

Лабораторная работа № 3

Python: циклы

Содержание

1 Цель работы.....	2
2 Краткие теоретические сведения.....	2
3 Оборудование рабочего места.....	2
4 Задание.....	2
5 Содержание отчета.....	2
6 Порядок выполнения работы.....	3
6.1 Необходимость использования циклов.....	3
6.2 Цикл for.....	5
6.3 Цикл while.....	13
6.4 Управление циклом: continue и break.....	15
6.5 Самостоятельная работа.....	19
6.6 Дополнительное задание.....	21

1 Цель работы

Изучить порядок составления и использования циклических конструкций на языке программирования Python.

2 Краткие теоретические сведения

Программирование — это процесс проектирования, написания и отладки программ. Что такое программа? Программа — это последовательность инструкций, предназначенная для исполнения устройством управления вычислительной машины [1]. Есть, также, официальные трактовки термина «Программа»:

Программа — данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определённого алгоритма. [2]

Программа — представленная в объективной форме совокупность данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств с целью получения определённого результата, включая подготовительные материалы, полученные в ходе разработки программы для ЭВМ, и порождаемые ею аудиовизуальные отображения. [3]

3 Оборудование рабочего места

Персональный компьютер под управлением ОС Linux или Windows с установленной средой программирования IDLE и языком программирования Python версии 3.4 и выше.

4 Задание

- Ознакомиться с формальными определениями, требующими использования циклической конструкции.
- Выполнить примеры, приведенные в разделе «Порядок выполнения работы».
- Решить примеры для самостоятельной работы.

5 Содержание отчета

Отчет представляется в виде текстового документа, содержащего скриншоты окна редактора, содержащего код программы и командного окна с результатом выполнения программы для всех самостоятельных заданий.

6 Порядок выполнения работы

6.1 Необходимость использования циклов

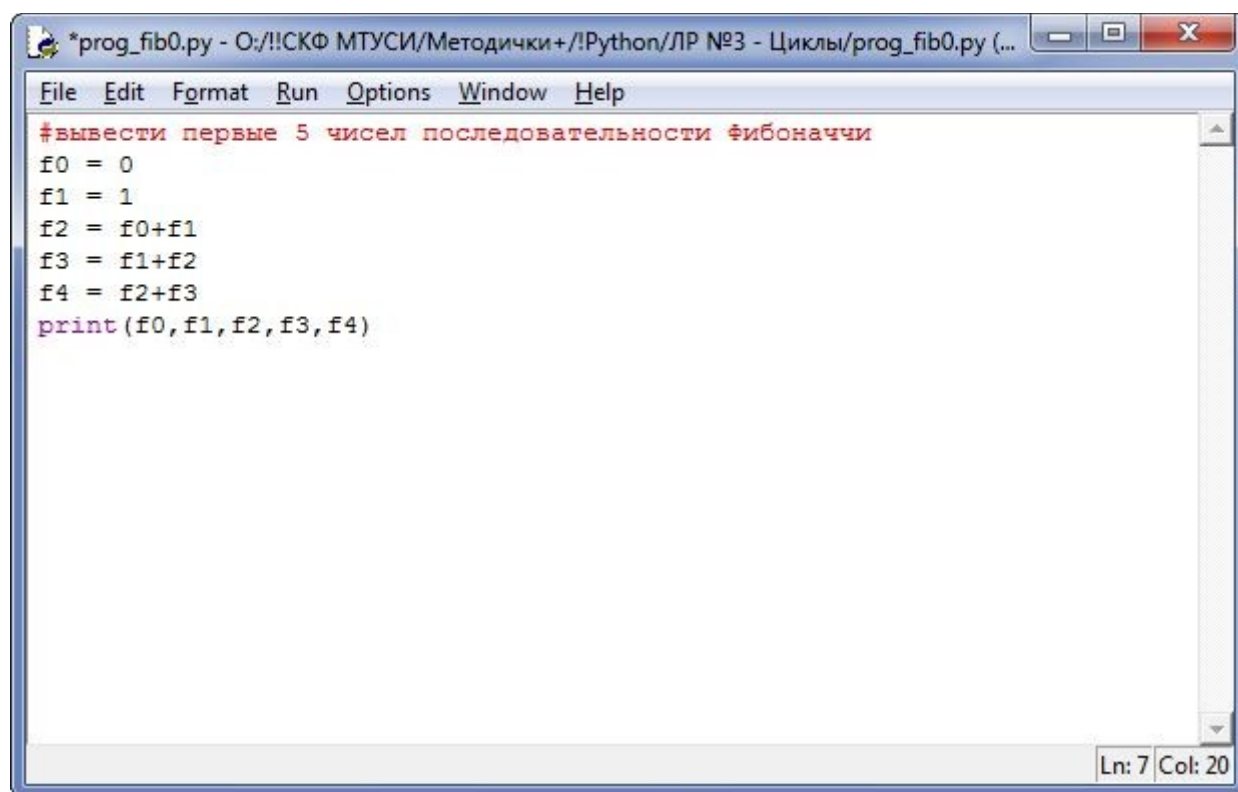
При составлении алгоритма и дальнейшем написании программы часто возникают ситуации, когда однотипные действия необходимо выполнить многократно над различными данными. Рассмотрим пример такого случая.

Числа Фибоначчи — это последовательность чисел, первые два числа из которой равны 0 и 1 соответственно, а каждое последующее — сумме двух предыдущих.

Формально последовательность чисел Фибоначчи задается следующим соотношением:

$$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2}, \quad n \geq 2, \quad n \in \mathbb{Z}.$$

Допустим, нам необходимо вывести на экран первые 5 чисел данной последовательности. Используя знания, полученные нами ранее, составим программу. Например, такую:

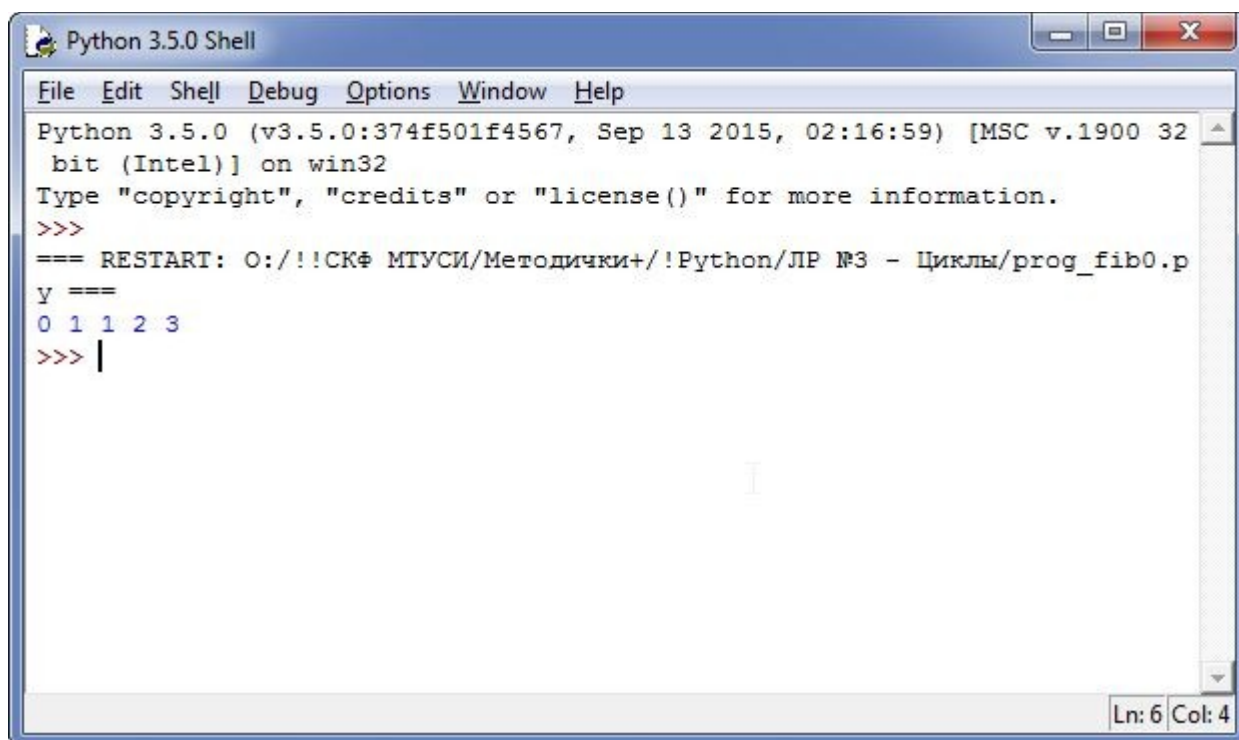


The screenshot shows a window titled '*prog_fib0.py - O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/prog_fib0.py (...'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python code:

```
#вывести первые 5 чисел последовательности фибоначчи
f0 = 0
f1 = 1
f2 = f0+f1
f3 = f1+f2
f4 = f2+f3
print(f0, f1, f2, f3, f4)
```

The status bar at the bottom right indicates 'Ln: 7 Col: 20'.

