Array.from({length : 26}, (_, i) => String.fromCharCode(97+i).repeat(i+1))  
console.log(lst)
```

## Список кубов

На вход программе подаются натуральное число  $n$ , а затем  $n$  целых чисел. Напишите программу, которая создаёт из указанных чисел список их кубов, а затем выводит его.

### Формат входных данных

На вход программе подаются натуральное число  $n$ , а затем  $n$  целых чисел, каждое на отдельной строке.

### Формат выходных данных

Программа должна вывести список, состоящий из кубов указанных чисел.

```
n = Number(prompt())  
let lst = Array.from({length : n }, (_, i) => (i+1)**3 )  
console.log(lst)
```

```
n = int(input())  
d=[]  
for i in range(n):
```

```
numbers = int(input())
d.append(numbers**3)
print(d)
```

## Список делителей

На вход программе подаётся натуральное число  $n$ . Напишите программу, которая создаёт список, состоящий из делителей введённого числа, а затем выводит его.

### Формат входных данных

На вход программе подаётся натуральное число  $n$ .

### Формат выходных данных

Программа должна вывести список, состоящий из делителей введённого числа.

Тестовые данные ☐

#### Sample Input 1:

17

#### Sample Output 1:

[1, 17]

```
let lst = []
for (let i = 1; i <= 17; i++){
    if (17 % i == 0){
        lst.push(i)
    }
}
console.log(lst)
```

```
num = int(input())
d = []
for i in range(1, num+1):
    if num % i == 0:
        d.append(i)
print(d)
```

## Суммы двух

На вход программе подаётся натуральное число  $n(n \geq 2)$ . Затем поступают  $n$  целых чисел. Напишите программу, которая создаёт из указанных чисел список, состоящий из сумм соседних чисел (00 и 11, 11 и 22, 22 и 33 и так далее).

### Формат входных данных

На вход программе подаются натуральное число  $n(n \geq 2)$ , а затем  $n$  целых чисел, каждое на отдельной строке.

### Формат выходных данных

Программа должна вывести список, состоящий из сумм соседних чисел.



**Sample Input 1:**

5  
1  
2  
3  
4  
5

**Sample Output 1:**

[3, 5, 7, 9]

```
n = int(input("Введите натуральное число n (n >= 2): "))
numbers = []
for _ in range(n):
    num = int(input())
    numbers.append(num)
sums = [numbers[i] + numbers[i + 1] for i in range(n - 1)]
```

```
const n = parseInt(prompt("Введите натуральное число n (n >= 2):"), 10);
let numbers = [];
for (let i = 0; i < n; i++) {
    const num = parseInt(prompt(`Введите целое число ${i + 1}:`), 10);
    numbers.push(num);
}
let sums = [];
for (let i = 0; i < n - 1; i++) {
    sums.push(numbers[i] + numbers[i + 1]);
}
console.log(sums);
```

## Remove outliers

При анализе данных, собранных в рамках научного эксперимента, бывает полезно удалить самое большое и самое маленькое значение.

На вход программе подаются натуральное число  $n$ , а затем  $n$  различных натуральных чисел. Напишите программу, которая удаляет наименьшее и наибольшее значение из указанных чисел, а затем выводит оставшиеся числа каждое на отдельной строке, не меняя их порядок.

**Формат входных данных**

На вход программе подаются натуральное число  $n$ , а затем  $n$  различных натуральных чисел, каждое на отдельной строке.

### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

Тестовые данные ☐

---

#### Sample Input:

```
10
9
17
189
3
55
78
11
7
888
160
```

---

#### Sample Output:

```
9
17
189
55
78
11
7
160
```

```
const n = parseInt(prompt("Введите натуральное число n:"), 10);
let numbers = [];
for (let i = 0; i < n; i++) {
    const num = parseInt(prompt(`Введите целое число ${i + 1}:`), 10);
    numbers.push(num);
}
const minValue = Math.min(...numbers);
const maxValue = Math.max(...numbers);
const filteredNumbers = numbers.filter(num => num !== minValue && num !==
maxValue);
filteredNumbers.forEach(num => console.log(num));
```

```

n = int(input())
x1 = []
for i in range(n):
    x = int(input())
    x1.append(x)

del x1[x1.index(max(x1))]
del x1[x1.index(min(x1))]
print(*x1, sep="\n")

```

## Всё сразу 2 🐦☐

Дополните приведённый ниже код, чтобы он:

1. Заменяет второй (по порядку) элемент списка на `17`;
2. Добавляет числа `4`, `5` и `6` в конец списка;
3. Удаляет первый (по порядку) элемент списка;
4. Удвоил список;
5. Вставляет число `25` по индексу `3`;
6. Вывел список с помощью функции `print()`

**Примечание.** Под удвоением списка мы понимаем добавление к исходному списку всех его элементов, сохраняя порядок. Например, при удвоении списка `['py', 'gen']` мы получаем список `['py', 'gen', 'py', 'gen']`.

