Методичка по циклам Python

while • for • range • break/continue • типовые ошибки • практика

Цель: научиться писать циклы осознанно — понимать, когда нужен while, когда for, как избегать бесконечных циклов, и как решать реальные задачи (валидация ввода, подсчёты, работа со строками).

1. Что такое цикл и зачем он нужен

Цикл — это повторение одного и того же блока кода несколько раз. Без циклов пришлось бы писать однотипные строки вручную.

Два главных типа задач:

- Повторять, пока условие истинно (мы не знаем заранее, сколько раз). Обычно это while.
- Повторить N раз или пройтись по последовательности (диапазон, строка и т.д.). Обычно это for.

Правило выбора: если заранее известен счётчик/диапазон — for. Если цикл заканчивается событием (ввели правильный пароль, встретили стоп-значение) — чаще while.

2. Цикл while

2.1 Базовый шаблон while

У while всегда должны быть: старт, условие продолжения, действие, изменение (чтобы цикл когда-нибудь завершился).

```
i = 1                # старт: с какого значения начинаем
while i <= 5:        # условие: пока оно True - цикл продолжается
    print(i)         # действие: что делаем каждый шаг
    i += 1           # изменение: шаг (без него будет бесконечный цикл)
```

Почему важно i += 1? Потому что иначе условие i <= 5 никогда не изменится.

2.2 Бесконечный цикл: как возникает

Бесконечный цикл — когда условие всегда True. Обычно причина: вы забыли менять переменную, от которой зависит условие.

Если вы вдруг зациклились в терминале: чаще всего помогает Ctrl + C (остановить программу).

2.3 Валидация ввода (реальный кейс)

Типичный сценарий: пользователь вводит данные, пока не введёт корректно. Удобнее всего делать через while True + break.

```
while True:
    s = input("Введи число от 1 до 10: ")

    if not s.isdigit():           # проверяем, что это цифры
        print("Это не число.")
        continue                  # просим ввод снова

    x = int(s)                    # преобразуем в int

    if x < 1 or x > 10:           # проверяем диапазон
        print("Нужно от 1 до 10.")
        continue

    print("Ок:", x)               # всё хорошо
    break                         # выходим из цикла
```

2.4 break и continue

break — немедленно выходит из цикла. continue — пропускает текущую итерацию.

```
# break: останавливаем ввод по стоп-значению
total = 0
while True:
    x = int(input("Число (0 - стоп): "))
    if x == 0:
        break
    total += x
print("Сумма:", total)
```

```
# continue: пропускаем нежелательные значения
total = 0
while True:
    x = int(input("Число (0 - стоп): "))
    if x == 0:
        break
    if x < 0:
        continue       # отрицательные не суммируем
    total += x
print("Сумма положительных:", total)
```

2.5 Частые паттерны while

- Счётчик: i = 0; while i < N: ...; i += 1
- Ввод до стоп-значения: while True + break при стопе
- Повтор до правильного ответа: while answer != correct
- Валидация ввода: while True + continue на ошибках + break на успехе

3. Цикл for

3.1 range: как работает

Чаще всего for используется с range. Важно: правая граница не включается.

```
for i in range(5):      # 0,1,2,3,4
    print(i)

for i in range(1, 6):   # 1,2,3,4,5
    print(i)

for i in range(0, 11, 2):   # шаг 2: 0,2,4,6,8,10
    print(i)

for i in range(5, 0, -1):   # обратный цикл: 5,4,3,2,1
    print(i)
```

3.2 break/continue в for

```
# Найти первое число, кратное 7
for x in range(1, 101):
    if x % 7 == 0:
        print("Первое кратное 7:", x)
        break

# Вывести 1..50, пропуская кратные 3
for i in range(1, 51):
    if i % 3 == 0:
        continue
    print(i)
```

3.3 Когда for лучше, чем while

Если вы точно знаете количество повторений или идёте по готовой последовательности, for короче и безопаснее (меньше шансов забыть шаг).