В результате выполнения программы на экран будет выведено следующее:



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32
bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/prog_fib0.p
y ===
0 1 1 2 3
>>> |
```

Да, мы написали программу, которая решает поставленную задачу. Но что делать, если необходимо будет вывести на экран 10 чисел? А если 50? Писать 50 строк кода, отличающихся только тем, что в них происходит сложение разных переменных?

Для многократного выполнения однотипных действий при программировании используются циклы. В языке Python используются две конструкции с циклом: `while` и `for`. Рассмотрим их ниже.

6.2 Цикл *for*

Цикл `for` в Python отличается от «простого» цикла «со счетчиком» в других языках программирования. Он представляет из себя перебор определенного набора значений, каждое из которых подставляется в переменную на одном шаге цикла и в общем виде записывается так:

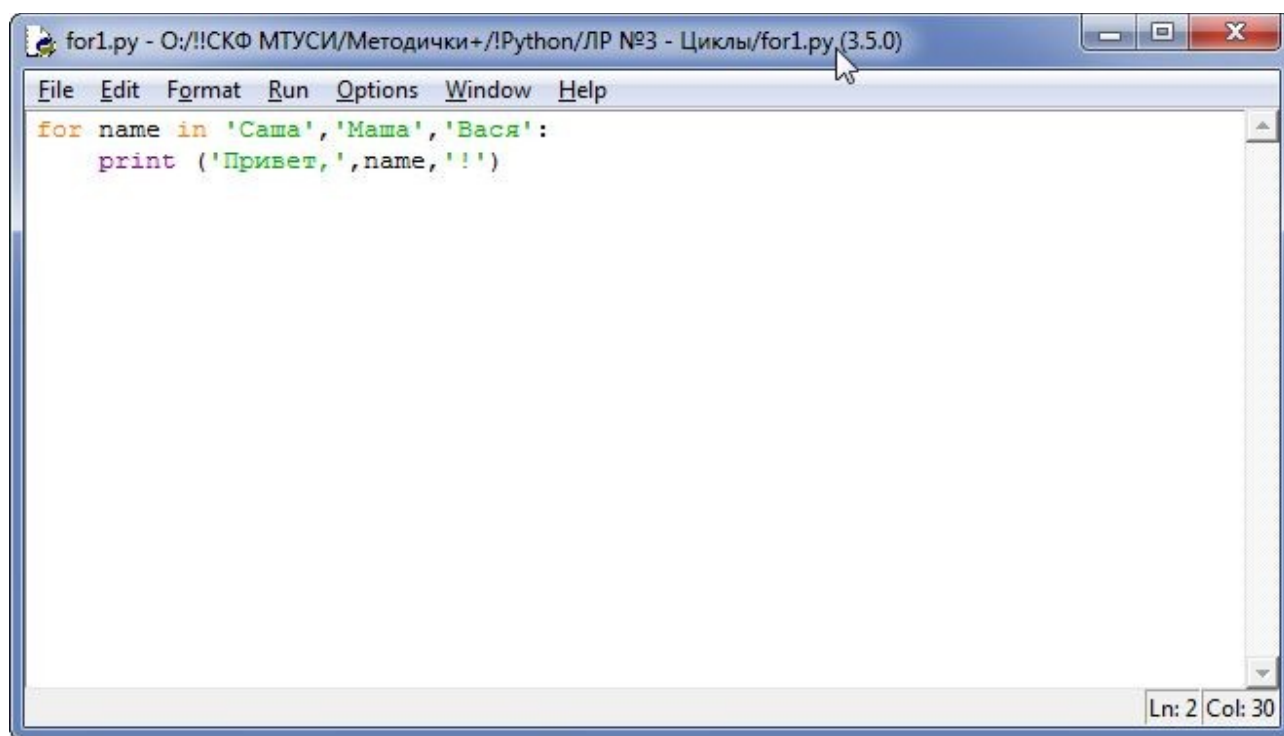
```
for [переменная] in [набор значений]:  
    [действие1]  
    [действие2]
```

Обратите внимание на отступ! В тело цикла попадают только те действия, которые записаны с отступом.

В качестве набора значений могут быть представлены:

- значения, перечисленные через запятую,
- диапазон значений,
- строка,
- словарь,
- список.

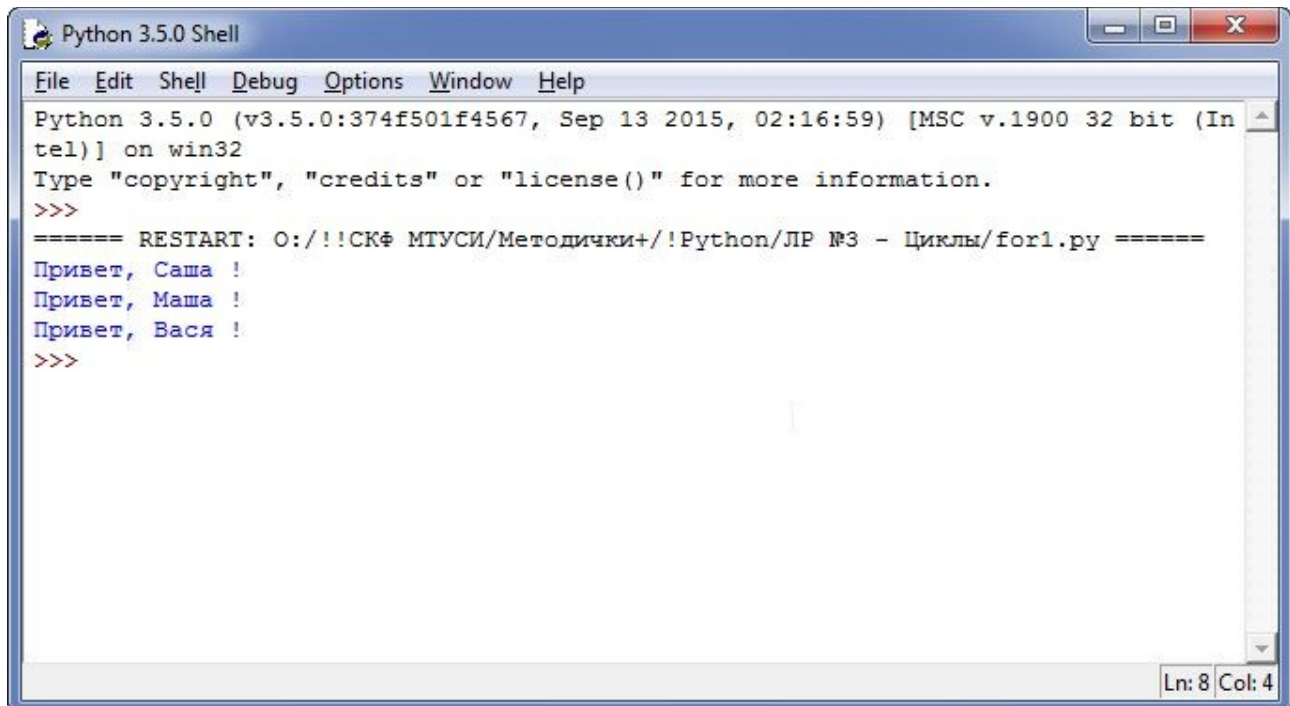
Рассмотрим пример перечисления значений:

A screenshot of a Python IDE window titled "for1.py - O:/!!СКФ МТУСИ/Методички+/IPython/ЛР №3 - Циклы/for1.py (3.5.0)". The window contains a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor shows the following Python code:

