

DOI: 10.32517/2221-1993-2022-21-4-70-73

О. Ю. Заславская*Российская академия образования, г. Москва, Россия;**Московский городской педагогический университет, г. Москва, Россия***О. Д. Любутов***Университетский лицей № 1511 предвуниверситария НИЯУ МИФИ, г. Москва, Россия*

ОСОБЕННОСТИ МЕТОДИКИ ПРЕПОДАВАНИЯ ЦИКЛА FOR В ЯЗЫКЕ PYTHON В СРЕДНЕЙ ШКОЛЕ

Аннотация

Сегодня во многих учебных заведениях в программу по информатике включено изучение языка программирования Python. Обладая бесспорными достоинствами, которые отмечаются и педагогами-исследователями, и учителями-практиками, язык Python имеет ряд особенностей, на которые следует обратить внимание школьников, чтобы предотвратить ошибки, которые могут быть допущены ими в будущем при переходе на другие языки программирования. Это касается в том числе изучения циклических конструкций, в частности цикла For. Реализация цикла For в языке Python принципиально отличается от реализации цикла с параметром, использующегося в процедурных языках программирования, которые изучались в рамках школьной программы ранее (BASIC, Pascal, C/C++ и т. п.). На сегодняшний день мы имеем ситуацию, когда методика изучения цикла с параметром не может быть применена к циклу For языка Python. Методике изучения этой циклической конструкции посвящена данная статья.

Ключевые слова: теория и методика обучения информатике, языки программирования, Python, цикл с параметром, цикл For.

1. Введение

В настоящее время растет число учебных заведений, в которых в рабочие программы по информатике включено изучение языка программирования Python. Достоинства этого языка неоднократно отмечались

многими авторами (см., например, [1, 3, 4, 8]). Но наряду с бесспорными плюсами язык Python имеет ряд особенностей, на которые следует обратить внимание школьников, чтобы предотвратить ошибки, которые могут быть допущены ими в будущем при переходе на другие языки программирования.

Контактная информация

Заславская Ольга Юрьевна, доктор педагогических наук, профессор;

ведущий аналитик, Лаборатория развития цифровой образовательной среды РАО, г. Москва, Россия; *адрес:* 119121, Россия, г. Москва, ул. Погодинская, д. 8;

профессор департамента информатизации образования, Институт цифрового образования, Московский городской педагогический университет, г. Москва, Россия; *адрес:* 129226, Россия, г. Москва, 2-й Сельскохозяйственный пр-д, д. 4, корп. 1;

e-mail: zaslavskaya@mgpu.ru

Любутов Олег Дмитриевич, учитель информатики, Университетский лицей № 1511 предвуниверситария НИЯУ МИФИ; *адрес:* 115409, Россия, г. Москва, Каширское ш., д. 31; *e-mail:* lod375039@yandex.ru

O. Yu. Zaslavskaya

Russian Academy of Education, Moscow, Russia;

Moscow City University, Moscow, Russia

O. D. Lyubutov

Lyceum 1511 of the Pre-University of the National Research Nuclear University MEPhI, Moscow, Russia

FEATURES OF THE METHOD OF TEACHING THE FOR LOOP IN PYTHON LANGUAGE IN SECONDARY SCHOOL

Abstract

Today, in many educational institutions, the study of the Python programming language is included in the informatics curriculum. Possessing indisputable advantages, which are noted by both research educators and practicing teachers, the Python language has a number of features that students should pay attention to in order to prevent mistakes that they may make in the future when switching to other programming languages. This also applies to the study of cyclic constructions, in particular the For loop. The implementation of the For loop in the Python language is fundamentally different from the implementation of the loop with a parameter used in procedural programming languages that were studied earlier in the school curriculum (BASIC, Pascal, C/C++, etc.). Today, we have a situation where the method of learning a loop with a parameter cannot be applied to the For loop of the Python language. The article is devoted to the methods of studying this cyclic construction.

Keywords: theory and methods of teaching informatics, programming languages, Python, loop with parameter, For loop.

2. Реализация циклических конструкций в различных языках программирования до Python

Создание первых компьютеров (ЭВМ) в середине прошлого века вызвало насущную потребность в программном обеспечении, в первую очередь в языках программирования [2, 5–7, 10]. Программы, написанные в машинных кодах, а затем и на первых языках ассемблера, не имели в своем наборе циклических конструкций. Цикл с предусловием (так же как и цикл с постусловием) реализовывался с помощью оператора условного перехода (*if*) и оператора безусловного перехода (*goto*). Частным случаем цикла с предусловием является цикл с параметром. Изображение цикла в виде блок-схем приведено на рисунке 1.

