Дисципліна «Основи програмування на Python»

Циклічні алгоритми. Конструкція while

Медведєв Д.Г.

Зараз ми розглядаємо одну з двох основних циклічних конструкції мови Python – операторів, які повторюють дію знову і знову. Перший з них, оператор `while`, пропонує спосіб написання універсальних циклів. Другий, оператор `for`, призначений для проходу по елементам у послідовності або в іншому ітеруючому об’єкті та виконання коду блоку для кожного елемента.

Оператор `while` – універсальна конструкція для ітерацій мовою Python. Виражаючись простими термінами, він багатократно виконує блок операторів (звичайно з відступом) до тих пор, поки перевірка в заголовній частині оцінюється як істинне значення. Це називається «циклом», тому що управління продовжує повертатися до початку оператора, якщо перевірка не має хибного значення. Коли результат перевірки стає хибним, управління переходить до оператора, наступного після блоку `while`. Тіло циклу виконується багатократно, поки перевірка в заголовній частині дає істинне значення. Якщо перевірка оцінюється в хибному значенні з самого початку, тоді тіло циклу ніколи не виконується, а оператор while пропускається.

У своїй найскладнішій формі оператор `while` складається з рядка заголовка з виразом перевірки, тіла з одним або великою кількістю операторів з відступами та необов'язковою частиною `else`, яка виконується, якщо керування закінчує цикл, а оператор `break` не зустрівся. Python продовжує оцінювати вираз перевірки в рядку заголовка та виконує оператори, вкладені в тіло циклу, якщо перевірка не повертає хибне значення:

```
while перевірка:
    оператори
else:
    оператори
```

Розглянемо декілька простих циклів поки в дії. Перший, який складається з оператора print, вкладеного в цикл `while`, лише наскінченно виводить повідомлення. Така поведінка зазвичай називається нескінченним циклом, хоча він не вічний, – необхідно натиснути комбінацію клавіш `<Ctrl+C>` для завершення його роботи:

```
while True:
    print('Type Ctrl-C to stop me!')
```

У наступному прикладі проводиться відкидання першого символу рядка до тих пір, поки він не стане порожнім і тому хибним.

```
x = 'spam'
while x:
    print(x, end=' ')
```

```
    x = x[1:]
# spam pam am m
```

Наступний фрагмент коду визначає, чи є позитивне ціле число простим, за рахунок пошуку дільників більше 1:

```
x = y // 2
while x > 1:
    if y % x == 0:
        print(y, 'has factor', x)
        break
    x -=1
else:
    print(y, 'is prime')
```

Замість установки прапорця, призначеного для перевірки, чи потрібно виходити з циклу, в коді застосовується оператор `break`, коли множник знайдено. У підсумку можна допустити, що конструкція `else` циклу буде виконана, тільки якщо множники не знайдені; попадання на break означає, що число непросте.

Конструкція `else` циклу також виконується, якщо тіло циклу жодного разу не виконується, тому що в цьому випадку не виконується і оператор `break`; в циклі while відбувається подібне, коли перевірка в заголовку з самого початку дає хибне значення. Таким чином, у попередньому прикладі все рівно буде виведено повідомлення є простим, якщо `x` значно менше або рівно 1 (тобто коли `y` рівно 2).