4. Работа со строкой в цикле

Строка — это последовательность символов, поэтому по ней можно итерироваться.

```
word = "Python"
for ch in word:
    print(ch)
```

Если нужен индекс символа — используем len и range:

```
word = "Python"
for i in range(len(word)):
    print(i, word[i])
```

Пример: посчитать пробелы и e/E:

```
text = input("Текст: ")
spaces = 0
e_count = 0

for ch in text:
    if ch == " ":
        spaces += 1
    if ch == "e" or ch == "E":
        e_count += 1

print("Пробелов:", spaces)
print("Букв e/E:", e_count)
```

5. Вложенные циклы и паттерны

Вложенный цикл — цикл внутри цикла. Обычно внешний отвечает за «строки», внутренний — за элементы внутри каждой строки.

5.1 Лестница из звёздочек

```
n = int(input("n: "))

for i in range(1, n + 1):     # сколько строк
    print("*" * i)            # печатаем i звёздочек
```

5.2 Таблица умножения (форматированная)

```
for a in range(1, 6):
    for b in range(1, 6):
        print(a * b, end="\t")   # end="\t" - не переносим строку
    print()                       # перенос строки после каждой строки таблицы
```

Важно: параметр end у print меняет то, чем заканчивается вывод. По умолчанию это перенос строки.

6. Топ ошибок и как дебажить циклы

- Бесконечный while: забыли изменить переменную условия (нет i += 1 или нет нового ввода).
- Неправильный range: ожидали 1..10, а написали range(10) и получили 0..9.
- Смешали типы: input() даёт строку, а вы сравниваете с числом без int().
- Ошибки отступов: тело цикла должно быть с отступом 4 пробела.
- break не там: поставили break слишком рано и цикл всегда обрывается.

Мини-дебаг приём:

Если непонятно, почему цикл работает странно — временно добавьте печать ключевых переменных на каждом шаге:

```
i = 0
while i < 3:
    print("DEBUG i =", i)
    i += 1
```

7. Шпаргалка: готовые шаблоны

Задача	Шаблон
Повторить N раз	for i in range(N): ...
1..N включительно	for i in range(1, N+1): ...
Обратный цикл	for i in range(N, 0, -1): ...

Пока условие True	while condition:
	  ...
	  update

Ввод до стоп-значения	while True:
	  x = ...
	  if x == stop: break
	  ...

Пропустить итерацию	if bad: continue
Выйти сразу	if done: break
Идём по строке	for ch in text: ...

8. Практика и домашка

Практика (в классе)

- A1: вывести 1..20 (for).
- A2: вывести 20..1 (for, шаг -1).
- A3: вывести чётные 0..30 (range с шагом 2).
- B4: ввод чисел до 0, вывести сумму (while True + break).
- B5: пароль до 'qwerty' (while).
- C6: посчитать пробелы и e/E (for по строке).
- C7: вывести индекс и символ (range(len)).
- D8: лестница '*' (for + умножение строки).

Домашка (5 задач)

- Д31: Таблица умножения на N (N*1..N*10).
- Д32: Сумма цифр числа (через строку или через %10 и //10).
- Д33: Угадай число (secret=7): больше/меньше, пока не угадает.
- Д34: Подсчёт гласных (a e i o u в обоих регистрах).
- Д35: 1..50, пропуск кратных 3 (continue).

9. Ответы (для преподавателя)

Эту секцию можно не показывать студентам сразу. Используй для разборов/проверки.

A1

```
for i in range(1, 21):
    print(i)
```

A2

```
for i in range(20, 0, -1):
    print(i)
```

A3

```
for i in range(0, 31, 2):
    print(i)
```

B4

```
total = 0
while True:
    x = int(input("Число (0 - стоп): "))
    if x == 0:
        break
    total += x
print("Сумма:", total)
```

B5

```
correct = "qwerty"
pwd = input("Пароль: ")

while pwd != correct:
    print("Неверно. Ещё раз.")
    pwd = input("Пароль: ")

print("OK")
```

C6

```
text = input("Введите строку: ")
spaces = 0
e_count = 0

for ch in text:
    if ch == " ":
        spaces += 1
    if ch == "e" or ch == "E":
        e_count += 1

print("Пробелов:", spaces)
print("Букв e/E:", e_count)
```

C7

```
text = input("Введите строку: ")

for i in range(len(text)):
    print(i, text[i])
```

D8

```
n = int(input("n: "))

for i in range(1, n + 1):
    print("*" * i)
```