Язык программирования Питон (Python)

Операторы

Для изменения естественного порядка выполнения команд используются ***операторы управления потоком***: **`if`**, **`for`** и **`while`**.

Оператор **`if`**

– для проверки условий: если условие верно, выполняется блок выражений (называемый «**`if-блок`**»), иначе выполняется другой блок выражений (называемый «**`else-блок`**»).

Блоки «**`elif`**» и «**`else`**» являются необязательными.

```
if  условие_1 :
    if-блок
elif  условие_2 :
    else-блок_1
elif  условие_3 :
    else-блок_2
else :
    else-блок_3
```

Пример:

```
number = 23
guess = int(input('Введите целое число : '))

if guess == number:
    print('Поздравляю, вы угадали,')
elif guess < number:
    print('Нет, загаданное число больше этого.')
else:
    print('Нет, загаданное число меньше этого.')
print('Завершено')
```

Пример:

```
if True:
    print('Да, это верно.')
```

Пример:

```
value = input("Введите pH: ")
if len(value) > 0:  # Ввели что-нибудь?
    pH = float (value)
    if pH == 7.0:   # Пров.равенства с действ.числ.!
        print(pH, "Вода")
    elif 7.36 < pH < 7.44:
        print(pH, "Кровь")
    else:
        print("Что это?!")
else:
    print("Введите значение pH!")
```

Оператор цикла `while`

– позволяет многократно выполнять блок команд до тех пор, пока выполняется некоторое условие;

– применяется, если заранее количество повторений цикла неизвестно;

– может иметь необязательный пункт **`else`**;

– если у цикла **`while`** имеется дополнительный блок **`else`**, он всегда выполняется, если только цикл не будет прерван оператором **`break`**.

```
while  условие :
   блок_команд_1
else :
   блок_команд_2
```

Пример:

```
number = 23
running = True

while running:
    guess = int(input('Введите целое число : '))

    if guess == number:
        print('Поздравляю, вы угадали,')
        running = False
    elif guess < number:
        print('Нет, загаданное число больше этого.')
    else:
        print('Нет, загаданное число меньше этого.')

else:
    print('Цикл закончен')

print('Завершено')
```

Оператор цикла `for`

– осуществляет итерацию по последовательности объектов, т.е. проходит через каждый элемент в последовательности

– может иметь необязательный пункт **`else`**.

Пример:

```
for i in range(1, 5):
    print(i)
else:
    print('Цикл for закончен')
```

Замечание. **`range()`** генерирует последовательность чисел, но только по одному числу за раз – когда оператор **`for`** запрашивает следующий элемент.

Пример: *перебор элементов списка*

```
for i in [1, 'два', 3, "четыре", 5.102, 60/9]:
    print(i)
```

Пример:

```
outer = [1, 2, 3, 4]      # внешний цикл
inner = [5, 6, 7, 8]      # вложенный цикл
for i in outer:
    for j in inner:
        print ('i=', i, 'j=', j)
```

Замечание. Циклы можно вкладывать друг в друга.

Пример: *перебор символов в строке*

```
country = "Rusia"
for ch in country:
    if ch.isupper():
        print(ch)
```

Замечание.

строковый метод **`isupper()`** проверяет верхний регистр символа и возвращает **`True`** или **`False`**.

Пример:

```
stroka = "привет"
for bukva in stroka:
    print(bukva, end=' * ')
```

Пример: работа со словарём

```
ab = { 'Swaroop'   : 'swaroop@swaroopch.com',
       'Larry'     : 'larry@wall.org',
       'Matsumoto' : 'matz@ruby-lang.org',
       'Spammer'   : 'spammer@hotmail.com'
     }

for name, address in ab.items():
    print("имя: ", name, "\t эл.адр.: ", address)
```

Пример:

```
d = {1:'one',2:'two',3:'three',4:'four'}
for key in d:
    d[key] = d[key] + '!'
print(d)
```

Оператор `break`

– служит для прерывания цикла – остановки выполнения команд даже если условие выполнения цикла ещё не приняло значения `False` или последовательность элементов не закончилась

– если циклы `for` или `while` прервать оператором `break`, соответствующие им блоки `else` выполняться не будут

Пример:

```
while True:
    s = input('Введите что-нибудь : ')
    if s == 'выход':
        break
    print('Длина строки: ', len(s))
print('Завершение')
```

Оператор `continue`

– используется для указания Python, что необходимо пропустить все оставшиеся команды в текущем блоке цикла и продолжить со следующей итерации цикла

Пример:

```
while True:
    s = input('Введите что-нибудь : ')
    if s == 'выход':
        break
    if len(s) < 3:
        print('Слишком мало')
        continue
    print('Введённая строка достаточной длины')
    print('Длина строки: ', len(s))
```