```
for name in 'Сама', 'Маша', 'Вася':  
    print ('Привет, ', name, '!')
```

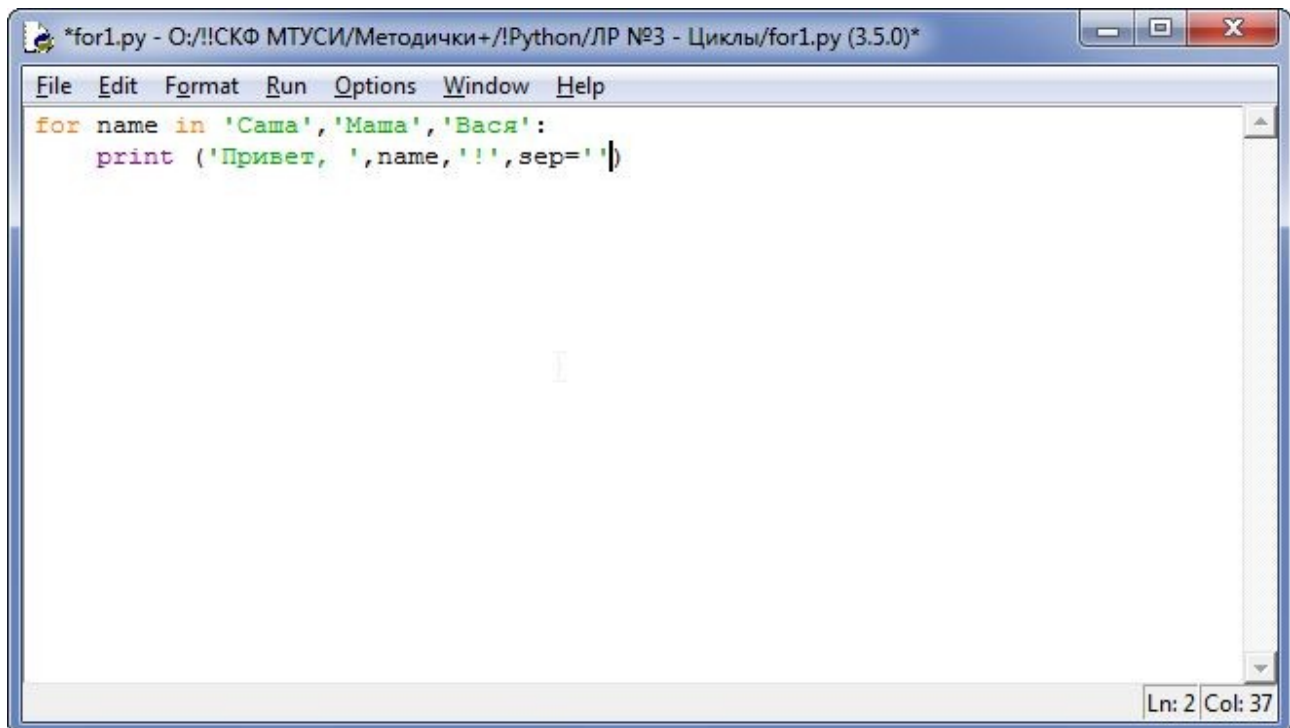
The status bar at the bottom right indicates "Ln: 2 Col: 30".

После выполнения такой программы на экран будет выведено:



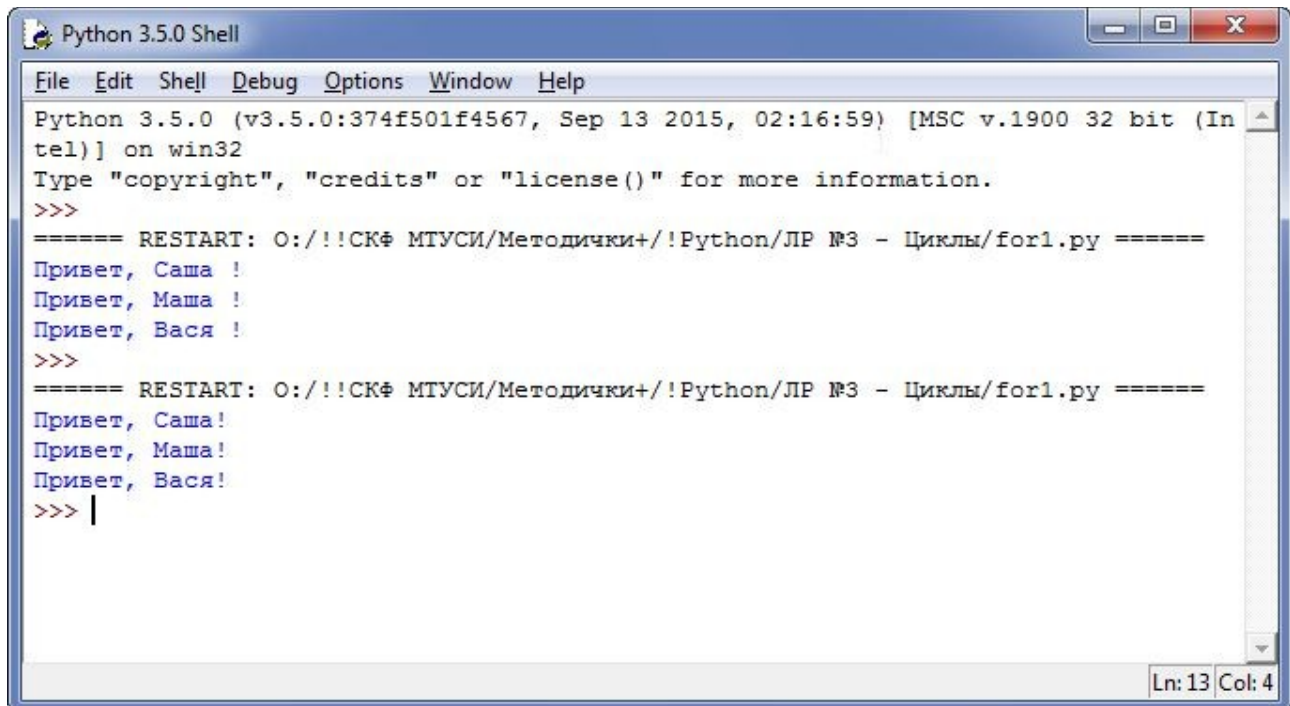
```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py =====
Привет, Саша !
Привет, Маша !
Привет, Вася !
>>>
```

Мелочь, а неприятно: пробел после слова «Привет» и запятой оказался кстати, а вот пробел после имени явно лишний! Одним из способов устранения данной неточности может быть изменение разделителя в выводе, и добавления пробела в первую часть строки:



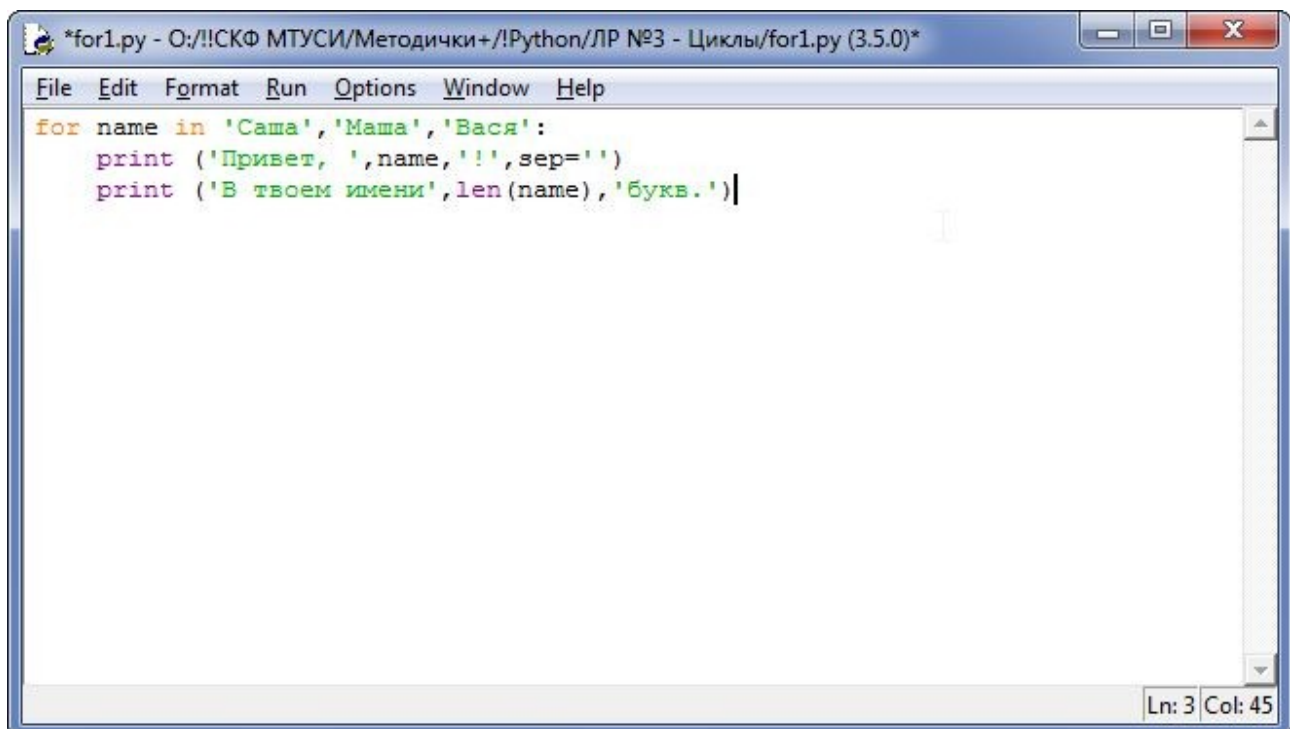
```
*for1.py - O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py (3.5.0)*
File Edit Format Run Options Window Help
for name in 'Саша', 'Маша', 'Вася':
    print ('Привет, ', name, '!', sep='|')
```

После этого вывод приобретает вид:



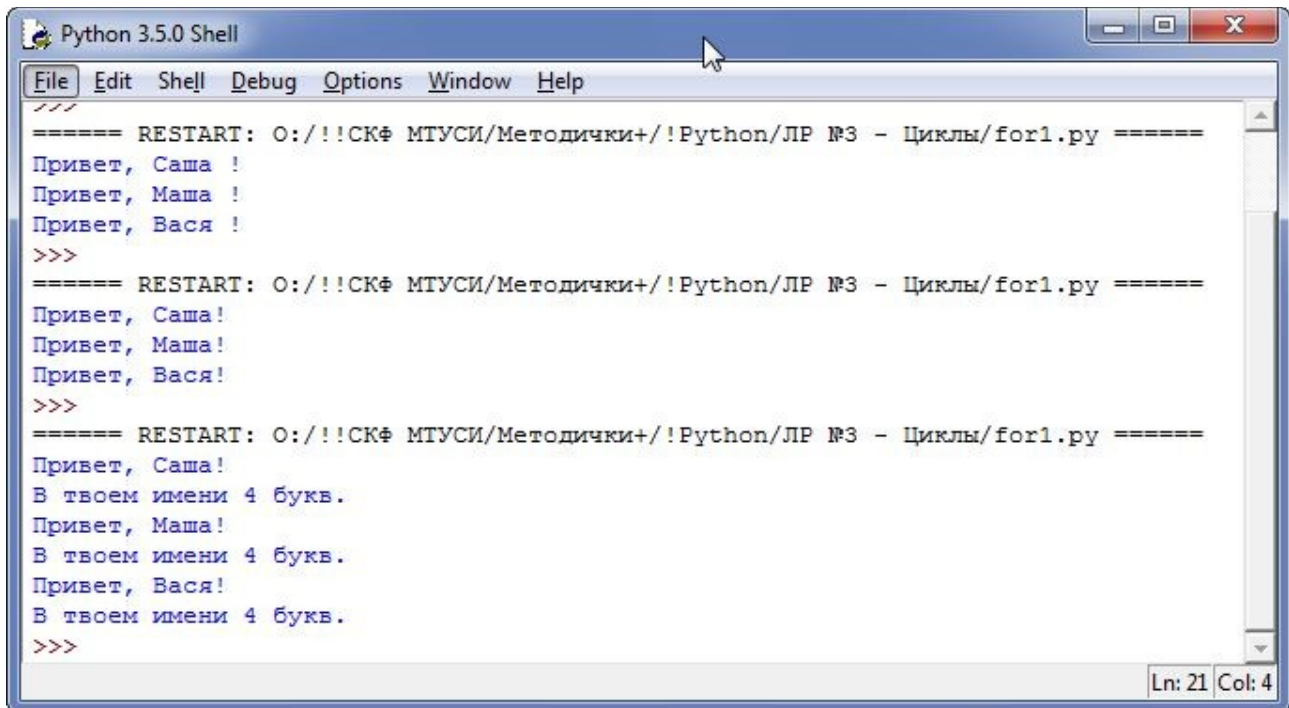
```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py =====
Привет, Саша !
Привет, Маша !
Привет, Вася !
>>>
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py =====
Привет, Саша!
Привет, Маша!
Привет, Вася!
>>> |
```

Продолжим работу с нашим примером! Добавим в программу ещё одно действие с переменной — определим длину имени. Используем для этого стандартную функцию `len`:



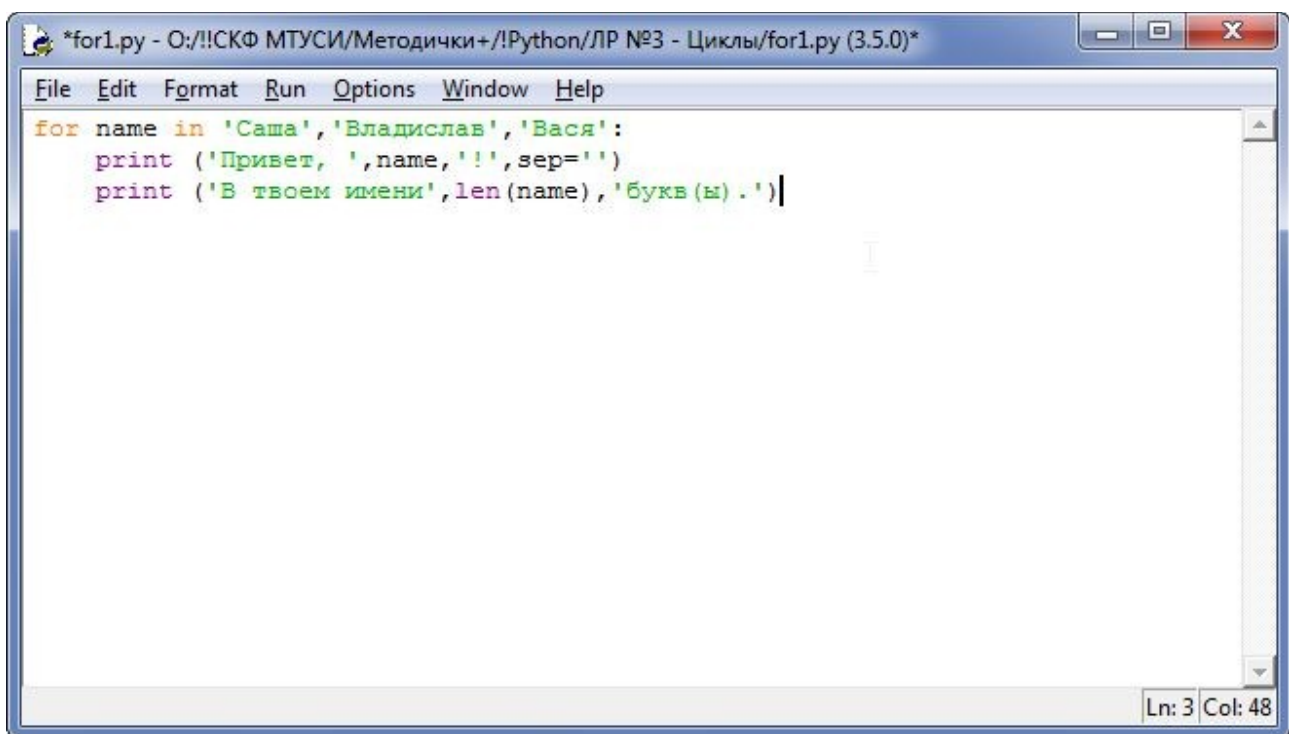
```
*for1.py - O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py (3.5.0)*
File Edit Format Run Options Window Help
for name in 'Саша', 'Маша', 'Вася':
    print ('Привет, ', name, '!', sep='')
    print ('В твоём имени', len(name), 'букв.')
```

Не самый удачный пример у нас получился:



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py =====
Привет, Саша !
Привет, Маша !
Привет, Вася !
>>>
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py =====
Привет, Саша!
Привет, Маша!
Привет, Вася!
>>>
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py =====
Привет, Саша!
В твоём имени 4 букв.
Привет, Маша!
В твоём имени 4 букв.
Привет, Вася!
В твоём имени 4 букв.
>>>
Ln: 21 Col: 4
```