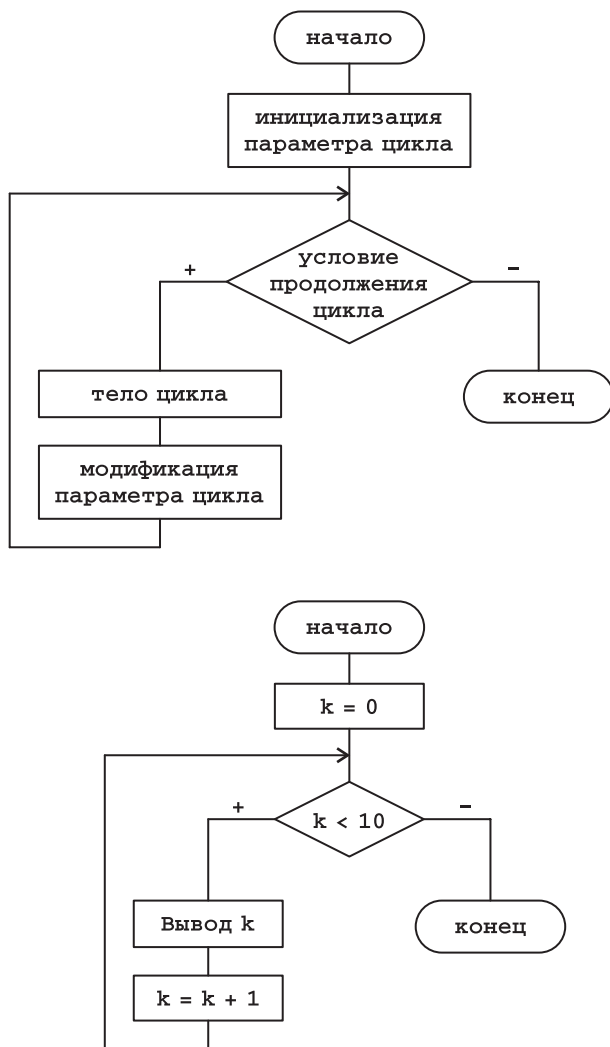


Рис. 1

Появление парадигмы **структурного программирования** практически исключило из употребления оператор безусловного перехода *goto*. Во всех структурных языках программирования появились **циклические конструкции**: цикл с предусловием, цикл с постусловием, цикл с параметром.

Блок-схема **цикла с параметром** изображена на рисунке 2.

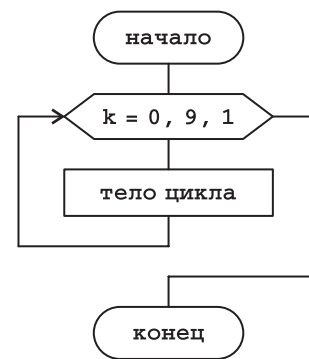


Рис. 2

В заголовке цикла (обозначается шестиугольником) задаются:

- параметр цикла (в примере на рисунке 2 это k);
- начальное значение параметра (0);
- конечное значение параметра (9);
- шаг изменения параметра (1).

Если значение шага не задано, то по умолчанию оно считается равным единице. От текущего значения параметра зависит выход из цикла (завершение цикла) и количество итераций выполнения тела цикла.

Методика изучения цикла с параметром подразумевала разбор ситуации, в которой происходит принудительное изменение параметра цикла в теле цикла.

Рассмотрим следующую программу, написанную на языке программирования C++:

```
#include <iostream>
using namespace std;
int main(){
    int k;
    for(k=0;k<10;k++){
        cout<<k<<endl;
        k=3;
    }
}
```

В результате выполнения этой программы на экране появится ноль, а затем бесконечное количество четверок, потому что выйти из этого цикла программа не сможет (для выхода переменная k должна превысить значение 9, а этого никогда не произойдет).

Об опасности закливания в результате модификации параметра школьники информировались в обязательном порядке. В некоторых учебниках постулируется недопустимость модификации параметра цикла, а некоторые среды программирования (например, PascalABC.NET) вообще выдают ошибку при попытке изменения параметра. В некоторых случаях подобный «запрещенный прием» позволяет сократить код программы, например, когда требуется выйти из нескольких вложенных циклов.

На рисунке 3 приведены две программы:

- а) с «классической» реализацией выхода из циклов с помощью флага и операторов *break*;
- б) использующая «запрещенный» выход, реализованный модификацией параметров циклов.

