

Циклические  
алгоритмы.

Цикл с  
постусловием.

Инструкции  
управления циклом

Основные алгоритмические  
структуры



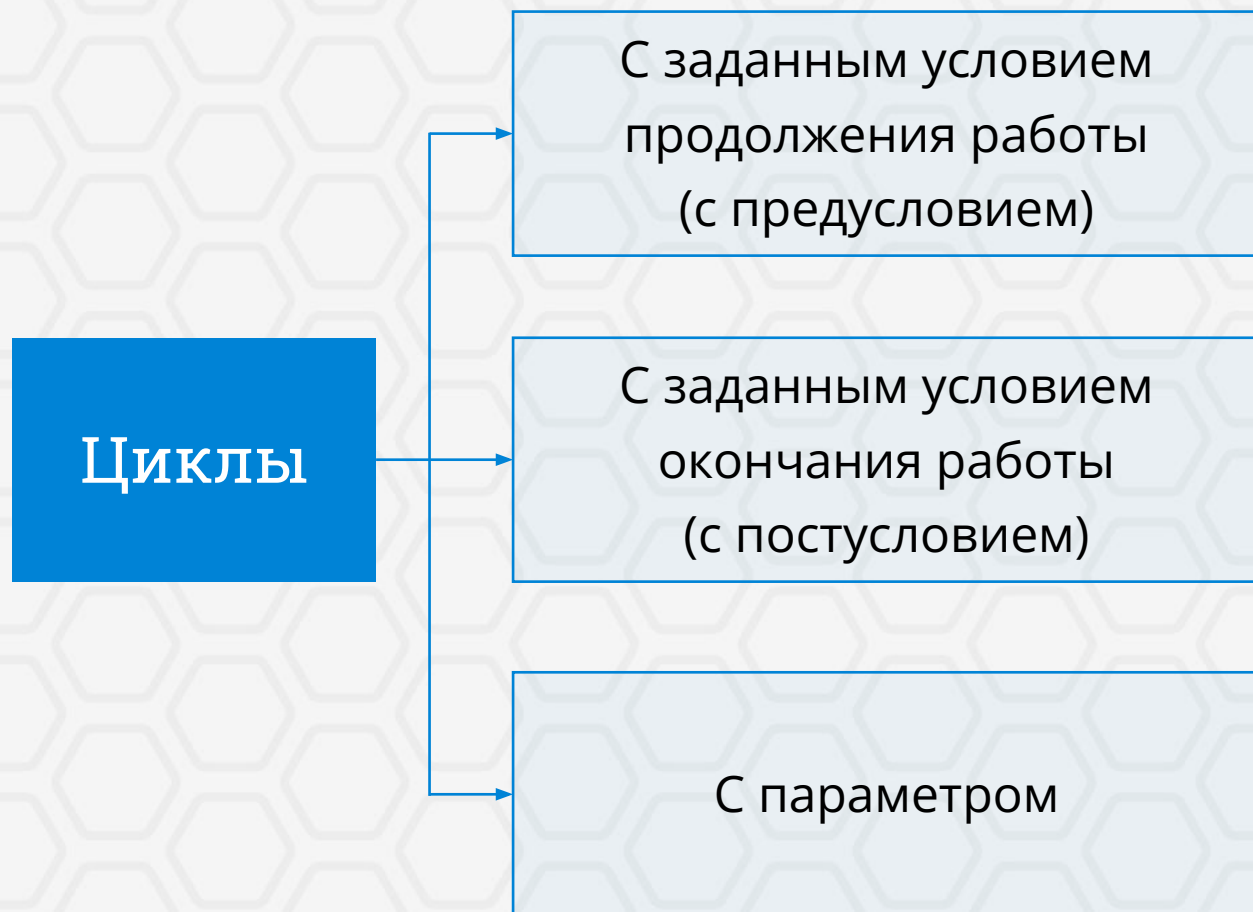
# Циклические алгоритмы

**Циклические алгоритмы** — это алгоритмы, содержащие циклы.

**Цикл** — это алгоритмическая конструкция, которая представляет собой последовательность действий, повторяющихся многократно.