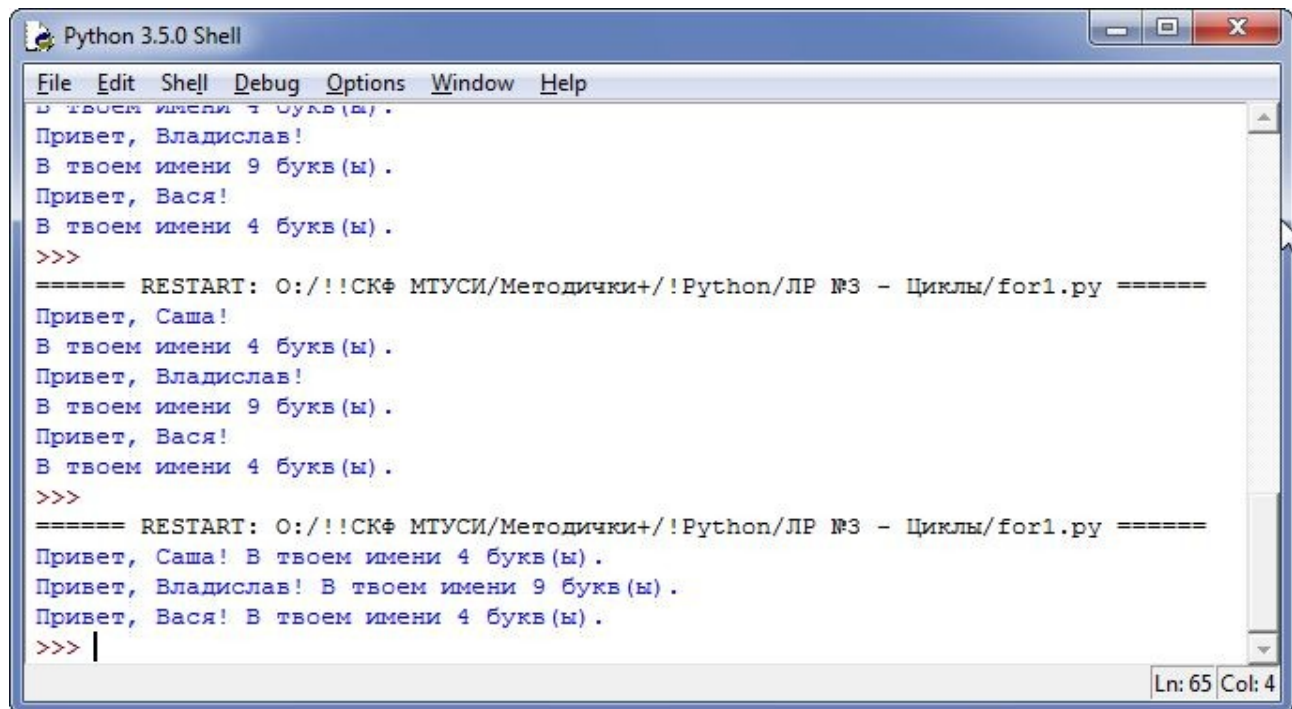
Изменим исходные данные и добавим и слегка подправим вывод:



```
*for1.py - O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py (3.5.0)*
File Edit Format Run Options Window Help
for name in 'Саша', 'Владислав', 'Вася':
    print ('Привет, ', name, '!', sep='')
    print ('В твоём имени', len(name), 'букв(ы) .')|
Ln: 3 Col: 48
```