```

let numbers = [8, 9, 10, 11]
numbers[1] = 17;
numbers.push(4, 5, 6);
numbers.shift();
numbers = Array(2).fill(numbers).flat()
numbers.splice(3, 0, 25);
console.log(numbers);

```

```

numbers = [8, 9, 10, 11]
numbers[1] = 17
numbers.append(4)
numbers.append(5)
numbers.append(6)
del numbers[0]
numbers=numbers*2
numbers.insert(3,25)
print(numbers)

```

## Сортировка чисел

На вход программе подаётся строка текста, содержащая целые числа. Из данной строки формируется список чисел. Напишите программу, которая сортирует и выводит данный список сначала по возрастанию, а затем по убыванию.

### Формат входных данных

На вход программе подаётся строка текста, содержащая целые числа, разделённые символом пробела.

### Формат выходных данных

Программа должна вывести две строки текста в соответствии с условием задачи.

```
let lst = prompt().split(" ").map(Number)
let lstSorted = lst.sort((a,b) => a-b)
console.log(" ".join(lstSorted))
let lstReverse = lst.sort((a,b) => b-a)
console.log(lstReverse)
```

```
seq = []
for el in input().split():
    seq.append(int(el))

seq.sort()
print(*seq)

seq.reverse()
print(*seq)
```

```
let lst = [3,4,6,8,2,3]
let lstSorted = lst.sort((a,b) => a-b)
console.log(lstSorted.join(" "))
let lstReverse = lst.sort((a,b) => b-a)
console.log(lstReverse)
lst.forEach((elem) => console.log(elem))
```

```
let lst = "23456".split("").map(Number)
console.log(lst)
[ 2, 3, 4, 5, 6 ]
```

```
numbers = Array(2).fill(numbers).flat()
```

```
let lst = Array.from({length : n }, (_,i) => (i+1)**3 )
```

```
let lst = Array.from({length : 4}, (_,i) => i)
console.log(lst)
lst= Array(2).fill(lst).flat()
console.log(lst)
lst=lst.map(String)
console.log(lst)
lst = lst.sort((a,b)=>b-a)
console.log(lst)
lst.forEach((item) => console.log(item))
```

```
//lst = lst.forEach((item) => item **2)// wrong
lst = lst.map((item) => item ** 2);
console.log(lst)
let lst2 =lst.reduce((a,b) => a+b,0)
console.log(lst2)
lst.slice
/*• splice() — добавляет, удаляет или заменяет элементы массива.
• arr.splice(1, 1, 'new'); // Удаляет 1 элемент с индекса 1 и добавляет 'new'
• slice() — возвращает новый массив, содержащий часть исходного.
• let subArray = arr.slice(1, 3); // [2, 3]
*/
let lstFilter = lst1.filter((item) => item % 2 == 0)
console.log(lstFilter)
```

```
let lst = ['daf', 'br', 'crr', 'r', 'ab'];
lst = lst.sort((a, b) => b.localeCompare(a));
console.log(lst);
```

Функции в JavaScript (JS) — это ключевой элемент языка, позволяющий организовывать код в логические блоки, которые можно многократно использовать. Ниже приведено подробное объяснение функций в JavaScript:

## Что такое функция?

Функция — это блок кода, предназначенный для выполнения определённой задачи. Она может принимать входные данные (аргументы), выполнять операции и возвращать результат.

Основные свойства функций:

1. **Повторное использование:** Один раз написав функцию, её можно вызывать многократно.
2. **Модульность:** Функции помогают разбивать сложный код на более управляемые части.
3. **Читаемость:** Код становится легче для понимания, тестирования и поддержки.

## Создание функции

### 1. Объявление функции

С помощью ключевого слова `function`:

```
function имяФункции(аргументы) {
  // Тело функции
  return результат; // необязательно
}
```

Пример:

```
function сложить(a, b) {  
  return a + b;  
}  
  
console.log(сложить(2, 3)); // 5
```

## 2. Функциональные выражения

Функция может быть записана как выражение и сохранена в переменную:

```
const умножить = function(a, b) {  
  return a * b;  
};  
  
console.log(умножить(4, 5)); // 20
```

## 3. Стрелочные функции (Arrow Functions)

Компактный синтаксис для создания функций:

```
const разделить = (a, b) => a / b;  
  
console.log(разделить(10, 2)); // 5
```

- Если тело функции содержит одну строку, можно не использовать фигурные скобки {} и return.
- Если параметров нет, пишут (), например: () => 42.

---

# Параметры функции

## Параметры по умолчанию

Можно задавать значения по умолчанию для параметров:

