
Яндекс

Яндекс

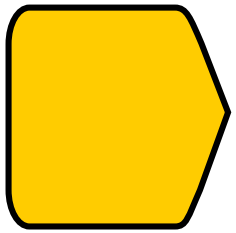
Сложные условия. Вложенные структуры

Сложное условие

Сложное условие

Как называются эти логические операции?

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



Чтобы задать, что два условия должны выполняться одновременно – используем `and` («и»), если достаточно выполнения одного из двух вариантов (или оба сразу), то используем `or` («или»), а если нужно убрать какой-то вариант, то используем `not` («не»).

Приоритет выполнения операций:

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

Пример

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.')
```

Ещё пример

```
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.')
```

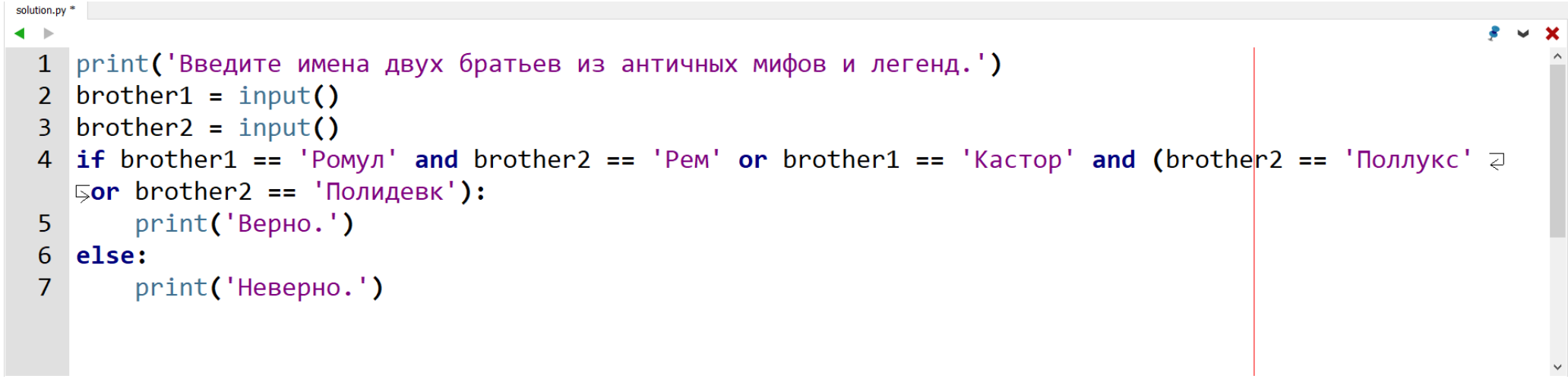
Ещё пример

```
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.')
```

Выпишите все возможные варианты, при которых ответом будет «Верно.»

PEP8

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



PEP8

solution.py *

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

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, обёрнутых в скобки;
- использование символа '\' (обратный слэш, или бекслэш) для обозначения места разрыва строки.

Мы будем использовать первый способ.

PEP8

```
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.')
```

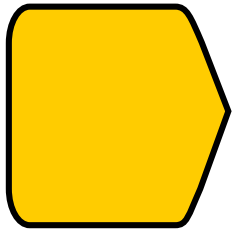
Если после перенесённой строки идёт один или несколько вложенных операторов (например, мы переносим строку с условием в операторе **if**), то отступ у перенесённой части должен быть на 4 пробела больше, чем у вложенного оператора. Сделайте правильные отступы для перенесённой строки. Предпочтительнее вставить перенос строки после логического оператора, а не перед ним.

Ещё пример

```
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.')
```

Выполнение нескольких команд. Отступы

Несколько команд



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, какие команды исполнять при выполнении каких условий. 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 с `elif`

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.

```
if условие_1:
    действие, если истинно
elif условие_2:
    действие, если истинно
else:
    действие, если ложны все условия
```

elif — это короткая запись для "**else: if**". Если не пользоваться короткой записью, то **if** пришлось бы писать на отдельной строчке и с отступом (а всё, что внутри этого **if** — с дополнительным отступом). Это не очень удобно, и **elif** избавляет от такой необходимости.

Пример

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

Команда 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.

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

```
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.')
```


Яндекс