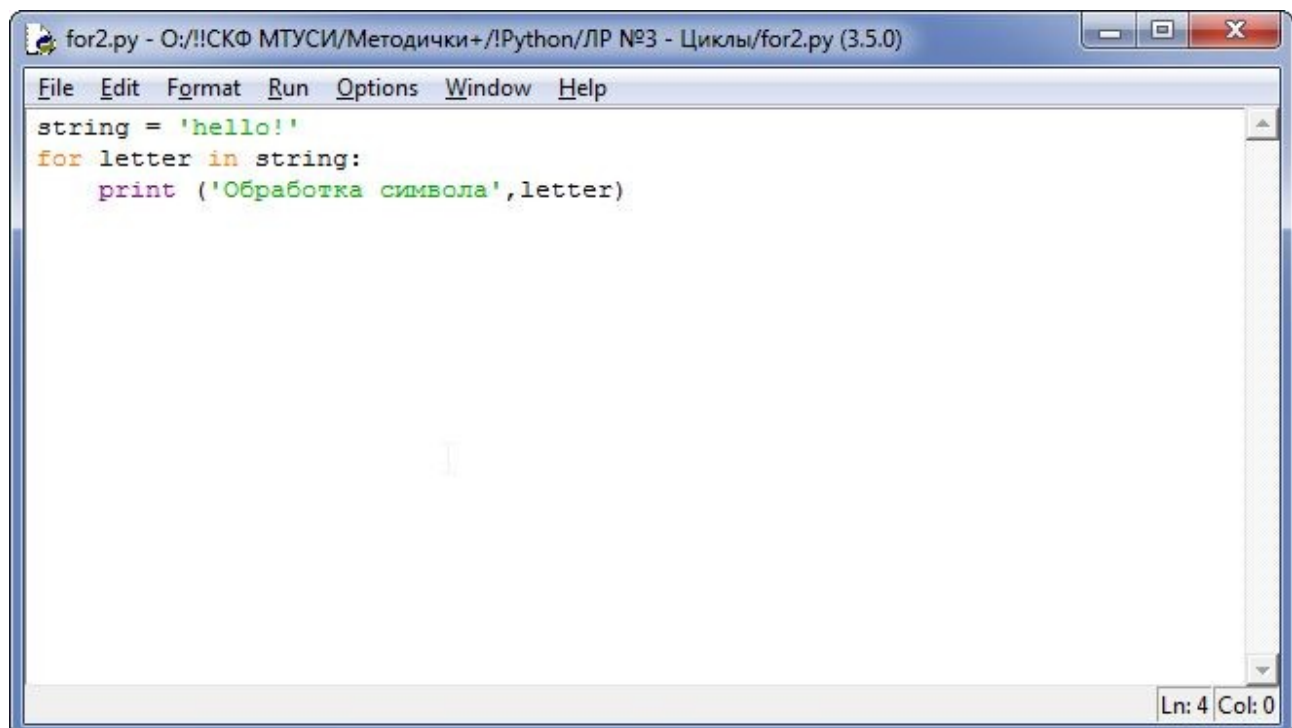
Выполните исправленный пример, обратите внимание на вывод.

Ещё немного поработаем с выводом — сделаем так, чтобы программа при выводе переходила на следующую строку только при переходе к следующему имени. Для этого необходимо изменить параметр `end` для одной из функций `print`. Самостоятельно исправьте программу, чтобы получить следующий вывод:



