IT CAREER HUB

Урок 9
Цикл for

IT CAREER HUB

Итерируемый объект

Итерируемый объект (или последовательность) — это объект, который может состоять из множества элементов и предоставляет их по одному, когда это необходимо.

Примером итерируемых объектов являются строки, так как их можно "разбить" на отдельные символы. Итерируемые объекты позволяют легко обрабатывать данные по отдельным элементам.

Цикл for

Цикл for — это конструкция, которая позволяет повторно выполнять блок кода (итерацию) для каждого элемента в последовательности.

Он последовательно проходит по каждому элементу, и повторяет определённый набор действий для каждого из них. Цикл будет выполнять блок кода до тех пор, пока не переберёт все элементы в последовательности, указанной в условии.

Синтаксис

```python
for переменная in последовательность:  # условие цикла

    # блок кода, выполняющийся для каждого элемента
```

- **Переменная** — это переменная, которая на каждом шаге цикла присваивает значение текущего элемента из последовательности.
- **Последовательность** — это итерируемый объект (например, строка), по которому цикл проходит.
- **Блок кода (или тело цикла)** — это инструкции, которые будут выполняться для каждого элемента последовательности.

Условие цикла

Цикл for будет работать до тех пор, пока есть элементы в последовательности, которые можно обработать. Как только все элементы будут пройдены, цикл завершится.

Пример

```python
text = "Python"

for letter in text:
```

```
    print(letter)
```

В каждой итерации переменная letter получает следующий символ строки, а затем выполняется блок кода — в данном случае это команда `print(letter)`, которая выводит этот символ.

Итерация

Итерация — это один шаг в процессе последовательного перебора элементов итерируемого объекта. На каждой итерации происходит обработка одного элемента из последовательности.

IT CAREER HUB

Задания для закрепления 1

Какой результат будет выведен при выполнении следующего кода?

```python
text = "Hello"

for letter in text:

    print(letter, end="")
```

a. H e l l o
b. Hello
c. H на новой строке
d. Ошибка

Посмотреть ответ

IT CAREER HUB

Функция range

Функция range() используется для создания последовательности чисел, которые можно использовать в цикле. Она позволяет задавать диапазоны чисел и управлять шагом между ними.

Синтаксис range

```python
range(start, stop, step)
```

- **start** (необязательный) — начальное значение (включительно). Если не указано, по умолчанию это 0.
- **stop** (обязательный) — конечное значение (не включается в последовательность).
- **step** (необязательный) — шаг, с которым создаётся последовательность. По умолчанию равен 1.

range с одним аргументом (stop):

Если указать только конечное значение, последовательность начинается с 0 и идёт до указанного числа (не включительно).

```python
for i in range(5):

    print(i)
```

range с двумя аргументами (start, stop):

Если указать два значения, последовательность начинается с первого (включительно) и идёт до второго числа (не включительно).

```python
for i in range(2, 6):
```

```
    print(i)
```

range с тремя аргументами (start, stop, step):

Когда используется три аргумента, `range()` создаёт последовательность, начиная с числа `start` (включительно), заканчивая числом `stop` (не включается), с шагом `step`, который указывает, через сколько элементов нужно брать следующее число.

```python
for i in range(1, 10, 2):

    print(i)
```

В этом примере шаг равен 2, поэтому цикл выводит числа через одно.

range с отрицательным шагом:

Функция `range()` также поддерживает отрицательные значения для шага `step`, что позволяет создавать последовательности чисел в обратном порядке. В этом случае `start` должно быть больше `stop`, чтобы значения уменьшались с каждым шагом.

