СОДЕРЖАНИЕ

Циклы в программировании

Представьте, что вам нужно 100 раз вывести на экран одну и ту же фразу. Вручную это делать неудобно и неоптимально. Циклы — инструмент, который позволяет программе многократно повторять определённые действия. Они помогают автоматизировать однотипные задачи и решать их быстрее и проще. Сегодня мы разберём цикл while, его синтаксис, работу с итерациями и познакомимся с полезными операторами для управления циклами.

Цикл while и его синтаксис

Цикл **while** повторяет блок кода до тех пор, пока выполняется заданное условие. Вот его базовый синтаксис:

```
while условие:
    # код, который будет выполняться, пока условие истинно
```

Условие — это логическое выражение, которое возвращает True или False. Пока условие остаётся истинным (True), программа продолжает выполнять блок кода.

Когда условие становится ложным (False), цикл прекращает свою работу, и программа переходит к следующему фрагменту кода.

```
number = 5
while number > 0:
    print(number)
    number -= 1  # уменьшаем значение на 1
```

Этот цикл выводит числа от 5 до 1. Как только number становится 0, условие становится ложным, и цикл завершает свою работу.

Итерация и создание счётчика

Итерация — это одно выполнение тела цикла. В каждом цикле важно контролировать количество итераций, чтобы избежать бесконечных циклов.

Для этого часто создают счётчик, который изменяется с каждой итерацией:

```
counter = 1
while counter <= 5:
    print(f"Итерация {counter}")
    counter += 1
```

В этом примере счётчик counter увеличивается на 1 в каждой итерации. Когда он достигает 6, цикл завершается.

Нахождение суммы цифр числа

Задачи, связанные с разбиением числа на цифры, можно решить при помощи цикла while. Например, для нахождения суммы цифр числа:

```
number = int(input())  # пользователь вводит число
sum_digits = 0

while number > 0:
    last_digit = number % 10  # получаем последнюю цифру
    sum_digits += last_digit  # добавляем её к сумме
    number //= 10  # уменьшаем число, убирая последнюю цифру

print("Сумма цифр:", sum_digits)
```

Этот цикл делит число на разряды, последовательно извлекая каждую цифру и суммируя их.

Вложенные циклы

Иногда в одном цикле нужно запустить другой цикл. Это называется **вложенным циклом**. Например:

```
i = 1
while i <= 3:
    j = 1
    while j <= 3:
        print(f"i = {i}, j = {j}")
        j += 1
    i += 1
```

Операторы break и continue

Оператор **break** завершает выполнение текущего цикла досрочно:

```
count = 0
while count < 10:
    if count == 5:
        break  # цикл завершится, когда count станет 5
    print(count)
    count += 1
```

Здесь цикл прерывается, как только счётчик достигает 5.

Оператор **continue** пропускает текущую итерацию цикла, переходя к следующей:

```
count = 0
while count < 5:
    count += 1
    if count == 3:
        continue  # цикл пропустит вывод числа 3
    print(count)
```

Важно понимать, что эти операторы влияют только на тот цикл, в котором они были вызваны. Если они используются внутри вложенного цикла, они не затрагивают внешний цикл.

Бесконечный цикл while

Иногда возникает необходимость в бесконечном цикле. Это цикл, который никогда не завершится самостоятельно, если его не прервать с помощью break или других условий:

```
while True:
    name = input("Введите имя: ")
    if name == "exit":
        break  # цикл прервётся, если пользователь введёт "exit"
    print(f"Привет, {name}!")
```

Бесконечные циклы часто используются для программ, которые должны постоянно ожидать ввод данных или отслеживать события.