Операторы циклов break, continue и else

Мы продолжаем курс по языку Python. Это занятие начнем со знакомства операторов break и continue цикла while. Суть этих операторов, в следующем:

- break – досрочное завершение цикла; continue –
- пропуск одной итерации цикла.

Давайте сначала рассмотрим работу оператора break. Если в качестве условия цикла записать значение True:

```
print("запуск цикла")

i = 0
while True:
    i += 1

print("завершение цикла")
```

То такой цикл будет работать «вечно», он сам никогда не остановится и строчку «завершение цикла» мы не увидим. Программу в этом случае можно только прервать и полностью остановить. Однако, если записать в тело цикла оператор **break,** то он прервется, как только встретится:

```
while True:
    i += 1
    break
```

Обратите внимание, оператор i += 1 один раз будет выполнен, а прерывание сработает только в строчке с оператором break.

Но вы скажете, конечно, зачем писать программы с «вечными» циклами, нужно корректно прописывать условие и все будет хорошо? Часто, именно так и следует поступать. Но бывают ситуации, когда все же требуется досрочно прерывать работу цикла. Например, у нас имеется список с числами: `d = [1, 5, 3, 6, 0, -4]`

И мы хотим определить, есть ли среди них хотя бы одно четное значение. Здесь удобно воспользоваться оператором break. Как только встретим первое четное значение, дальнейшую проверку можно не проводить:

```
d = [1, 5, 3, 6, 0, -4]
flFind = False
i = 0

while i < len(d):
    flFind = d[i] % 2 == 0
    if flFind:
        break

    i += 1

print(flFind)
```

Я, думаю, что принцип работы оператора break понятен. Он прерывает работу любого оператора цикла, как только выполнение программы перейдет к нему.

Следующий оператор continue позволяет пропускать одну итерацию тела цикла. Давайте, опять же, на конкретном примере посмотрим, где и как его целесообразно применять.

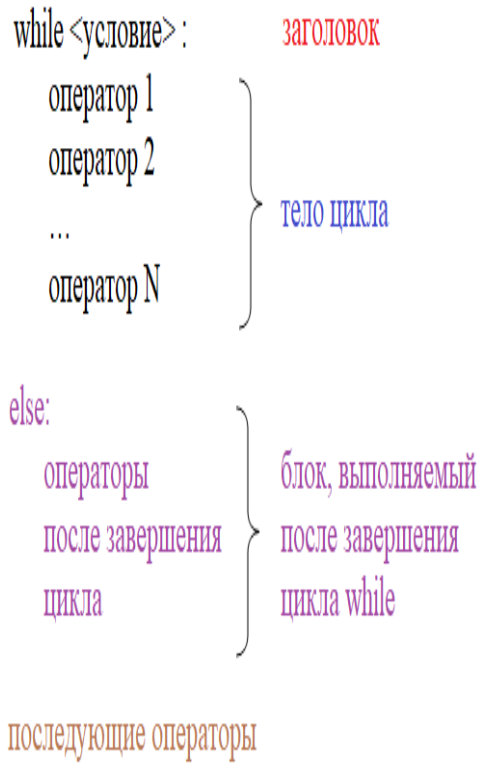
Предположим, что мы просим пользователя вводить с клавиатуры числа и все нечетные значения суммируем. Как только пользователь введет 0, подсчет суммы прекращается. Реализовать эту программу удобно, следующим образом:

```
s = 0
d = 1
while d != 0:
    d = int(input("Введите целое
    if d % 2 == 0:
        continue

    s += d
    print("s = " + str(s))
```

Смотрите, мы здесь на каждой итерации цикла проверяем, если текущее значение d четное, то последующий подсчет суммы и вывод ее в консоль не выполняется, пропускается. При этом, цикл продолжает свою работу и переходит к следующей итерации, пока пользователь не введет число 0. Благодаря оператору continue мы основную логику программы прописываем непосредственно в теле цикла, а не во вложенном блоке условного оператора. То есть, если программу переписать без continue, то она бы выглядела так:

```
s = 0
d = 1

while d != 0:
    d = int(input("Введите целое
    if d % 2 != 0:
        s += d
        print("s = " + str(s))
```

Здесь получается массивный вложенный блок операторов. Конечно, на производительность это никак не влияет, но, все же такие вложения лучше избегать, так программа становится менее читабельной и удобной для дальнейшего редактирования.



В заключение этого занятия рассмотрим третий оператор циклов else. Да, мы уже говорили об этом операторе, когда рассматривали условия. У циклов он тоже есть и сейчас разберем, как работает.

Синтаксис этого оператора, следующий. После тела цикла прописывается необязательный оператор else, в котором указываются операторы, исполняющиеся после завершения цикла:

И здесь возникает вопрос: а чем блок else отличается от блока операторов, просто идущих после блока while? Ведь когда цикл while завершается, мы так и так переходим к последующим операторам!Но обратите внимание вот на эту на фразу «штатное завершение цикла». Штатное завершение – это когда условие цикла стало равно False и оператор while прекращает свою работу. Только в этом случае мы перейдем в блок else. Давайте я покажу это на конкретном примере. Предположим, что мы вычисляем сумму вида:

$$S = 1/2 + 1/3 + 1/4 + 1/10 + ... + 1/0$$

И если в этом сумме встречается деление на ноль, то вычисления следует прервать. Реализуем эту программу, следующим образом:

```
S=0
i=-10

while i < 100:
    if i == 0:
        break

    S += 1/i
    i += 1
else:
    print("Сумма вычислена коррек

print(S)
```

Смотрите, если здесь при вычислении суммы встречается деление на 0, то срабатывает break и цикл досрочно прерывается, то есть, завершается в нештатном режиме. В этом случае блок else пропускается и мы не видим сообщения, что сумма вычислена корректно. Если же все проходит штатно и цикл завершается по своему условию, то в консоли появляется сообщение «Сумма вычислена корректно», означающее выполнение блока else.