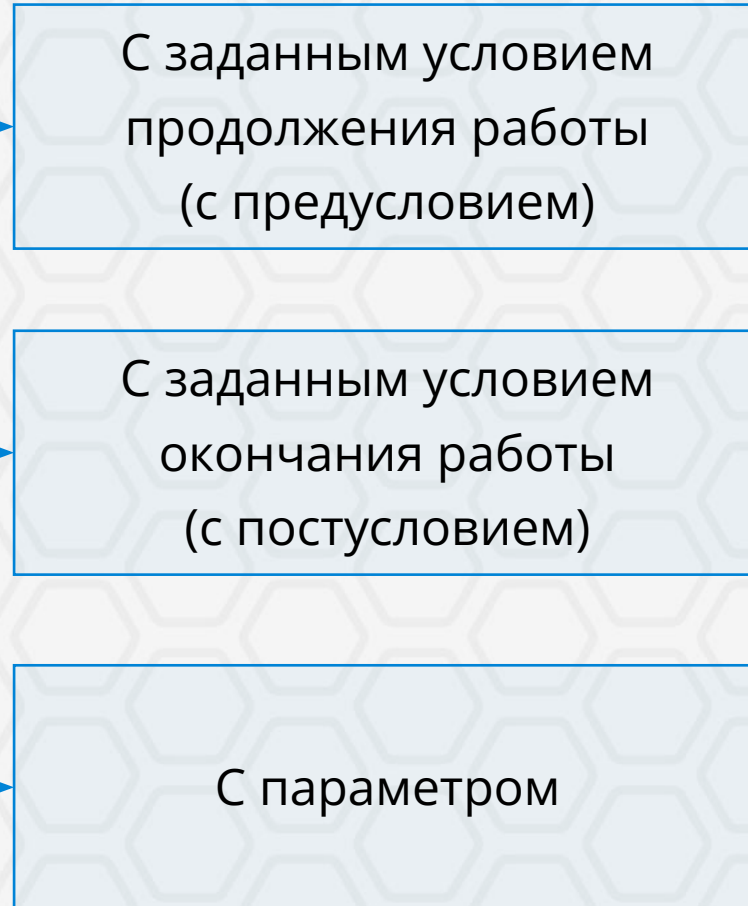
# Виды циклов



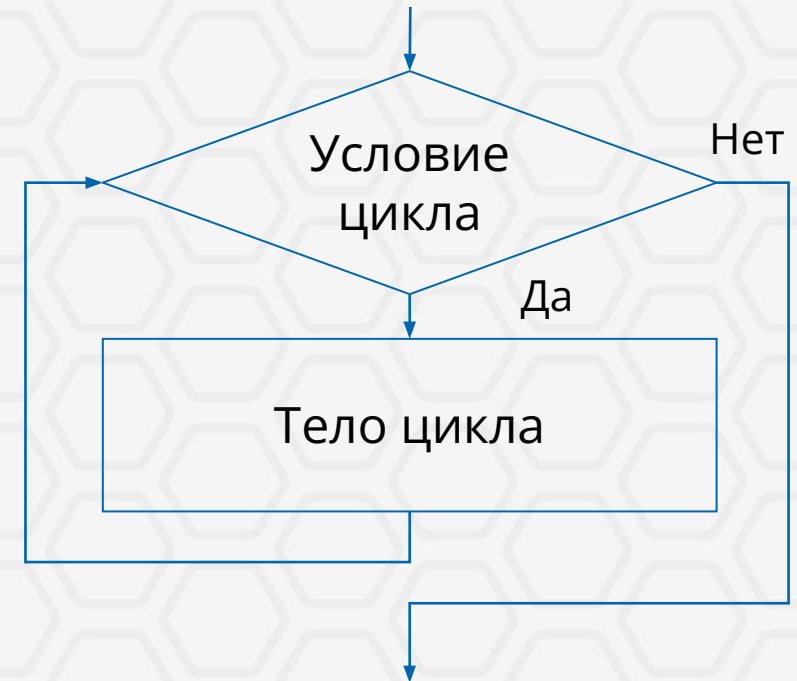


# Виды циклов

## Циклы



## Блок-схема цикла с предусловием:



# Вопросы к изучению

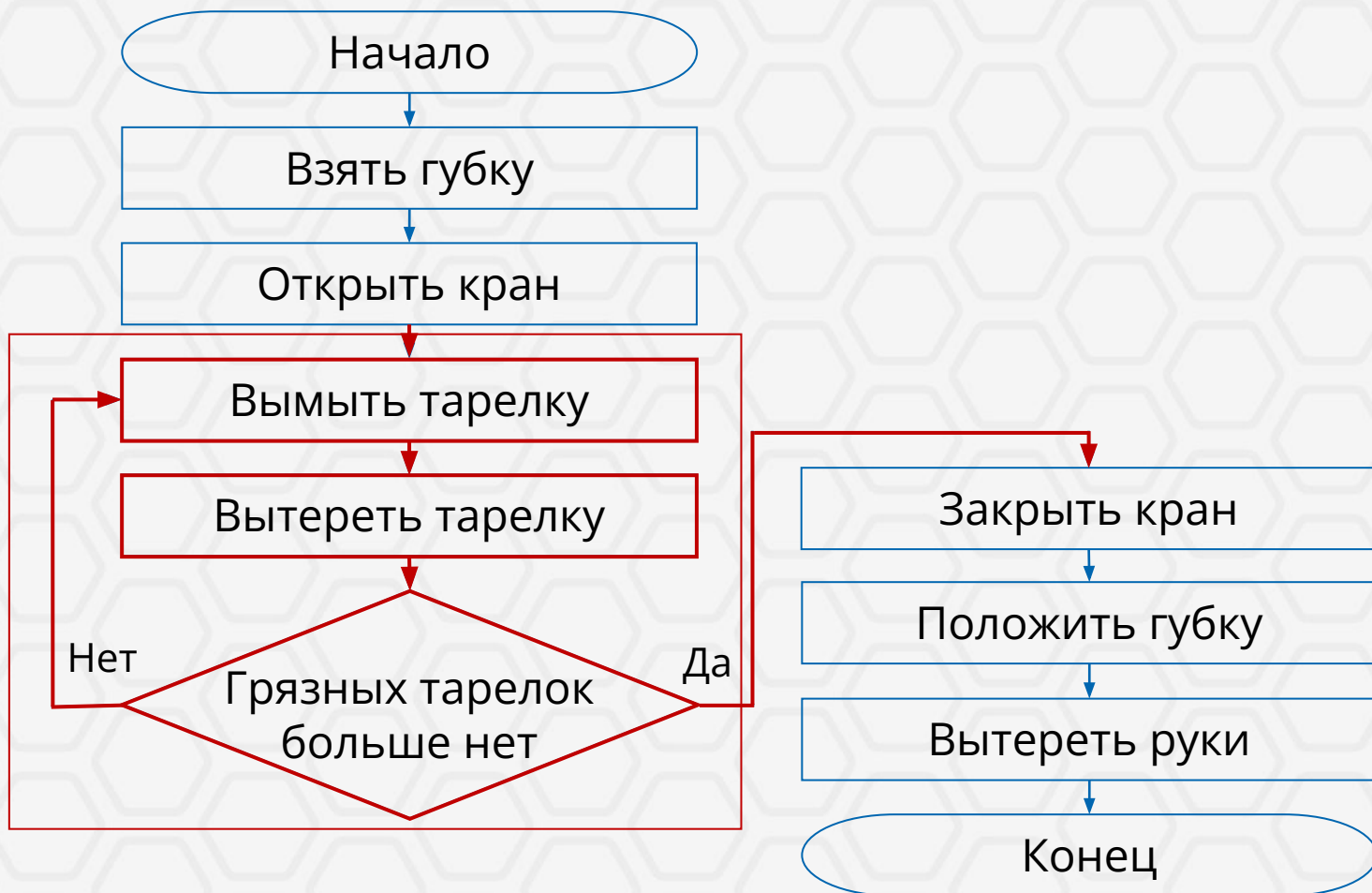
1

Цикл с  
постусловием.