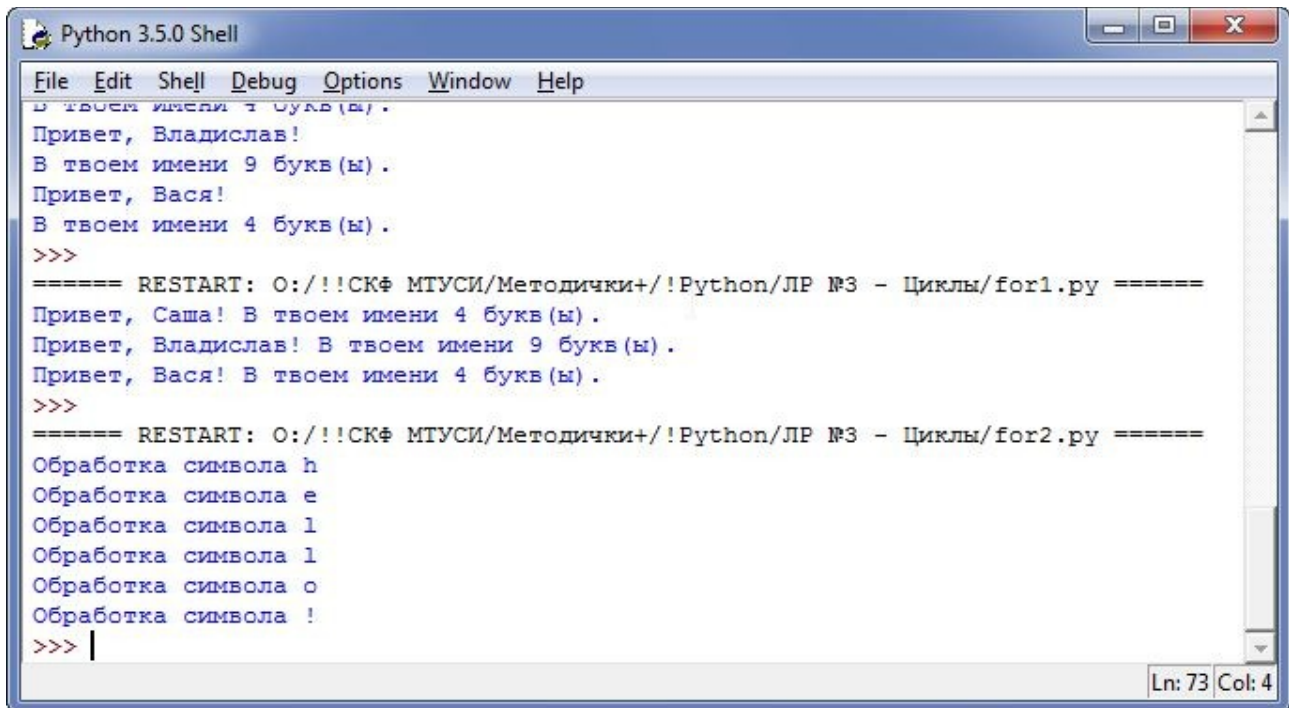
```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
В твоём имени 4 букв(ы) .
Привет, Владислав!
В твоём имени 9 букв(ы) .
Привет, Вася!
В твоём имени 4 букв(ы) .
>>>
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py =====
Привет, Саша!
В твоём имени 4 букв(ы) .
Привет, Владислав!
В твоём имени 9 букв(ы) .
Привет, Вася!
В твоём имени 4 букв(ы) .
>>>
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py =====
Привет, Саша! В твоём имени 4 букв(ы) .
Привет, Владислав! В твоём имени 9 букв(ы) .
Привет, Вася! В твоём имени 4 букв(ы) .
>>> |
Ln: 65 Col: 4
```

Если в качестве набора значений подана строка, цикл может выглядеть, например, следующим образом:



```
for2.py - O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for2.py (3.5.0)
File Edit Format Run Options Window Help
string = 'hello!'
for letter in string:
    print ('Обработка символа', letter)
Ln: 4 Col: 0
```

Результат работы цикла, в этом случае:



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
В твоём имени 7 букв(ы) .
Привет, Владислав!
В твоём имени 9 букв(ы) .
Привет, Вася!
В твоём имени 4 букв(ы) .
>>>
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for1.py =====
Привет, Саша! В твоём имени 4 букв(ы) .
Привет, Владислав! В твоём имени 9 букв(ы) .
Привет, Вася! В твоём имени 4 букв(ы) .
>>>
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for2.py =====
Обработка символа h
Обработка символа e
Обработка символа l
Обработка символа l
Обработка символа o
Обработка символа !
>>> |
```

Наиболее близок привычному многим «циклу со счетчиком» режим использования цикла `for`, когда в качестве набора значений используется диапазон (`range`). В этом случае в общем виде цикл выглядит следующим образом:

```
for i in range(start,to,step):
    print(i)
```

В данной записи `range` – это диапазон значений, которые последовательно принимает переменная `i`. В полном варианте записи вводится 3 аргумента:

`start` – нижняя граница, начальное значение,

`to` – верхняя граница, не входит в диапазон (конечное значение будет на единицу меньше),

`step` – шаг, с которым происходит движение от `start` к `to`.

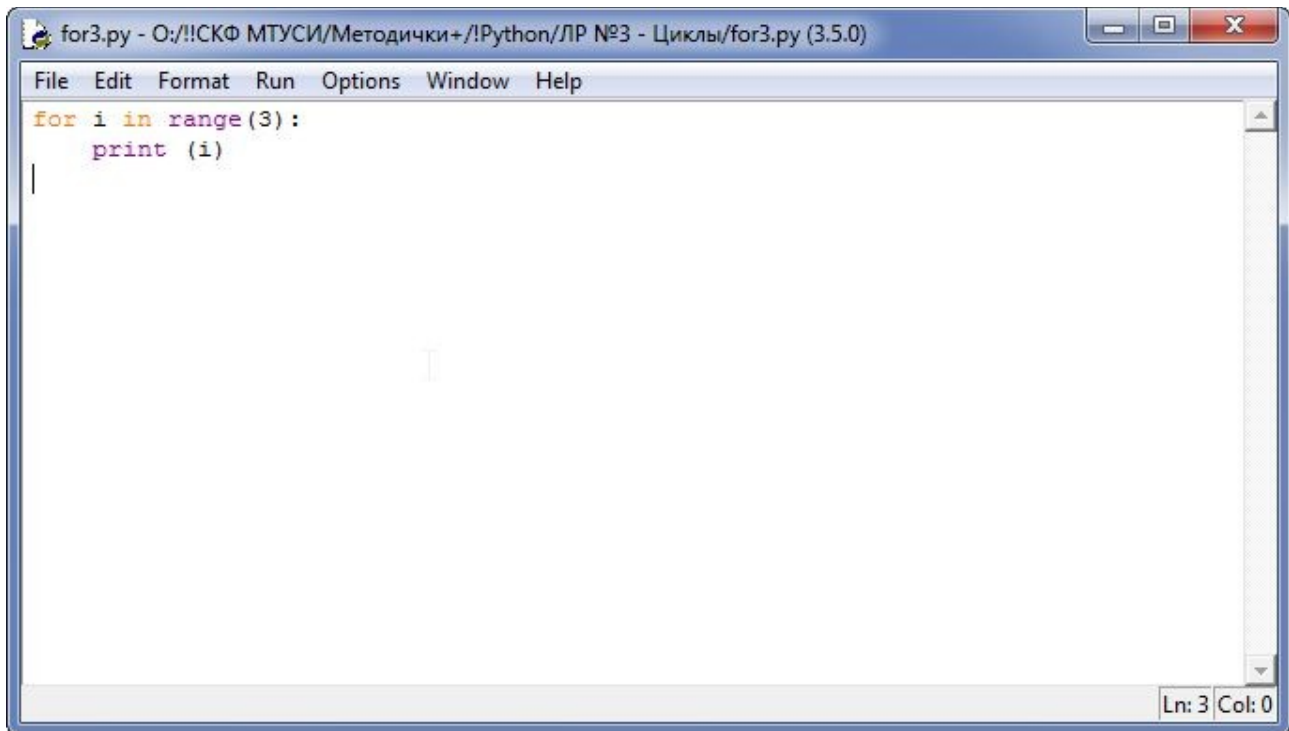
Допустимо использование `range` с 1 или 2 аргументами. В этом случае они распознаются как:

- 1 аргумент - `to`
- 2 аргумента - `start,to`

Пропущенные элементы заменяются значениями по умолчанию:

- `start=0`
- `step=1`

Например, в следующем цикле:

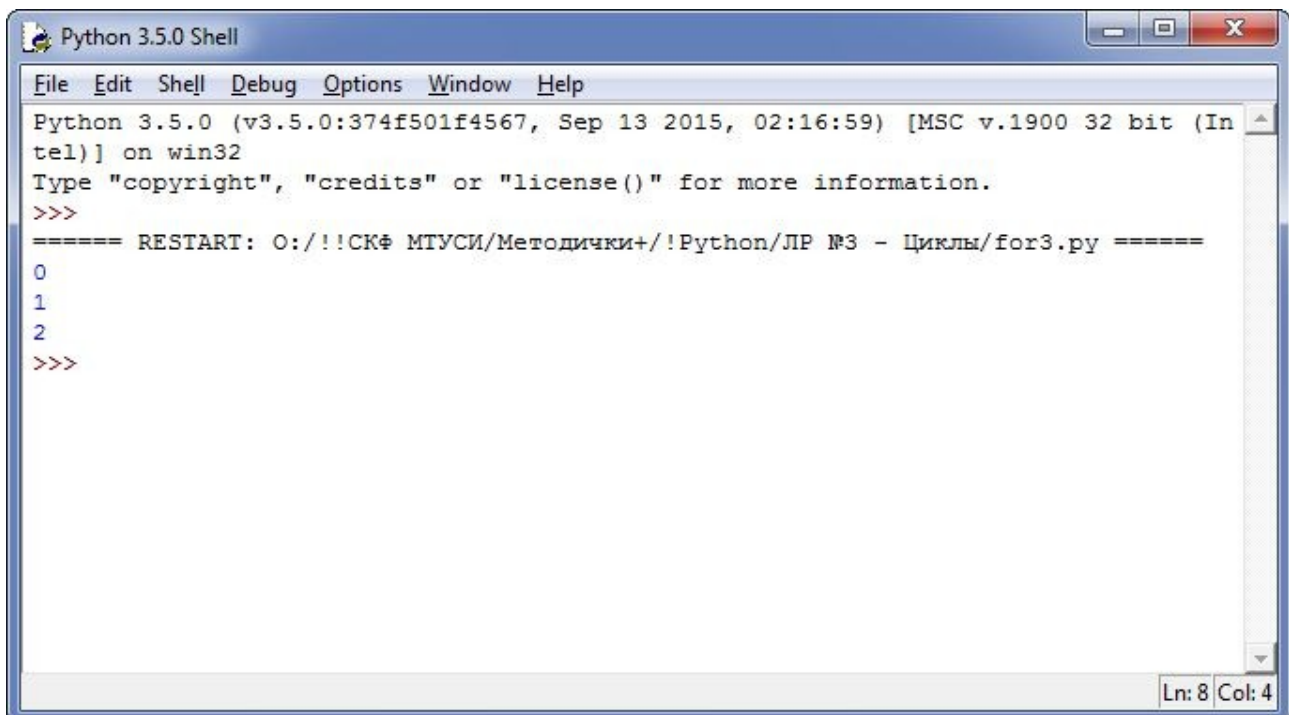


The screenshot shows a window titled "for3.py - O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for3.py (3.5.0)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python code:

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

The status bar at the bottom right indicates "Ln: 3 Col: 0".

переменная *i* примет значения:

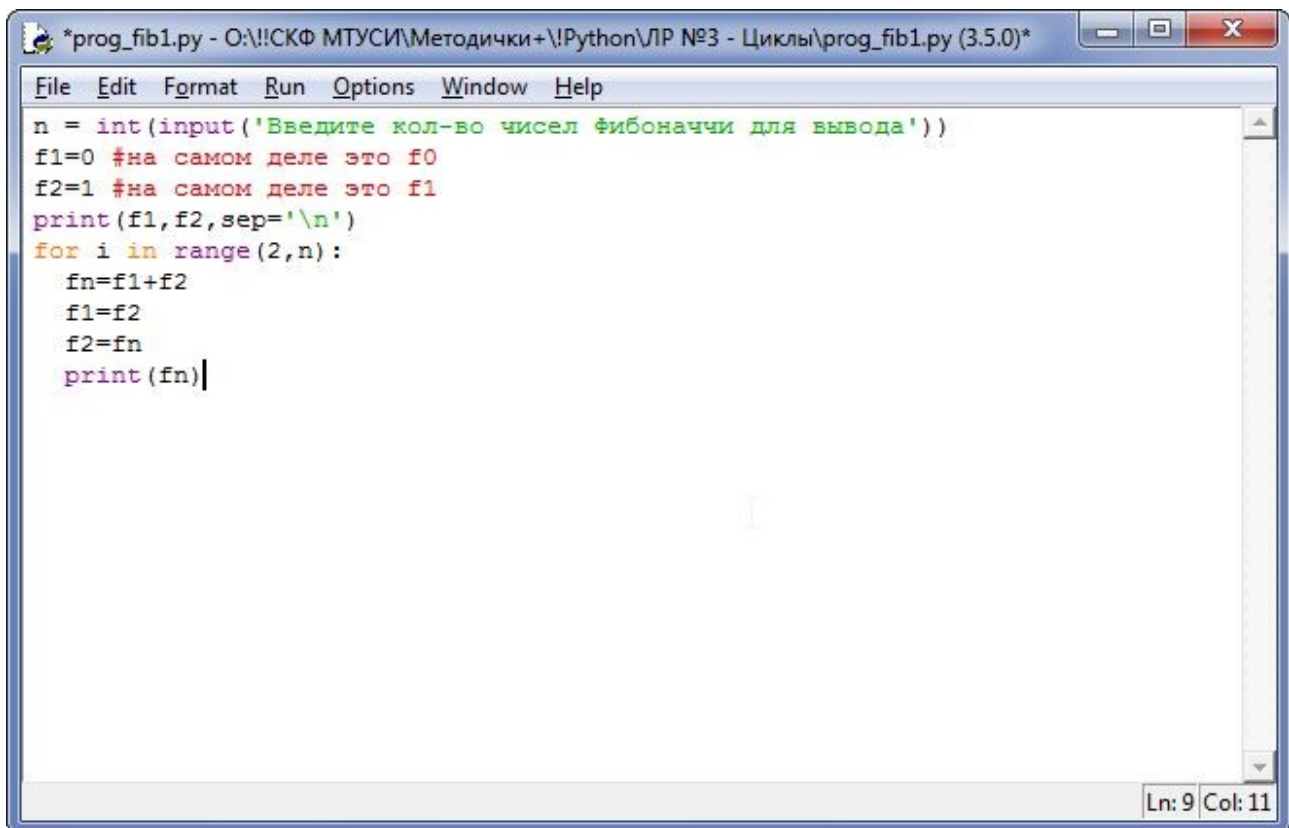


The screenshot shows a "Python 3.5.0 Shell" window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The text in the shell is as follows:

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/for3.py =====  
0  
1  
2  
>>>
```

