Модуль 2. Базовый
Урок 1. Циклы

Циклы for, while языке Python. Варианты применения.

Циклы – одни из основных концепций языка Питон. Они позволяют выполнить одно и то же действие несколько раз подряд. Например, с помощью циклов часто выводят записи с базы данных, а также они крайне полезны во время работы с массивами данных. В Python циклы выглядят несколько иначе от своих аналогов в прочих языках.

Цикл while

Первый цикл, который мы рассмотрим, это цикл **while**. Он имеет следующее формальное определение:

```
while условное_выражение:
        инструкции
```

После ключевого слова while указывается условное выражение, и пока это выражение возвращает значение True, будет выполняться блок инструкций, который идет далее.

Все инструкции, которые относятся к циклу while, располагаются на последующих строках и должны иметь отступ от начала строки.

```
choice = "y"
while choice.lower() == "y":
   print("Привет")
     choice = input("Для продолжения нажмите Y, а для выхода любую другую клавишу: ")
print("Работа программы завершена")
```

В данном случае цикл while будет продолжаться, пока переменная choice содержит латинскую букву "Y" или "y".

Сам блок цикла состоит из двух инструкций. Сначала выводится сообщение "Привет", а потом вводится новое значение для переменной choice. И если пользователь нажмет какую-то другую клавишу, отличную от Y, произойдет выход из цикла, так как условие choice.lower() == "y" вернет значение False. Каждый такой проход цикла называется итерацией.

Также обратите внимание, что последняя инструкция print("Работа программы завешена") не имеет отступов от начала строки, поэтому она не входит в цикл while.

Всегда проверяйте, чтобы в коде не было бесконечных циклов. Результатом их работы становится зависание и прекращение работы программы. Если уж и использовать цикл, то нужно устанавливать интервал обновления в несколько секунд. Ещё можно создать бесконечный цикл и прервать его через оператор break.

Другой пример - вычисление факториала:

```
# ! Программа по вычислению факториала
number = int(input("Введите число: "))
i = 1
factorial = 1
while i <= number:
    factorial *= i
    i += 1
print("Факториал числа", number, "равен", factorial)
```

Здесь вводит с консоли некоторое число, и пока число-счетчик i не будет больше введенного числа, будет выполняться цикл, в котором происходит умножения числа factorial.

Консольный вывод:

```
Введите число: 6
Факториал числа 6 равен 720
```

Цикл for

Другой тип циклов представляет конструкция **for**. Цикл for вызывается для каждого числа в некоторой коллекции чисел. Коллекция чисел создается с помощью функции **range()**. Формальное определение цикла for:

```
for int_var in функция_range:
    инструкции
```

После ключевого слова **for** идет переменная int_var, которая хранит целые числа (название переменной может быть любое), затем ключевое слово **in**, вызов функции range() и двоеточие.

А со следующей строки располагается блок инструкций цикла, которые также должны иметь отступы от начала строки.

При выполнении цикла Python последовательно получает все числа из коллекции, которая создается функцией range, и сохраняет эти числа в переменной int_var. При первом проходе цикл получает первое число из коллекции, при втором - второе число и так далее, пока не переберет все числа. Когда все числа в коллекции будут перебраны, цикл завершает свою работу.

Рассмотрим на примере вычисления факториала:

```
#! Программа по вычислению факториала
number = int(input("Введите число: "))
factorial = 1
for i in range(1, number+1):
    factorial *= i
print("Факториал числа", number, "равен", factorial)
```

Вначале вводим с консоли число. В цикле определяем переменную i, в которую сохраняются числа из коллекции, создаваемой функцией range.

Функция range здесь принимает два аргумента - начальное число коллекции (здесь число 1) и число, до которого надо добавлять числа (то есть number +1).

Допустим, с консоли вводится число 6, то вызов функции range приобретает следующую форму: range(1, 6+1)

Эта функция будет создавать коллекцию, которая будет начинаться с 1 и будет последовательно наполняться целыми числами вплоть до 7. То есть это будет коллекция [1, 2, 3, 4, 5, 6].

При выполнении цикла из этой коллекции последовательно будут передаваться числа в переменную i, а в самом цикле будет происходить умножение переменной i на переменную factorial. В итоге мы получим факториал числа.