Примеры с отрицательным шагом

```python
for i in range(10, 0, -2):

    print(i)



for i in range(-4, -8, -1):

    print(i)
```

IT CAREER HUB

IT CAREER HUB

Задания для закрепления 2

1. Какой результат будет выведен при выполнении следующего кода?

```python
for i in range(5):

    print(i)
```

a. Числа 1 2 3 4 5 на одной строке
b. Числа 0 1 2 3 4 на разных строках
c. Числа 1 2 3 4 5 на разных строках
d. Ошибка

Посмотреть ответ

2. Какой результат будет выведен при выполнении следующего кода?

```python
for i in range(1, 10, 2):

    print(i)
```

a. Числа 1 2 3 4 5 6 7 8 9 на одной строке
b. Числа 1 3 5 7 9 на разных строках
c. Числа 2 4 6 8 10 на разных строках
d. Ошибка

Посмотреть ответ

3. Какой результат будет выведен при выполнении следующего кода?

```python
for i in range(10, 0, -2):

    print(i)
```

a. Числа 10 8 6 4 2 0 на разных строках
b. Числа 10 8 6 4 2 на разных строках
c. Числа 9 7 5 3 1 на разных строках
d. Ошибка

Посмотреть ответ

Операторы break, continue, else в цикле for

В Python цикле for можно использовать специальные операторы — break, continue и else — для управления выполнением цикла. Операторы работают также, как и в цикле while.

Оператор break

Оператор break позволяет прервать выполнение цикла досрочно, как только будет выполнено определённое условие. Цикл завершится, даже если элементы в последовательности ещё остались.

Пример

```python
for letter in "Python":

    if letter == "h":

        break  # Останавливаем цикл, если найден символ "h"

    print(letter)
```

Оператор continue

Оператор `continue` позволяет пропустить текущую итерацию цикла и перейти к следующей, не завершая сам цикл. Он используется, когда нужно игнорировать определённые элементы, но продолжить обработку остальных.

```python
Пример:

for letter in "Python":

    if letter == "h":
```

```
        continue  # Пропускаем букву "h" и продолжаем цикл
    print(letter)
```

Оператор else

Оператор else в цикле `for` выполняет блок кода, если цикл завершился нормально, без использования оператора `break`. Это полезно, когда нужно выполнить определённые действия, если цикл прошёл через все элементы без прерывания.

Пример

Python

```python
for letter in "Python":
    if letter == "a":
        break  # Этот код никогда не выполнится, так как "a" нет в строке
    print(letter)
else:
    print("Цикл завершён нормально.")
```

IT CAREER HUB

Пример с break и else

Python

```python
for letter in "Python":
    if letter == "h":
        break  # Цикл прерывается на символе "h"
    print(letter)
else:
    print("Цикл завершён нормально.")  # Этот блок не выполнится
```

IT CAREER HUB

Задания для закрепления 3

1. Какой результат будет выведен при выполнении следующего кода?

```python
for letter in "Python":

    if letter == "h":

        break

    print(letter, end=' ')
```

a. P y t
b. P y t h
c. P y t o n
d. Ошибка

Посмотреть ответ

2. Какой результат будет выведен при выполнении следующего кода?

```python
for letter in "Python":

    if letter == "h":

        continue

    print(letter, end=' ')
```

a. P y t h o n
b. P y t o n
c. P y h o n
d. Ошибка

Посмотреть ответ

3. Какой результат будет выведен при выполнении следующего кода?

```python
for letter in "Python":
    if letter == "a":
        break
    print(letter, end=' ')
else:
    print("Цикл завершён нормально.")
```

a. Вывод: P y t h o n
b. Вывод: P y t h o n Цикл завершён нормально.
c. Вывод: Цикл завершён нормально.
d. Ошибка

Посмотреть ответ

IT CAREER HUB

Вложенные циклы

Вложенные циклы — это конструкции, в которых один цикл находится внутри другого. Вложенный цикл выполняется полностью для каждого прохода внешнего цикла.

Такие конструкции позволяют обрабатывать сложные структуры данных, например, многомерные массивы или создавать таблицы.

Синтаксис вложенных циклов:

```python
for внешняя_переменная in внешняя_последовательность:

    # код, выполняемый внутри внешнего цикла

    for внутренняя_переменная in внутренняя_последовательность:

        # код, выполняемый внутри обоих циклов
```

- Внешний цикл выполняет итерации по своей последовательности.
- Для каждой итерации внешнего цикла, внутренний цикл проходит через все свои элементы.
- Когда внутренний цикл заканчивает выполнение всех итераций, внешний цикл переходит к следующей итерации.
- При необходимости, можно использовать переменную внешнего цикла внутри внутреннего.