2

Инструкции  
управления циклом.

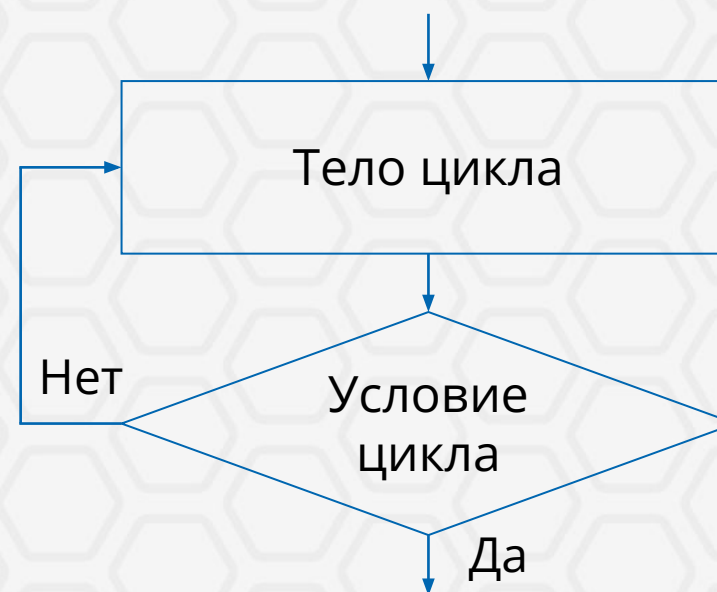
# Алгоритм мытья посуды





## Цикл с постусловием:

- ✓ работает до тех пор, пока не выполнится его условие;
- ✓ в любом случае выполняется в программе хотя бы один раз.