```
function приветствие(имя = "Гость") {  
  console.log(`Привет, ${имя}!`);  
}
```

```
приветствие(); // Привет, Гость!  
приветствие("Анна"); // Привет, Анна!
```

## Остаточные параметры (...rest)

Позволяют передавать любое количество аргументов в виде массива:

```
function сумма(...числа) {  
  return числа.reduce((a, b) => a + b, 0);  
}  
  
console.log(сумма(1, 2, 3, 4)); // 10
```

---

# Возвращение значения

Функция может возвращать значение с помощью ключевого слова `return`:

```
function квадрат(число) {  
  return число * число;  
}  
  
console.log(квадрат(5)); // 25
```

Если `return` отсутствует, функция возвращает `undefined`.

---

## Область видимости (Scope)

1. **Локальная область видимости** Переменные, объявленные внутри функции, видимы только внутри этой функции.
  2. `function пример() {`
  3.   `let x = 10;`
  4.   `console.log(x); // 10`
  5. `}`
  6. `// console.log(x); // Ошибка: x не определён`
  7. **Глобальная область видимости** Переменные, объявленные вне функций, доступны во всём скрипте.
- 

## Замыкания

Замыкание — это функция, которая «запоминает» своё окружение, даже если выполняется вне него.

```
function создатьСчётчик() {  
  let счёт = 0;  
  return function() {  
    счёт++;  
    return счёт;  
  };  
}  
  
const счётчик = создатьСчётчик();  
console.log(счётчик()); // 1  
console.log(счётчик()); // 2
```

---

## Рекурсия

Функция может вызывать сама себя.

```
function факториал(n) {  
  if (n === 1) return 1;  
  return n * факториал(n - 1);  
}  
  
console.log(факториал(5)); // 120
```

---

# Методы и встроенные функции

Функции можно использовать как методы объектов:

```
const объект = {  
  имя: "Иван",  
  приветствие() {  
    console.log(`Привет, ${this.имя}!`);  
  },  
};
```

```
объект.приветствие(); // Привет, Иван!
```

---

## Особенности функций в JS

1. **Hoisting (Поднятие):** Функции, объявленные через `function`, поднимаются в начало области видимости.
  2. `привет(); // Привет!`
  3. `function привет() {`
  4.  `console.log("Привет!");`
  5. `}`
  6. **Анонимные функции:** Функции без имени.
  7. `setTimeout(function() {`
  8.  `console.log("Через секунду...");`
  9. `}, 1000);`
  10. **Функции высшего порядка (Higher-Order Functions):** Функции, принимающие другие функции как аргументы или возвращающие их.
  11. `function выполнить(func, a, b) {`
  12.  `return func(a, b);`
  13. `}`
  - 14.
  15. `console.log(выполнить(сложить, 2, 3)); // 5`
- 

Если у вас есть вопросы или требуется помощь с практическими примерами, дайте знать!



## Звёздный прямоугольник 1

Напишите функцию `draw_box()`, которая выводит звёздный прямоугольник с размерами  $14 \times 10$  в соответствии с образцом:

```
*****
```

```
*      *
```

```
*      *
```

```
*      *
```

```
*      *
```

```
*      *
```



```

*
*
*
*
*
*
*
*****

```

```

function drawBox(){
  str = "*".repeat(10)
  console.log(str)
  for (let i = 0; i <= 13 ; i++ ){
    console.log("*          *")
  }
  console.log(str)
}
drawBox()

```

```

def draw_box():
    for i in range(14):
        if i == 0 or i == 13:
            print("*"*10)
        else:
            print("*", " "*6, "*")
draw_box()

```

## Звёздный треугольник 1

Напишите функцию `draw_triangle()`, которая выводит звёздный прямоугольный треугольник с катетами, равными 1010, в соответствии с образцом:

```

*
*
*
*
*
*
*

```

```
*****
```

```
*****
```

```
*****
```

```
*****
```

```
function drawBox(){
  for (j = 1; j<=10;j++){
    str = "*".repeat(j)
    console.log(str)
  }
}
drawBox()
```

```
def draw_box():
    for i in range(1,11):
        print("*"*i)
draw_box()
```

## Звёздный треугольник

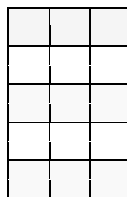
Напишите функцию `draw_triangle(fill, base)`, которая принимает два параметра:

- `fill` — символ заполнитель;
- `base` — величина основания равнобедренного треугольника;

а затем выводит его.

**Примечание.** Гарантируется, что основание треугольника – нечётное число.

Тестовые данные ☐



---

### Sample Input 1:

```
*
```

```
9
```

---

### Sample Output 1:

