4. Python. Условный оператор

Автор: Александр Килинкаров, © 2023
E-mail: linxam@gmail.com
Telegram: @linxam
Youtube: https://youtube.com/@it4fun_a
Сайт: kilinkarov.ru

Поток выполнения программы

Когда мы говорим о выполнении программы, то мы должны представить себе, как именно выполняется, написанный нами код. Это и называется *потоком выполнения программы*. Его можно сравнить с потоком реки.

Программа выполняется сверху вниз, сторка за строкой.

Но иногда нам хочется, чтобы программа как бы принимала решение, и в зависимости от условий, поток шёл либо одним путём, либо другим. Чтобы на нашей "дороге" были "перекрестки и развилки".

Тут нам на помощь приходит **условный оператор**

Условный оператор. if

Первый пример

Начинается он словом *if*, за которым написано какое-то *условие*, результатом которого будет *истина* или *ложь*, и в конце строки ставим *двоеточие*.

Например:

```
t = int(input('Какая температура за бортом? '))
```

```
if t < 10:
    print('холодно')

print('Тут мы уже находимся не внутри
if.')
```

Пока что сосредоточимся на том условии, которое мы написали в строке *if*. Я там применил знак сравнения *меньше*. Опишем, какие знаки сравнения есть в python.

```
> больше
< меньше
>= больше, либо равно
<= меньше, либо равно
== равно
!= не равно
```

Итак, вот эти знаки можно писать, чтобы сравнивать что-то с чем-то. При этом, *результатом сравнения будет* **True** или **False**.

Теперь поговорим о **содержимом if**. Обратите внимание на следующую строчку, после if. Она написана с отступом. И она находится *внутри* if'а. Именно благодаря отступу, python понимает, что эта строка - содержимое if'а. Так что, отступ этот очень важен. Обычно, по договоренности, этот отступ равен 4-м пробелам. И отступы в скрипте везде должны быть одной величины. Нельзя комбинировать в одном файле пробелы и знаки табуляции. Поэтому, многие среды разработки автоматически заменяют табуляцию на четыре пробела, чтобы формат отступов был одинаковым.

Содержимое if'а, которое написано с отступом, еще называют **блок кода**. В нашем примере *блок кода* состоит из одной строки, но на самом деле, мы можем включить туда столько строчек кода, сколько нам нужно.

Теперь обратите внимание на последнюю строчку кода, которая выводит сообщение, что мы находимся уже не в if'е. В этой строчке кода

мы убрали отступ вначале строки. Именно по тому, что отступ в четыре пробела ушел, python понимает, что if закончился.

Итак, как работает этот наш if?

Если в строке if возникает **истина (True)**, то **блок кода внутри if выполняется**, мы как бы заходим внутрь if.

Если в условии if возникает **ложь (False)**, то **блок кода внутри if НЕ выполняется**. И мы сразу перескакиваем к той, строке, которая находится после if'а. (которая в нашем примере печатает то, что мы уже не в if'е)

Второй пример

В первом случае, мы, выполняя поток программы, проверяли условие, и если оно истинное, выполняли какой-то блок кода. Если же условие ложное, то мы просто пропускали блок кода внутри if'а, и шли дальше.

А что, если мы хотим организовать развилку? Чтобы если *True*, то выполняем один блок кода. А если *False*, то обязательно выполняем другой блок кода. В общем, хотим "поехать" либо по одной дорожке, либо по другой. Но обязательно "проехать" по одной из них.

Здесь нам поможет использование **else**. (иначе)

В этом случае предложение if, если его сказать на обычном человеческом языке, звучит вот так: **Если** вот*это*истинно *то сделай вот это*, а **иначе** *сделай вот то*.

Давайте запишем какой-нибудь пример на *python*:

```
t = int(input('Какая температура за бортом? '))

if t <= 10:
    print('холодно')
else:
    print('тепло')
```

```
print('конец программы')
```

При данном коде, мы получим один из следующих вариантов работы программы:

```
Какая температура за бортом? 5
холодно
конец программы
```

```
Какая температура за бортом? 23
тепло
конец программы
```

При данном нашем if'е, мы обязательно получим либо один вариант, либо второй.

Первая строка - вне if'а. Она выполняется в любом случае, и спрашивает температуру.

Третья строка тоже вне if'а. И она тоже выполняется в любом случае, и пишет "конец программы".

А вот вторая строка зависит от if. Либо там будет написано *тепло*, либо *холодно*. Но обязательно одно из двух выполнится. Мы либо "едем" по ветке **if**, либо по ветке **else**.

Третий пример

Теперь давайте рассмотрим вариант, когда у нас есть не два варианта, а три и больше. То есть, нам нужно организовать не развилку в коде, а целый перекресток. Тут нам поможет версия условного оператора с **elif**'ами. **Elif** означает **иначе если**.

Итак, нам нужно, чтобы наша программа при разных температурах выводила разные слова (а если пофантазировать, то можно написать погодное приложение или виджет, который отображает разные картинки).

- при температура 0 и меньше - *мороз*
- от 1 до 10 градусов - *холодно*
- от 11 до 20 градусов - *прохладно*

- от 21 до 30 градусов - *жарко*
- от 31 и выше - *ад*

Напишем код:

```
t = int(input('Какая температура за бортом? '))

if t <= 0:
        print('мороз')
elif t <= 10:
        print('холодно')
elif t <= 20:
        print('прохладно')
elif t <= 30:
        print('жарко')
else:
        print('ад')

print('Тут мы уже после if')
```

Здесь мы организовали выбор из 5 возможных вариантов. При выполнении программы, python вначале проверяет условие *if*, если оно не

подходит, переходит к *elif*. При первом же совпадении, выполняется та ветка, условие которой дало *True*, остальные ветки пропускаются.

Итак, подведём итоги. Условный оператор начинается словом **if**. За ним может быть, а может и не быть любое количество **elif**. И в самом конце может быть, а может и не быть **else**. Если **else** используем, то он пишется в самом конце **if**'а, последней веткой.