## Инструкции управления циклом

### break

останавливает исполнение  
текущего цикла и переходит  
к исполнению команд,  
следующих после него.





# Запись цикла с постусловием

В языке Python нет отдельной инструкции для записи цикла с постусловием, поэтому он реализуется через бесконечный цикл с предусловием:

```
while True:  
    <инструкция 1>  
    <инструкция 2>  
    ...  
    <инструкция n>  
    if <условие цикла>:  
        break
```



# Задача

Написать программу, которая вычисляет наибольший общий делитель двух целых положительных чисел.

**Наибольший общий делитель (НОД)** — это наибольшее число, на которое без остатка делятся оба числа.



**Эвклид**  
(3 век до н.э.)

## Усовершенствованный алгоритм Эвклида

Наибольшее число заменяется своим остатком от деления на наименьшее число до тех пор, пока одно из чисел не станет равным нулю.

После этого ненулевое число — это наибольший общий делитель исходных чисел.

## Цикл с блоком «else»:

**while** <условие>:  
<инструкция 1>  
<инструкция 2>  
...  
<инструкция n>

**else**:  
<инструкция n + 1>  
<инструкция n + 2>  
...  
<инструкция m>

Выполняется, если  
цикл завершил  
свою работу без  
инструкции **break**.





## Инструкции управления циклом

### break

останавливает исполнение текущего цикла и переходит к исполнению команд, следующих после него.

### continue

останавливает текущее исполнение тела цикла и переходит к следующему, начиная с проверки условия.



# Задача

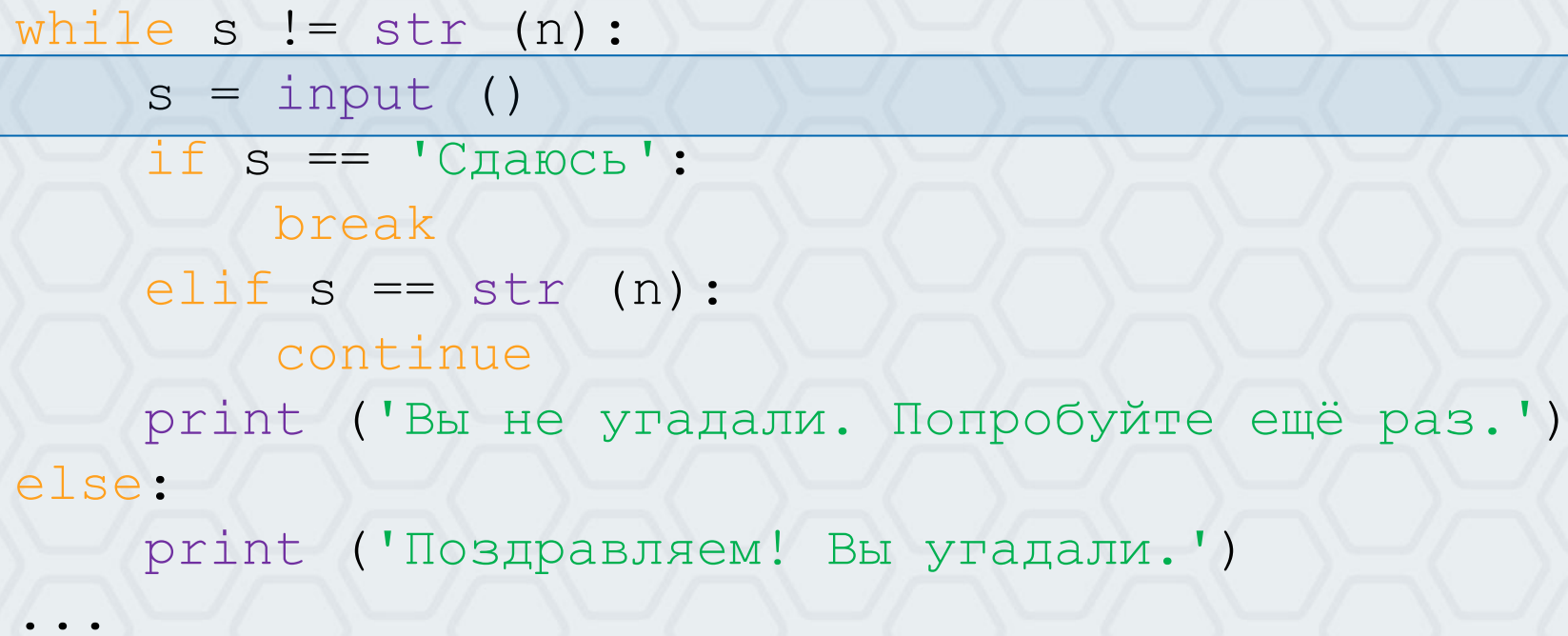
Написать программу, генерирующую случайное целое число на промежутке **[a; b]**, предоставляющую пользователю неограниченное число попыток для того, чтобы его угадать. Если пользователю надоело угадывать, то он должен ввести слово «**Сдаюсь**».