```
*
```

```
**
```

```
***
```

```
****
```

```
*****
****
***
**
*
```

```
function drawBox(fill,base){
  for (let j = 1; j <= Math.floor(base /2);j++){
    str = fill.repeat(j)
    console.log(str)
  }
  for (let j = Math.floor (base /2); j > 0;j--){
    str = fill.repeat(j)
    console.log(str)
  }
}
drawBox(" ",9)
```

```
def draw_thing(n,y):
    d=n//2
    for i in range(d+1):
        print(i*y)
    print ((d+1)*y)
    for i in range(d,0,-1):
        print(i*y)
a = input()
b = int(input())
draw_thing(b,a)
```

## ФИО

Напишите функцию `print_fio(name, surname, patronymic)`, которая принимает три параметра:

- `name` — имя человека;
- `surname` — фамилия человека;
- `patronymic` — отчество человека;

а затем выводит на печать ФИО человека.

```
function printFio(name, surname, patronymic){
  return name[0]+surname[0]+patronymic[0]
}
let name = prompt()
let surname = prompt()
```

```
let patronymic = prompt()
console.log(printFio(name, surname, patronymic))
```

```
def print_fio(name, surname, patronymic):
    full_name = (surname[0] + name[0] + patronymic[0]).upper()
    print(full_name)

name, surname, patronymic = input(), input(), input()

print_fio(name, surname, patronymic)
```

## Сумма цифр

Напишите функцию `print_digit_sum()`, которая принимает одно натуральное число `num` и выводит на печать сумму его цифр.

```
function printDigitalSum(Number){
    num = num.toString()
    return num.split('').map(item=>parseInt(item)).reduce((a,b) => a+b ,0)
}
let num = Number(prompt())
console.log((printDigitalSum(num)))
```

```
def print_digit_sum(num):
    print(sum(int(i) for i in str(num)))

num = int(input())
print_digit_sum(num)
```

## Делители 1

Напишите функцию `get_factors(num)`, принимающую в качестве аргумента натуральное число и возвращающую список всех делителей данного числа.

**Примечание.** Приведённый ниже код:

```
function getFactors(num){
    let res = []
    for (let i = 1; i <= num; i++){
        if (num % i ==0){
            res.push(i)
        }
    }
    return res
}
let num = Number(prompt())
console.log((getFactors(num).join(" ")))
```

```
def get_factors(num):
    result = []
    for i in range(1, num+1):
        if num % i == 0:
            result.append(i)
    return result
print(get_factors(int(input())))
```

## Найти всех

Напомним, что строковый метод `find('a')` возвращает местоположение первого вхождения символа `a` в строке. Проблема заключается в том, что данный метод не находит местоположение всех символов `a`.

Напишите функцию с именем `find_all(target, symbol)`, которая принимает два аргумента: строку `target` и символ `symbol` и возвращает список, содержащий все местоположения этого символа в строке.

**Примечание 1.** Если указанный символ не встречается в строке, то следует вернуть пустой список.

**Примечание 2.** Приведённый ниже код:

```
print(find_all('abcdabcaaa', 'a'))
```

```
function find_all(target, symbol) {
    let res = [];
    let start = 0;

    while (start < target.length) {
        let placeOfIndex = target.indexOf(symbol, start);

        if (placeOfIndex === -1) {
            break;
        }

        res.push(placeOfIndex);
        start = placeOfIndex + 1;
    }

    return res;
}

let target = prompt("Введите строку для поиска:");
let str = prompt("Введите символ для поиска:");
console.log(find_all(target, str));
```

```
def find_all(target,symbol):
    d=[]
    for i in range(len(target)):
        if symbol in target[i]:
            d.append(i)
    return d
s = input()
char = input()
print(find_all(s, char))
```

## Merge lists 1

Напишите функцию `merge(list1, list2)`, которая принимает в качестве аргументов два отсортированных по возрастанию списка, состоящих из целых чисел, и объединяет их в один отсортированный список.

**Примечание 1.** Списки `list1` и `list2` могут иметь разную длину.

**Примечание 2.** Можно использовать списочный метод `sort()`, а можно обойтись и без него. 🤖

**Примечание 3.** Приведённый ниже код:

```
print(merge([1, 2, 3], [5, 6, 7, 8]))
print(merge([1, 7, 10, 16], [5, 6, 13, 20]))
```

```
function merge(list1, list2){
    return list1.concat(list2).sort((a,b) => a-b)
}
let list1 = prompt().split(" ")
let list2 = prompt().split(" ")
console.log(merge(list1,list2))
```

```
def list_n(lst):
    return sorted(lst)

def merge(numbers1,numbers2):
    lst=list_n(numbers1)+list_n(numbers2)
    return sorted(lst)

numbers1 = [int(c) for c in input().split()]
numbers2 = [int(c) for c in input().split()]

print(merge(list_n(numbers1), list_n(numbers2)))
```