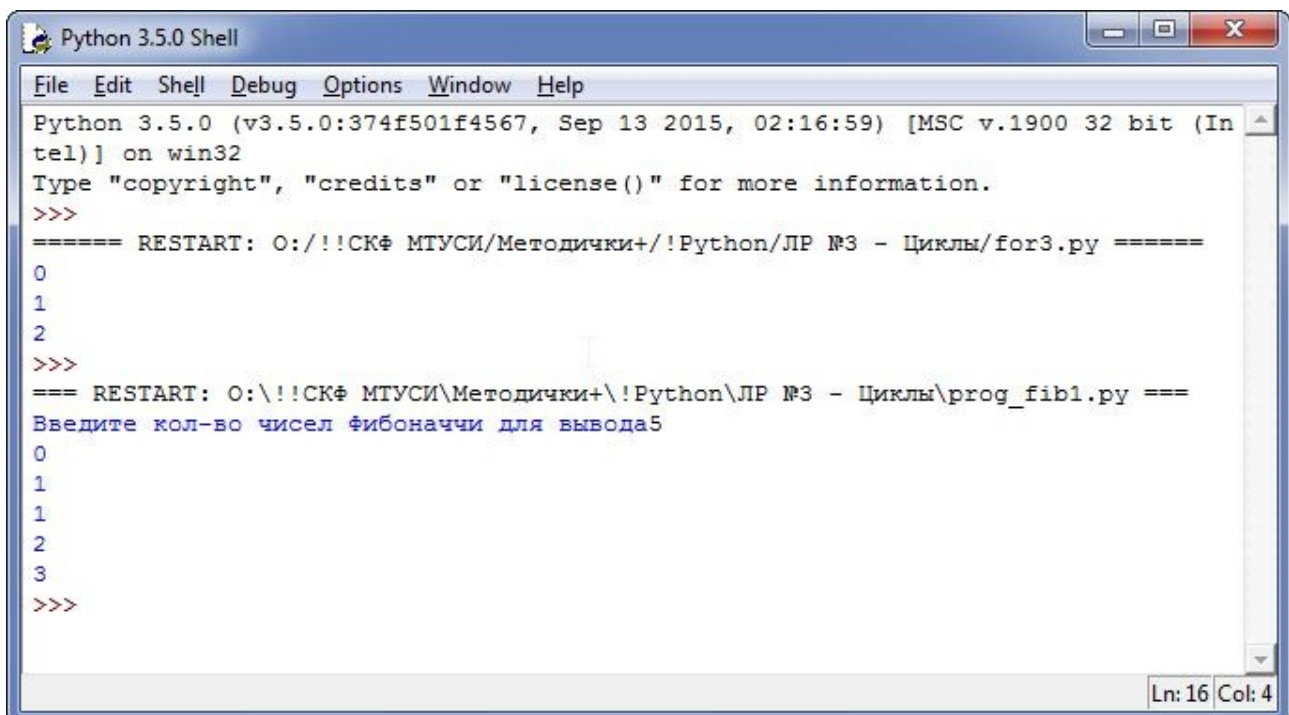
The status bar at the bottom right indicates "Ln: 8 Col: 4".