$$a < b$$



# Принцип работы цикла

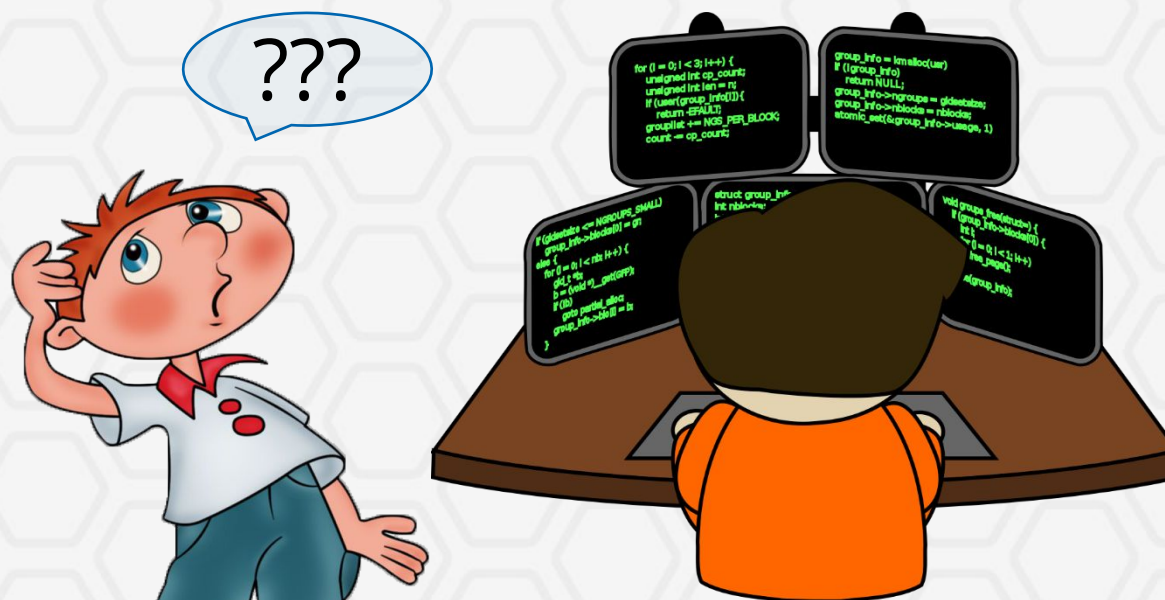
```
while s != str (n):  
    s = input ()  
    if s == 'Сдаюсь':  
        break  
    elif s == str (n):  
        continue  
    print ('Вы не угадали. Попробуйте ещё раз.')  
else:  
    print ('Поздравляем! Вы угадали.')  
...
```





# Понятность кода

**Инструкции управления циклом** следует использовать аккуратно, так как они могут затруднять читаемость кода.



# Циклические алгоритмы. Цикл с постусловием. Инструкции управления циклом

## Инструкции управления циклом:

- ✓ завершение работы цикла – **break**;
- ✓ прерывание текущего исполнения цикла – **continue**.

## Блок «else» в цикле

выполняется, если исполнение цикла было завершено без помощи инструкции завершения работы цикла.



# Циклические алгоритмы.

## Цикл с постусловием.

### Инструкции управления циклом

#### Цикл с постусловием:

- ✓ работает до тех пор, пока не выполнится его условие, которое проверяется после исполнения тела цикла;
- ✓ в любом случае выполняется в программе хотя бы один раз.



#### Цикл с постусловием на языке Python:

**while True:**

<тело цикла>

**if** <условие цикла>:

**break**