
Яндекс

Яндекс

Условный оператор. Отступы. Операции над строками

Повторение

Переменные

Переменная имеет имя и значение. Имя переменной может начинаться только с буквы и может включать в себя буквы, цифры и символ подчеркивания. Имя переменной должно отражать ее назначение.

Чтобы задать переменной значение, необходимо после знака равно (оператора присваивания) указать значение переменной.

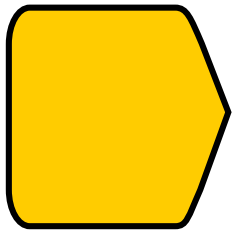
```
name = 'Tom'
```

Еще значение переменной можно получить из ввода. Для этого используем команду **`input()`**. В этом случае значение переменной задает пользователь.

```
name = input()
```

Условный оператор

Условный оператор



Условный оператор используется, когда некая часть программы должна быть выполнена, только **если верно** какое-либо условие. Для записи условного оператора используются ключевые слова **if** и **else** («если», «иначе»), двоеточие, а также **отступ в четыре пробела**.

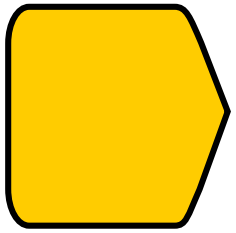
```
if условие:
    Действия, если условие верно
else:
    Действия, если условие неверно
```

Отступ в 4 пробела принят в сообществе **Python (PEP 8).** При этом программа может работать и при других вариантах, но читать ее будет неудобно. **Пробелы — самый предпочтительный метод отступов.** Табуляция должна использоваться только для поддержки кода, написанного с отступами с помощью табуляции. Python 3 запрещает смешивание табуляции и пробелов в отступах.

Пример

```
print('Введите пароль:')
password = input()
if password == 'qwerty':
    print('Доступ открыт.')
else:
    print('Ошибка, доступ закрыт!')
```

Обратите внимание: в начале условного оператора **if** выполняется сравнение, а не присваивание.



Сравнение — это проверка, которая не меняет значение переменной (в сравнении может вообще не быть переменных), а присваивание — команда, которая меняет значение переменной.
Для сравнения нужно использовать двойной знак равенства: **==**.
Также заметьте, что после `else` никогда не пишется никакого условия.

Другой пример

В качестве условия можно использовать и другие операции отношения:

```
print('Представься, о незнакомец!')
name = input()
if name == 'Цезарь':
    print('Аве, Цезарь!')
else:
    print('Приветик.')
```

< меньше

> больше

<= меньше или равно

>= больше или равно

== равно

!= не равно

PEP8

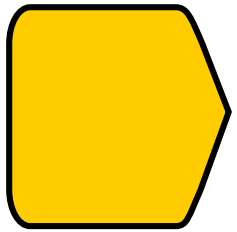
Все операции отношения окружаются пробелами с двух сторон.

Правильно: `if bird == "Тук-тук":`

Неправильно: `if bird=="Тук-тук":`

Операции над строками

Операции над строками



Операция сложения для строк выполняет **конкатенацию** двух строк, то есть склеивает их содержимое вместе.
Например:
Операция `"При" + "вет"` в результате даст строку `"Привет"`

Давайте попробуем:

```
x = '10'
y = '20'
z = x + y
print(z)   # '1020'
```

И опять немного рекомендаций по оформлению (PEP8) — ставьте пробелы вокруг знаков операций (*, +, - и т. д.)

Правильно: `z = x + y`

Неправильно: `z = x+y`

Пример

Обратите внимание, что запись:

x + y = z **недопустима**.

Оператор присваивания ожидает слева переменную, которой надо присвоить значение, а в правой части находится значение или выражение, которое надо сначала вычислить, а затем присвоить.

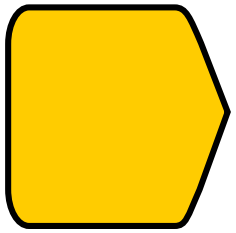
Мы могли сократить нашу программу и написать в таком виде:

```
x = '10'
y = '20'
print(x + y)
```

А еще такой результат можно получить вот таким образом:

```
print('10' + '20')
```

Умножение строки на число



Для строк так же можно выполнять умножение. Умножать можно строку на число или число на строку. Операция называется **дублирование**. В результате нее, начальная строка будет повторена заданное количество раз.

Например: `3 * '20'` то же, что и `'20' * 3` и результат будет `'202020'` и в том, и в другом случае.

Пример использования:

```
x = '10'
y = '20'
print(x * 2 + y * 3)
```

Что будет на экране после запуска такой программы?

Сравнение строк

При сравнении строк принимается во внимание символы и их регистр. Так, цифровой символ условно меньше, чем любой алфавитный символ. Алфавитный символ в верхнем регистре условно меньше, чем алфавитные символы в нижнем регистре. Например:

```
str1 = "1a"
str2 = "aa"
str3 = "Aa"
print(str1 > str2)   # False, так как первый символ в str1 - цифра
print(str2 > str3)   # True, так как первый символ в str2 - в нижнем регистре
```

Вначале сравнение идет по первому символу. Если начальные символы обеих строк представляют цифры, то меньшей считается меньшая цифра, например, "1a" меньше, чем "2a".

Сравнение строк

Символ в верхнем регистре меньше, чем такой же в нижнем:

```
str1 = "Tom"
str2 = "tom"
print(str1 == str2)   # False - строки не равны
```

Если начальные символы представляют алфавитные символы в одном и том же регистре, то проверяют их следование по алфавиту. Так, `"aa"` меньше, чем `"ba"`, а `"ba"` меньше, чем `"ca"`.

Если первые символы одинаковые, в расчет берутся вторые символы при их наличии.

```
str1 = "Лесной"
str2 = "Лесная"
print(str1 > str2)   # True - после одинаковых
# символов в строке 1 стоит буква "о",
# а в строке 2 "а"
```


Яндекс