Консольный вывод программы:

```
Введите число: 6
Факториал числа 6 равен 720
```

Функция range

Функция range имеет следующие формы:

- **range(stop)**: возвращает все целые числа от 0 до stop
- **range(start, stop)**: возвращает все целые числа в промежутке от start (включая) до stop (не включая). Выше в программе факториала использована именно эта форма.
- **range(start, stop, step)**: возвращает целые числа в промежутке от start (включая) до stop (не включая), которые увеличиваются на значение step

Примеры вызовов функции range:

```
1       range(5)          # 0, 1, 2, 3, 4
2       range(1, 5)       # 1, 2, 3, 4
3       range(2, 10, 2)   # 2, 4, 6, 8
4       range(5, 0, -1)   # 5, 4, 3, 2, 1
```

Например, выведем последовательно все числа от 0 до 4:

```
1       for i in range(5):
2           print(i, end=" ")
```

Вложенные циклы

Одни циклы внутри себя могут содержать другие циклы. Рассмотрим на примере вывода таблицы умножения:

```
for i in range(1, 10):
    for j in range(1, 10):
        print(i * j, end="\t")
    print("\n")
```

Внешний цикл for i in range(1, 10) срабатывает 9 раз, так как в коллекции, возвращаемой функцией range, 9 чисел. Внутренний цикл for j in range(1, 10) срабатывает 9 раз для одной итерации внешнего цикла, и соответственно 81 раз для всех итераций внешнего цикла.

В каждой итерации внутреннего цикла на консоль будет выводится произведение чисел i и j. В итоге мы получим следующий консольный вывод:

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36

5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

Выход из цикла. break и continue

Для управления циклом мы можем использовать специальные операторы **break** и **continue**. Оператор break осуществляет выход из цикла. А оператор continue выполняет переход к следующей итерации цикла.

Оператор break может использоваться, если в цикле образуются условия, которые несовместимы с его дальнейшим выполнением. Рассмотрим следующий пример:

```
#! Программа Обменный пункт
print("Для выхода нажмите Y")
while True:
    data = input("Введите сумму для обмена: ")
    if data.lower() == "y":
        break  # выход из цикла
    money = int(data)
    cache = round(money / 56, 2)
    print("К выдаче", cache, "долларов")
print("Работа обменного пункта завершена")
```

Здесь мы имеем дело с бесконечным циклом, так как условие **while True** всегда истинно и всегда будет выполняться. Это популярный прием для создания программ, которые должны выполняться неопределенно долго.

В самом цикле получаем ввод с консоли. Мы предполагаем, что пользователь будет вводить число - условную сумму денег для обмена. Если пользователь вводит букву "Y" или "y", то с помощью оператора break выходим из цикла и прекращаем работу программы. Иначе делим введенную сумму на обменный курс, с помощью функции round округляем результат и выводим его на консоль. И так до бесконечности, пока пользователь не захочет выйти из программы, нажав на клавишу Y.

Консольный вывод программы:

```
Для выхода нажмите Y
Введите сумму для обмена: 20000
```

```
К выдаче 357.14 долларов
Введите сумму для обмена: Y
Работа обменного пункта завершена
```

Но что, если пользователь введет отрицательное число? В этом случае программа также выдаст отрицательный результат, что не является корректным поведением. И в этом случае перед вычислением мы можем проверить значение, меньше ли оно нуля, и если меньше, с помощью оператора continue выполнить переход к следующей итерации цикла без его завершения:

```
#! Программа Обменный пункт
print("Для выхода нажмите Y")
while True:
    data = input("Введите сумму для обмена: ")
    if data.lower() == "y":
        break  # выход из цикла
    money = int(data)
    if money < 0:
        print("Сумма должна быть положительной!")
        continue
    cache = round(money / 56, 2)
    print("К выдаче", cache, "долларов")
print("Работа обменного пункта завершена")
```

Также обращаю внимание, что для определения, относится ли инструкция к блоку while или к вложенной конструкции if, опять же используются отступы.

И в этом случае мы уже не сможем получить результат для отрицательной суммы:

```
Для выхода нажмите Y
Введите сумму для обмена: -20000
Сумма должна быть положительной!
Введите сумму для обмена: 20000
К выдаче 357.14 долларов
Введите сумму для обмена: y
Работа обменного пункта завершена
```

Задание:

Вывести на экран все числа в диапазоне от 1 до 497, которые делятся на 23 без остатка.