Сложные условия. Вложенные структуры

Аннотация

На этом уроке мы узнаем, как можно сразу проверять выполнение нескольких условий, научимся объединять условия с помощью логических операций, а также познакомимся с множественным ветвлением. Научимся проверять вхождение подстроки в строку с помощью команды in.

1.Сложное условие. Логические операции

На прошлом уроке мы написали программы, в которых действие выполнялось только при выполнении определенного условия. А что делать, если условий несколько?

Иногда в условном операторе нужно задать сложное условие. Для этого можно использовать логические операции **and** («и»), **or** («или») и **not** («не»).

Чтобы задать одновременное выполнение двух условий, используем `and` («и»), если достаточно выполнения одного из двух вариантов (или обоих сразу) — используем `or` («или»), а если нужно убрать какой-то вариант — `not` («не»).

Приоритет выполнения операций:

```
1. not
2. and
3. or
```

Если нужно изменить приоритет операций или вы забыли правила, используйте скобки.

Например, вот так можно проверить, что оба условия выполнены:

```
print('Как называются первая и последняя буквы греческого алфавита?')
greek_letter_1 = input()
greek_letter_2 = input()
```

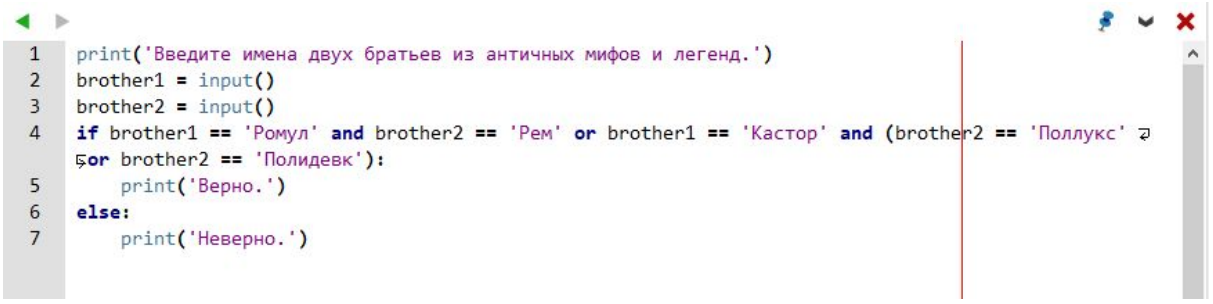
```
if greek_letter_1 == 'альфа' and greek_letter_2 == 'омега':
    print('Верно.')
else:
    print('Неверно.')
```

Ниже еще несколько примеров.

```
print('Как греки или римляне называли
главу своего пантеона - бога грома?')
ancient_god = input()
if ancient_god == 'Зевс' or ancient_god == 'Юпитер':
    print('Верно.')
else:
    print('Неверно.')
```

```
print('Введите имена двух братьев из
античных мифов и легенд.')
brother1 = input()
brother2 = input()
if brother1 == 'Ромул' and brother2 == 'Рем' or brother1 ==
'Кастор' and (brother2 == 'Поллукс' or brother2 ==
'Полидевк'):
    print('Верно.')
else:
    print('Неверно.')
```

Обратите внимание: если программу из предыдущего примера вставить в IDE Wing, часть кода условного оператора будет выходить за ограничительную красную черту среды.



PEP 8

По стандарту PEP 8 длина строки должна быть ограничена максимум 79 символами.

Есть несколько способов переноса длинных строк.

-Использование подразумеваемых продолжений Python внутри круглых, квадратных и фигурных скобок: длинные строки могут быть

разбиты на несколько строк, заключенных в скобки.

-Использование символа ’\’ (обратный слеш, или бэкслеш) для обозначения места разрыва строки.

Мы будем использовать второй способ.

Если после перенесенной строки идет один или несколько вложенных операторов (например, мы переносим строку с условием в операторе `if`), отступ у перенесенной части должен быть на четыре пробела больше, чем у вложенного оператора.

Сделайте правильные отступы для перенесенной строки. Предпочтительнее вставить перенос строки после логического оператора, но не перед ним.

