ПИТОН

Ветвления и циклы

Меджидов Руслан

if в C и в Python

По сравнению с языком C и другими СИ-подобными языками, оператор if в Python менее загромождён, чтобы облегчить жизнь программистов, требуя набора меньшего объема символов.

```
if (x < y)
{
    x = 1;
    y = 2;
}
```

```
if x < y:
    x = 1
    y = 2
```

Особенности синтаксиса

Новым, по сравнению с языком C, компонентом синтаксиса if в Python является символ двоеточия.

Все составные операторы в Python следуют общему шаблону:

строка_заголовка: вложенный_блок_кода

```
if x < y:
    x = 1
    y = 2
```

Круглые скобки, в которые можно заключить условие, необязательны, но и ошибкой не являются.

То, что конец строки является концом оператора, а отступ — заменяет блок, заключенный в фигурные скобки, добавляет компактности коду.

Отступы и нужный if в C

К какому из операторов if относится else (язык C)?

```
if (x)
    if (y)
        оп_
1;
else
    оп_2;
```

Ответ: if(y). Блок else в языке C относится к последнему оператору if.

Неправильный ответ можно дать, благодаря неверным (логически, но не синтаксически) отступам перед else для языка C.

То есть для понятности кода, отступы надо изменить, но язык позволяет и такое написание.

Отступы и нужный if в Python

В языке Python подобная ошибка возникнуть не может, так как блок else относится не к последнему оператору if, а к соответствующему по отступам:

```
if x:
    if y:
        оп_
1
else:
    оп_2
```

В данном случае else относится к первому оператору if x.

Многочисленные проверки

Оператор if языка Python способен заменить оператор switch языка СИ, благодаря одному, или более необязательных блоков elif:

```
switch  (x)
case сост_1:
      оп_1;
case сост_2:
      оп_2;
default:
      оп_3;
```

```
if условие_1:
      оп_1
elif
условие_2:
      оп_2
else:
      оп_3
```

Данная конструкция, отчасти, может быть более функциональной, так как проверки условия оператора if разнообразнее простой проверки на равенство оператора switch-case.

Область видимости Python

В языке Python операторы ветвлений и циклов не создают собственную область видимости, как это происходит в операторах ветвлений, циклов и блоках кода {} в языке C:

```
if (1)
{
      int x = 55;
}
printf("%d", x);
```

error: 'x' undeclared

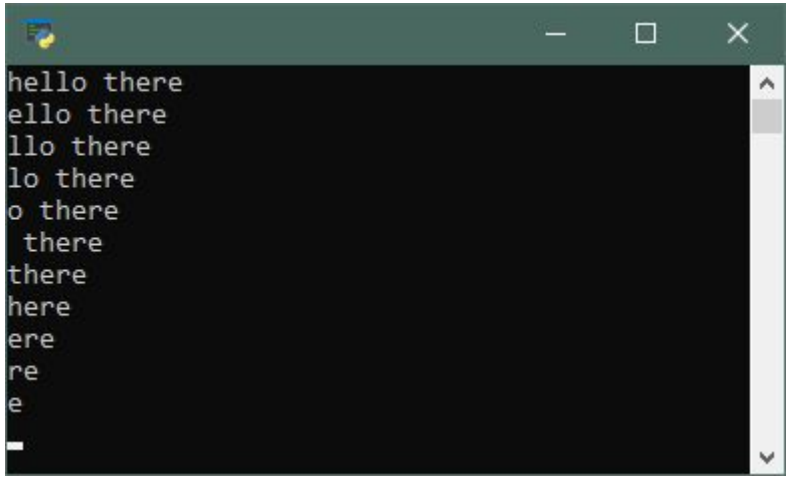
```
if 1:
    x = 55
print(x)
```

55
Ошибки нет.

Оператор while

Оператор while многократно выполняет блок операторов до тех пор, пока проверка в заголовочной части оценивается как истинное значение:

```
x = 'hello there'
while x:
    print(x)
    x = x[1:]
```



Данный пример нуждается в пояснении:

- x = x[1:] — срез укорачивает строку на один первый символ
- while x — любой непустой объект считается истиной, цикл будет выполняться пока строка не опустеет

Общий синтаксис оператора while

Общий синтаксис оператора несколько отличается от языка СИ:

```
while условие:
    операторы
else:
    операторы
```

Цикл может содержать блок else — необязательную часть. Ее операторы выполняются, если выход из цикла произошел «планово» (не с помощью оператора break).

for в языке Python

Оператор for в языке Python, предназначен для прохода по элементам в последовательности или в другом итерируемом объекте и выполнения блока кода для каждого элемента.

То есть, другими словами, for больше не является универсальным циклом
как в языке C, но позволяет совершать действия над каждым элементом строки, списка, файла и прочих объектов.

for в языке C

```
for (инициализация; условия; увеличение)
{
    делаем что угодно;
}
```

for в Python

```
for переменная in объект
    действия
```

Пример работы оператора for

Как использовать оператор for легче понять из примера:

```
for i in 'hello there':
    print(i * 2, end='')
```

В результате выполнения примера будет выведено сообщение:

«hheellllloo tthheerree»

Оператор for языка Python во многом напоминает оператор foreach, таких языков как Java и C#, однако может содержать блок else подобно оператору while в Python.

for i in range(n)

Для повторения цикла некоторое заданное число раз n можно использовать цикл for вместе с функцией range:

```
for i in range(3):
print(i)
```

Функция range может также принимать не один, а два или три параметра. Вызов range(a, b) означает, что индексная переменная будет принимать значения от a до b - 1.

Третий параметр означает величину изменения (шаг) индексной переменной.

continue и break

Данные операторы аналогичны одноименным операторам СИ:

- Оператор break переходит за пределы ближайшего заключающего его цикла, то есть начинают выполняться строки кода после всего оператора цикла,
в котором расположен этот break.
- Оператор continue переходит в начало ближайшего заключающего цикла
на строку заголовка цикла, то есть позволяет игнорировать все тело цикла, находящееся после continue.

```
for i in 'hello world':
    if i == 'a':
        break
else:
    print('Буквы a в строке нет')
```

Задача

Определить число знаков (порядок) целого положительного числа, не используя методы str и len.

```
x = 88005553535   # исходное число
res = 0
while x:      # аналог на C: while (x != 0)
    res += 1  # инкремент (++res для C)
    x //= 10  # краткая форма записи для x = x // 10
print(res)
```

ПИТОН

Ветвления и циклы

Меджидов Руслан