Пример вложенного цикла

```python
for i in "AB":

    for j in "12":

        print(i, j)
```

IT CAREER HUB

Пример с выводом времени

Python

```python
for hour in range(24):

    for minute in range(60):

        print("Время (часов:минут): ", hour, ':', minute, sep='')
```

IT CAREER HUB

Вложенные циклы с использованием while и for

Вложенные циклы — это не только комбинация двух for циклов, но также можно использовать комбинации for и while циклов для решения различных задач.

Пример: вывод времени за три часа, но только до конца дня

```python
current_hours = int(input("Введите текущий час: "))  # Текущее время

hours = 3

end_time = current_hours + 3



while current_hours < 24 and current_hours < end_time:   # Внешний цикл с использованием while

    for minutes in range(60):  # Внутренний цикл с использованием for

        print("Время (часов:минут): ", current_hours, ':', minutes, sep='')

    current_hours += 1  # Увеличение значения часов на 1
```

IT CAREER HUB

Задания для закрепления 4

Какой результат будет выведен при выполнении следующего кода?

```python
for i in range(3):
    for j in range(3):
        print(i + j, end=" ")
```

a. 1 2 3 2 3 4 3 4 5
b. 1 2 3 3 4 5 5 6 7
c. 0 1 2 1 2 3 2 3 4
d. Ошибка

Посмотреть ответ

IT CAREER HUB

Ответы на задания

Задания на закрепление 1	Вернуться к заданиям
1. Результат выполнения кода	Ответ: b
Задания на закрепление 2	Вернуться к заданиям
1. Результат выполнения кода	Ответ: b
2. Результат выполнения кода	Ответ: b
3. Результат выполнения кода	Ответ: b
Задания на закрепление 3	Вернуться к заданиям
1. Результат выполнения кода	Ответ: a
2. Результат выполнения кода	Ответ: b
3. Результат выполнения кода	Ответ: b
Задания на закрепление 4	Вернуться к заданиям
Результат выполнения кода	Ответ: c

IT CAREER HUB

IT CAREER HUB

Практические задания

1. Факториал

Напишите программу, которая находит факториал числа, введённого пользователем и выводит его на экран. Не используйте модуль math для решения.

Факториал числа — это произведение всех натуральных чисел от 1 до самого этого числа включительно.

5! = 1 × 2 × 3 × 4 × 5 = 120

Пример вывода:

```python
Введите число: 5
Факториал числа 5 равен 120
```

Решение:

```python
num = int(input("Введите число: "))
factorial = 1

for i in range(1, num + 1):
    factorial *= i

print("Факториал числа", num, "равен", factorial)
```

IT CAREER HUB

2. Звездный прямоугольник

Напишите программу, которая рисует прямоугольник из символов *, где ширина и высота вводятся пользователем. Используйте вложенные циклы для решения задачи.

Пример вывода:

```
Python
Введите ширину: 5
Введите высоту: 3
*****
*****
*****
```

IT CAREER HUB

Решение:

```python
width = int(input("Введите ширину: "))
height = int(input("Введите высоту: "))

for i in range(height):
    for j in range(width):
        print("*", end="")
    print()  # Переход на новую строку
```

3. Простое число

Напишите программу, которая проверяет, является ли введённое пользователем число простым. Простое число — это число, которое делится только на себя и на 1.

Пример вывода:

```
Python
Введите число: 11
Является простым
-------------------------------------
Введите число: 12
Не является простым
```

Решение (базовое):

```python
num = int(input("Введите число: "))

if num > 1:
    for i in range(2, num):
        if num % i == 0:
            print("Не является простым")
            break
    else:
        print("Является простым")
else:
    print("Не является простым")
```

Решение (оптимизированное):

```python
num = int(input("Введите число: "))

if num > 1:
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            print("Не является простым")
            break
    else:
        print("Является простым")
else:
    print("Не является простым")
```