Тогда представленный выше программный код может быть записан так:

```
print('Введите имена двух братьев из
античных мифов и легенд.')
brother1 = input()
brother2 = input()
if brother1 == 'Ромул' and brother2 == 'Рем' or brother1 ==
'Кастор' and\
        (brother2 == 'Поллукс' or brother2 ==
'Полидевк'):
    print('Верно.')
else:
    print('Неверно.')
```

Рассмотрим еще несколько примеров.

```
print('Введите любые два слова, но это не
должны быть "белый" и "медведь" разом.')
word1 = input()
word2 = input()
if not (word1 == 'белый' and word2 == 'медведь'):
    print('Верно.')
else:
    print('Неверно.')
```

А теперь попробуйте решить задачи: «Елочка, гори», «Елочка-2», «Елочка-3».

2.Вложенные условия

Блок кода

В команде `if` при выполнении условия можно выполнять более одной команды. Для этого все их необходимо выделить отступом. Такая запись называется **блоком кода**. По отступам интерпретатор определяет, при выполнении каких условий какие команды исполнять. Аналогично можно делать и для команды `else`.

```
print('Представься, о незнакомец!')
name = input()
if name == 'Цезарь' or name == 'Caesar':
    print('Аве, Цезарь!')
    print('Слава императору!')
else:
    print('Приветик.')
    print('Погода сегодня хорошая.')
print('Засим - заканчиваем.')
```

Перед последней строчкой нет отступа, это означает, что она будет выполнена в конце работы программы в любом случае. А вот две предыдущие строчки будут выполнены, только если условие `if` окажется ложным.

Блоки кода в Python очень гибко устроены: внутри них можно писать любой другой код, в том числе условные операторы. Среди команд, которые выполняются, если условие `if` истинно («внутри if») или ложно («внутри else»), могут быть и другие условные операторы. Тогда команды, которые выполняются внутри этого внутреннего `if` или `else`, записываются с дополнительным отступом.

Изучите пример ниже. `elif` — это короткая запись для «`else: if`». Если не пользоваться короткой записью, if пришлось бы писать на отдельной строчке и с отступом (а все, что внутри этого `if`, — с дополнительным отступом). Это не очень удобно, и `elif` избавляет от такой необходимости.

```
print('Представься, о незнакомец!')
name = input()
if name == 'Цезарь' or name == 'Caesar':
    print('Аве, Цезарь!')
    print('В честь какого бога устроим
сегодня празднество?')
    god = input()
    if god == 'Юпитер':
        print('Ура Громовержцу!')
    # если оказалось, что имя бога не
'Юпитер', то проверяем,
    # не равно ли оно строке 'Минерва'
```

```
    elif god == 'Минерва':
        print('Ура мудрой воительнице!')
    # следующая строка будет выполнена,
    # только если имя бога не 'Юпитер' и не
'Минерва'
    else:
        print('Бога по имени', god, 'мы не знаем,
но слово Цезаря - закон.')
    # эта команда будет выполнена
независимо от того,
    # какое имя бога ввёл пользователь,
если только изначально
    # он представился Цезарем
    print('Слава императору!')
else:
    print('Приветик.')
    print('Погода сегодня хорошая.')
print('Засим - заканчиваем.')
```

А более простой вариант этой программы теперь попробуйте написать самостоятельно.

3.Команда in

Теперь рассмотрим новую команду для работы со строками — команду `in`.

Команда in

Команда `in` позволяет проверить, что одна строка находится внутри другой.

Например: строка «на» находится внутри строки «сложная задача».

В таком случае обычно говорят, что одна строка является **подстрокой** для другой.

```
text = input()
if 'хорош' in text and 'плох' not in text:
    print('Текст имеет положительную
эмоциональную окраску.')
elif 'плох' in text and 'хорош' not in text:
    print('Текст имеет отрицательную
эмоциональную окраску.')
else:
    print('Текст имеет нейтральную или
смешанную эмоциональную окраску.')
```

Первое условие окажется истинным, например, для строк «все хорошо» и

«какой хороший день», но не для «ВсЕ ХоРоШо» и не для «что-то хорошо, а что-то и плохо». Аналогично второе условие окажется истинным для строк «все плохо», «плохое настроение» и т. д.