Обе программы просматривают тройки значений элементов массива *mas[]* до тех пор, пока сумма значений тройки не будет равна нулю. В этом случае программа

```

int main()
{
    int i,j,k,flag;
    int mas[11]={-5,-4,-3,-2,-1,0,1,2,3,4,5};
    flag=0;
    for(i=0;i<9;i++)
        if (flag==0)
            for (j=i+1;j<10;j++)
                if (flag==0){
                    for (k=j+1;k<11;k++){
                        if (mas[i]+mas[j]+mas[k]==0){
                            cout<<i<<j<<k<<endl; flag=1; break;
                        }
                    }
                }
            else break;
        else break;
}

```

а

```

int main(){
    int i,j,k,flag;
    int mas[11]={-5,-4,-3,-2,-1,0,1,2,3,4,5};
    for(i=0;i<9;i++)
        for (j=i+1;j<10;j++)
            for (k=j+1;k<11;k++){
                if (mas[i]+mas[j]+mas[k]==0){
                    cout<<i<<j<<k<<endl; k=10; j=9; i=8;
                }
            }
}

```

б

Рис. 3

должна напечатать индексы (номера) элементов массива, составляющих тройку, и завершить свое выполнение.

Легко заметить, что во второй программе на шесть строк меньше, чем в первой. Но пользоваться этим приемом можно, только хорошо представляя себе работу цикла *for*.

3. Особенности реализации и использования цикла For в Python

Циклическая конструкция *for* в языке программирования Python принципиально отличается от рассмотренного выше цикла с параметром.

Заголовок цикла записывается следующим образом:

```

for <переменная>
    in <строка|список|множество|итератор|range()|>:

```

Переменная, объявленная в заголовке цикла, не является параметром в «классическом» смысле. Изменение ее значения в теле цикла никак не влияет на работу цикла. Это происходит из-за того, что цикл *for* реализует обращение к итерируемому контейнеру, записанному после *in* (это может быть одна из конструкций: строка, список, множество, итератор, встроенная функция *range()*), получает из него очередное значение (либо исключение — в этом случае работа цикла завершается), которое присваивается переменной, после чего выполняется тело цикла.

В свете вышеизложенного возникает **методическая проблема при объяснении работы цикла *for***: для подробного разбора требуется сначала изучить понятия строки, списка, итератора, функции *range()*, т. е. тех элементов языка, которые обычно изучаются гораздо позже цикла с параметром. Для решения этой проблемы обычно применяется **изучение цикла *for* в два захода**.

Первый заход. При первом знакомстве с циклом ученикам просто объясняют, что переменная цикла получает свои значения из функции *range()*, при этом не углубляются в механизм работы данной функции.

Например, для следующего кода:

```

for k in range(0,10,1):
    print(k)

```

следует объяснить, что сначала переменная *k* принимает начальное значение 0, затем выполняется тело цикла (на экран выводится 0). Затем переменная *k* принимает очередное значение, которое ей передает функция *range()* (это ранее выданное значение, увеличенное на шаг), и снова выполняется тело цикла (на экран выводится 1). Этот процесс повторяется до тех пор, пока переменной не будет присвоено значение 9, и после выполнения тела происходит завершение цикла. Данное объяснение не затрагивает механизм генерации значений в функции *range()*.

Второй заход. В процессе дальнейшего изучения языка, после знакомства с итераторами (списками, множествами, словарями), следует вернуться к оператору *for* и разобрать его работу на новом уровне, с учетом изученных элементов языка. Особое внимание следует уделить изучению существующих итераторов (в языке Python имеется модуль *itertools*, в котором реализован большой выбор различных итераторов), а также возможности создания собственных итераторов. Это позволит школьникам всесторонне и подробно изучить работу оператора цикла *for*.

Такая «неклассическая» организация цикла *for* приводит и к **отличиям в его использовании**.

Рассмотрим следующий пример:

```

for k in range(0,10,1):
    print(k,end=" ")
    k=3

```

Эта программа уже была рассмотрена выше, только написана она была на языке C++. Запуск этой программы приводил к ошибке, потому что параметр цикла никогда не достигал конечного значения (оставался равным 3). В программе, написанной на языке Python, на экран будут выведены числа от 0 до 9, после чего цикл штатно завершит свою работу. И вот это принципиальное отличие в реализации цикла создает предпосылку для ошибок, если неопытный программист захочет переписать свою программу с языка Python на другой процедурный язык.

Рассмотрим еще один пример программы:

```

for k in range(1,4,1):
    for k in range(2,5,1):
        print(k,end=" ")

```

На других языках программирования подобная программа сразу вызовет ошибку. Два вложенных цикла не

могут иметь один и тот же параметр (переменную k). Однако в языке Python такая программа работает и выводит на экран следующую строку:

```
2 3 4 2 3 4 2 3 4
```

Подобные возможности языка вызывают соблазн «сэкономить» переменную и закладывают фундамент для плохого стиля программирования.

