Урок №6. True и False, break и continue

План урока

1. Повторение
2. Тип bool
3. Флаги
4. Оператор break
5. Оператор continue

Аннотация

Этот урок посвящён условиям выхода из циклов. Рассматривается булев тип, даются задачи на использование флагов. Затем рассматриваются операторы `break` и `continue`, позволяющие в некоторых случаях избавиться от флагов.

Повторение

Задача "FizzBuzz (https://lms.yandexlyceum.ru/task/view/909)".

Если `a` и `b` — числа (допустим, действительные), то у выражения `a+b` есть какое-то значение (зависящее от значений `a` и `b`) и тип — тоже действительное число. А можно ли сказать, что какое-то значение и тип есть у выражения `a==b`, или это просто конструкция, которая всегда должна стоять в условии `if` или `while`? Да, такое выражение имеет тип под названием `bool` и значение: `True` (**истина**) или `False` (**ложь**). По-русски `bool` — это **булев тип** или булево значение (в честь математика Джона Буля), иногда говорят ещё «**логический тип**»).

```
if True:
    print('Эта строка будет выведена на экран.')
else:
    print('Эта строка никогда не будет выведена на экран.')
print(2 * 2 == 4)  # выведет True
a = input()
b = input()
# Теперь переменная equal равна True, если строки a и b равны, и False в противном случ
ае
equal = (a == b)
if equal and len(a) < 6:
    print('Вы ввели два коротких одинаковых слова.')
```

Обычно переменные с булевым значением (их ещё называют флагами) применяют так: изначально флаг устанавливается в `False`, потом программа как-то работает, а при наступлении определённого события флаг устанавливается в `True`. После выполнения этой части программы идёт проверка, «поднят» ли флаг. В зависимости от её результата выполняется то или иное действие. Иными словами, флаг — это переменная с булевым значением, которая показывает, наступило ли некое событие.

В примере ниже (это программа — терапевтический тренажёр для избавления физиков-экспериментаторов от синхрофазотронозависимости) имеется флаг `said_forbidden_word`, который означает «сказал ли пользователь запретное слово "синхрофазотрон"». Флаг равен `True`, если сказал, и `False`, если нет. В самом начале пользователь ещё ничего не успел сказать, поэтому флаг установлен в `False`. Далее на каждой итерации цикла, если пользователь сказал запретное слово, то флаг устанавливается в `True` и остаётся в таком состоянии (при необходимости флаг можно и «опустить»). Как только флаг оказывается равен `True`, поведение программы меняется: перед каждым вводом выдаётся предупреждение, а в конце выдаётся другое сообщение. Переменным-флагам особенно важно давать осмысленные имена (обычно — утверждения вроде `said_forbidden_word`, `found_value`, `mission_accomplished`, `mission_failed`), ведь флагов в программе бывает много.

```
forbidden_word = 'синхрофазотрон'
# можно было использовать и sep='', чтобы кавычки не отклеились от слова
print('Введите десять слов, но постарайтесь случайно не ввести слово "' + forbidden_wor
d + '"!')
said_forbidden_word = False
for i in range(10):
    if said_forbidden_word:
        print('Напоминаем, будьте осторожнее, не введите снова слово "' + forbidden_wor
d + '"!')
    word = input()
    if word == forbidden_word:
        said_forbidden_word = True
    # вместо предыдущих двух строк также можно написать:
    # said_forbidden_word = (said_forbidden_word or word == forbidden_word)
if said_forbidden_word:
    print('Вы нарушили инструкции.')
else:
    print('Спасибо, что ни разу не упомянули', forbidden_word)
```

Задачи:

- **«Найди кота» (https://lms.yandexlyceum.ru/task/view/910),**
- **«Найди кота — 2» (https://lms.yandexlyceum.ru/task/view/911).**

Необязательные задачи:

- **«Найди кота — 3» (https://lms.yandexlyceum.ru/task/view/912),**
- **«Найди кота — 4» (https://lms.yandexlyceum.ru/task/view/913).**

Если нужно прекратить работу цикла, как только случится некое событие, то кроме флага есть и другой способ — оператор разрыва цикла `break` (он работает и для цикла `for`). Это не функция и не заголовок блока, а оператор, который состоит из одного слова. Он немедленно прерывает выполнение цикла `for` или `while`.

```
for i in range(10):
    print('Итерация номер', i, 'начинается...')
    if i == 3:
        print('Ха! Внезапный выход из цикла!')
        break
    print('Итерация номер', i, 'успешно завершена.')
print('Цикл завершён.')
```

В частности, нередко встречается такая конструкция: цикл, выход из которого происходит не по записанному в заголовке цикла условию (это условие делается всегда истинным — как правило, просто True), а по оператору break, который уже заключён в какой-то условный оператор:

```
while True:
    word = input()
    if word == 'стоп':
        break
    print('Вы ввели:', word)
print('Конец.')
```

Впрочем, злоупотреблять этой конструкцией и вообще оператором `break` не стоит. Когда программист читает ваш код, он обычно предполагает, что после окончания цикла `while` условие в заголовке этого цикла ложно. Если же из цикла можно выйти по команде `break`, то это уже не так. Логика кода становится менее ясной.

Оператор continue немедленно завершает текущую итерацию цикла и переходит к следующей.

```
for i in range(10):
    print('Итерация номер', i, 'начинается...')
    if i == 3:
        print('...но её окончание таинственно пропадает.')
        continue
    print('Итерация номер', i, 'успешно завершена.')
print('Цикл завершён.')
```

Если оператор `break` или `continue` расположен внутри вложенного цикла, то он действует именно на вложенный цикл, а не на внешний. Нельзя выскочить из вложенного цикла сразу на самый верхний уровень.