Приведем еще один пример с задачей, где требуется выводить на экран суммы цифр элементов списка:

```
a=[527,364,88,1002,12,678,555,13,7,0,999]
for k in a:
    s=0
    while k>0:
        s+=k%10
        k//=10
    print(s,end=" ")
```

Программа будет нормально работать и выведет суммы цифр для всех чисел, содержащихся в списке. Если же эту программу переписать на другой язык программирования, то она уйдет в бесконечный цикл, так как в теле цикла обнуляется переменная k .

Привычка не обращать внимания на переменную цикла формирует у школьников стиль программирования, который может в дальнейшем оказать им медвежью услугу при работе с другими языками программирования. Разрешение этой проблемы видится в строгом контроле за подобными ситуациями со стороны учителя и запрете на изменение переменной в теле цикла. Тогда текст предыдущей программы должен будет выглядеть следующим образом:

```
a=[527,364,88,1002,12,678,555,13,7,0,999]
for k in a:
    s=0
    x=k
    while x>0:
        s+=x%10
        x//=10
    print(s,end=" ")
```

Здесь мы видим, как с помощью дополнительной переменной x можно обойтись без изменения переменной цикла.

4. Выводы

Появление новых концепций программирования и возникновение новых языков разработки, реализующих эти концепции, приводит к необходимости

пересмотра методики преподавания [9, 11] даже таких базовых конструкций языка программирования, как цикл с параметром. К сожалению, вместо простых «атомарных» операторов (таких как условный и безусловный операторы) в новых языках используются более сложные конструкции, которые являются комфортными в использовании, но менее очевидными в процессе их изучения. Если бы наряду с итерируемым оператором цикла *for* в языке программирования Python был бы реализован «классический» цикл с параметром, то изучение циклических конструкций несколько упростилось бы.

Список источников

1. Гэддис Т. Начинаем программировать на Python. 4-е изд.: пер. с англ. СПб.: БХВ-Петербург, 2019. 768 с.
2. Довек Ж., Леви Ж.-Ж. Введение в теорию языков программирования. М.: ДМК-Пресс, 2016. 134 с.
3. Заславская О. Ю. Требования к подготовке учителя информатики в условиях реализации деятельностного подхода // Вестник Российского университета дружбы народов. Серия: Информатизация образования. 2010. № 3. С. 21–27. EDN: MUOEQX.
4. Златопольский Д. М. Основы программирования на языке Python. М.: ДМК-Пресс, 2017. 284 с.
5. Кауфман В. Ш. Языки программирования. Концепции и принципы. М.: ДМК-Пресс, 2017. 464 с.
6. Кетков Ю. Л., Кетков А. Ю. Практика программирования: Бейсик, Си, Паскаль. СПб.: БХВ-Петербург, 2011. 480 с.
7. Кетков Ю. Л., Кетков А. Ю. Свободное программное обеспечение. FREE PASCAL для студентов и школьников. СПб.: БХВ-Петербург, 2011. 384 с.
8. Кузнецов А. А., Григорьев С. Г., Гриншкун В. В., Левченко И. В., Заславская О. Ю. Информатика и ИКТ. 8 класс: учебник для общеобразовательных учреждений. М.: Просвещение; Дрофа, 2010. 255 с.
9. Ростовых Д. А., Смольникова И. А., Полянская А. В., Гриншкун В. В., Филатова Н. И., Сайков Б. П., Локотко Е. Г., Ежова Г. Л., Дагестани К. Ф., Шкловец Ю. И., Первин Ю. А., Рапуто А. Г., Мудракова О. А., Чеканова М. С., Острикова Е. Г., Горелов Н. К., Спичекова Т. П., Никишина И. Н., Козлов О. А., Скарга В. А., Сальников В. Д., Довгань В. В., Цифаркина М. А., Соколова И. В., Иванченко Д. А. Подготовка и профессиональная деятельность учителей и преподавателей информатики. Компетентностный подход. М.: Российский государственный социальный университет, 2010. 212 с. EDN: TJTLCS.
10. Страуструп Б. Язык программирования C++. Специальное издание: пер. с англ. Н. Н. Мартынов. М.: БИНОМ, 2020. 1136 с.
11. Zaslavskaya O. Yu., Zaslavsky A. A., Bolnokin V. E., Kravets O. Ya. Features of ensuring information security when using cloud technologies in educational institutions // International Journal on Information Technologies and Security. 2018. Vol. 10. No. 3. P. 93–102. EDN: UXEUVG.