Вернемся к нашему первоначальному примеру — выводу чисел Фибоначчи на экран. Используем цикл `for` для того, чтобы решить данную задачу. Добавим запрос на ввод с клавиатуры количества чисел для вывода. Получится примерно такая программа:



```
*prog_fib1.py - O:\!\СКФ МТУСИ\Методички+\!Python\ЛР №3 - Циклы\prog_fib1.py (3.5.0)*
File Edit Format Run Options Window Help
n = int(input('Введите кол-во чисел фибоначчи для вывода'))
f1=0 #на самом деле это f0
f2=1 #на самом деле это f1
print(f1,f2,sep='\n')
for i in range(2,n):
    fn=f1+f2
    f1=f2
    f2=fn
    print(fn)
Ln: 9 Col: 11
```

А результат её выполнения будет следующим:



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: O:\/!\СКФ МТУСИ\Методички+\!Python\ЛР №3 - Циклы\for3.py =====
0
1
2
>>>
=== RESTART: O:\/!\СКФ МТУСИ\Методички+\!Python\ЛР №3 - Циклы\prog_fib1.py ===
Введите кол-во чисел фибоначчи для вывода5
0
1
1
2
3
>>>
Ln: 16 Col: 4
```

Задание

Исправьте программу так, чтобы числа Фибоначчи выводились без перехода на новую строку.

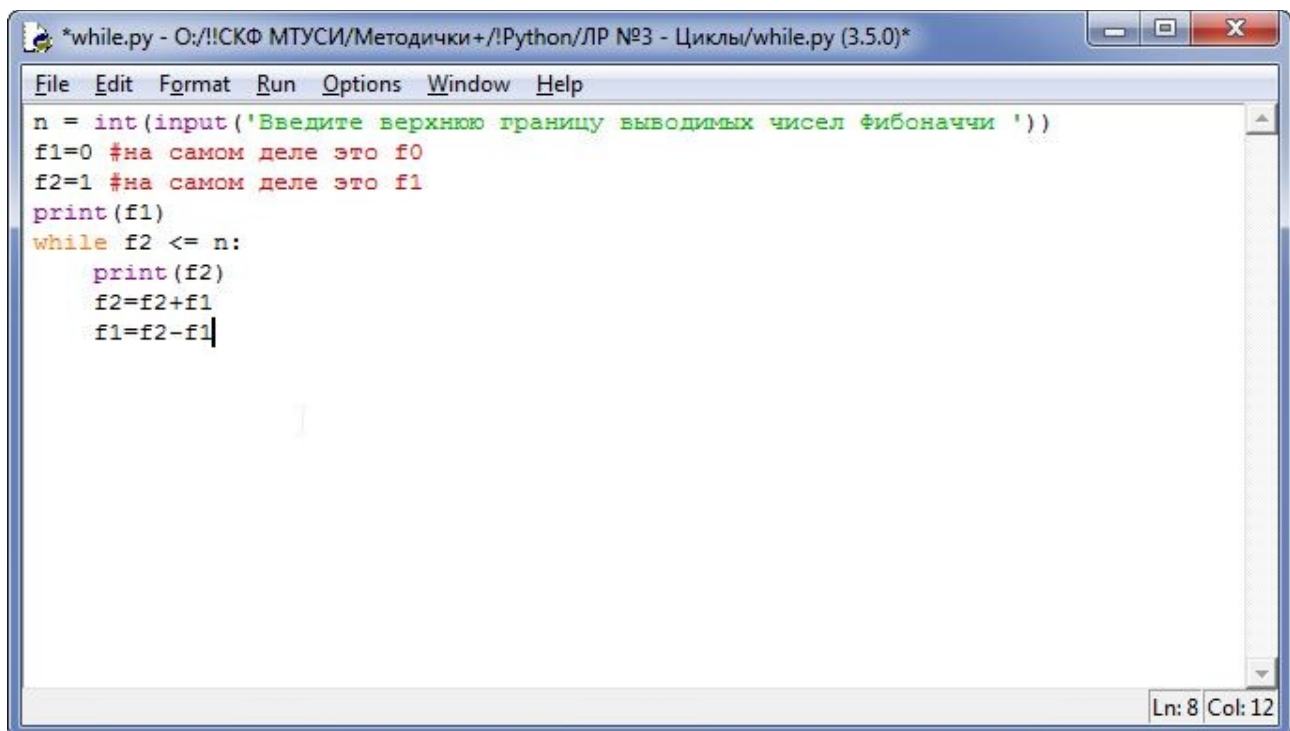
6.3 Цикл *while*

Цикл *while* – это цикл, который многим знаком как «цикл с предусловием». В общем виде его можно записать так:

```
while [логическое выражение]:  
    [действие 1]  
    [действие 2]
```

И снова обратите внимание на оформление цикла! Для него используются двоеточие («:») и отступы. Цикл работает следующим образом: в первой же итерации сначала выполняется логическое выражение. Если его результат «истина» (True), выполняются действия в теле цикла. Обратите внимание: в цикле обязательно должна изменяться хотя бы одна переменная из тех, что используется в логическом выражении. В противном случае он будет бесконечным.

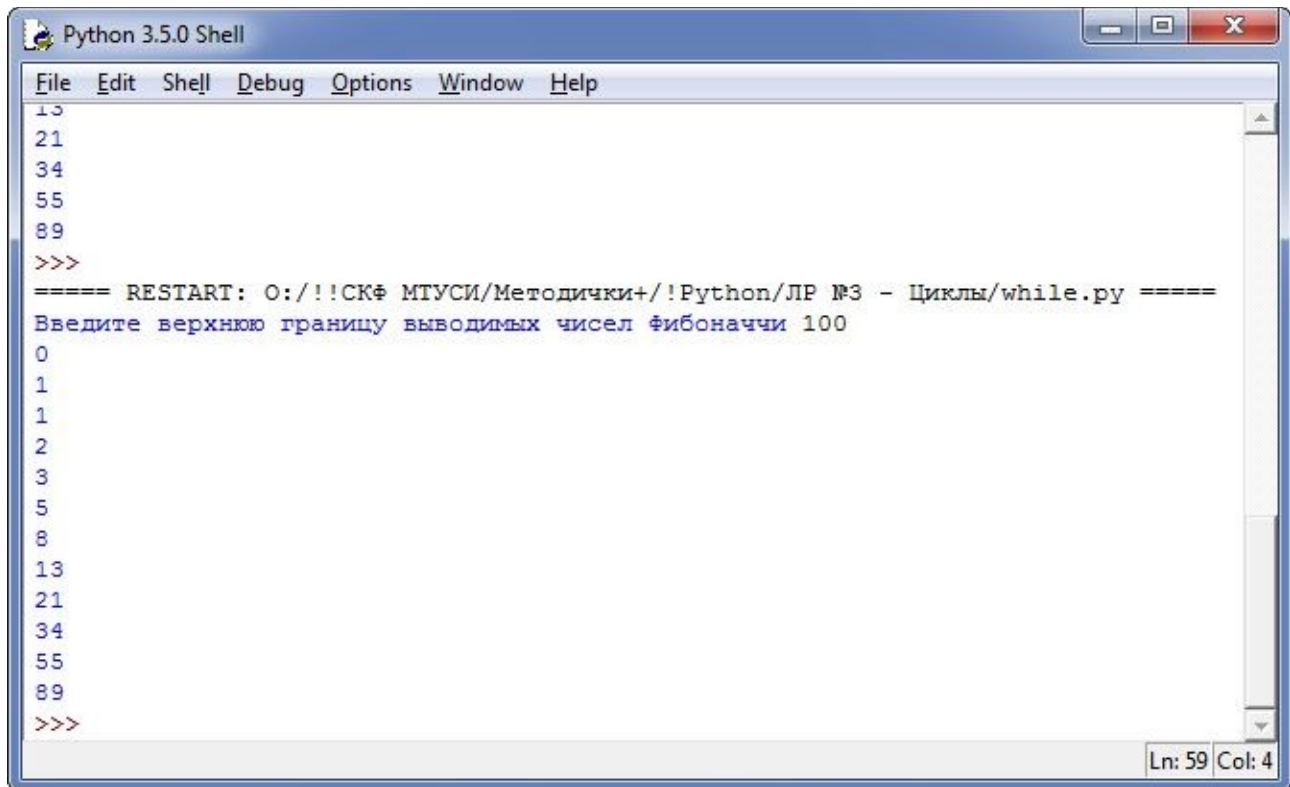
Рассмотрим пример применения данного цикла. Продолжим работать с числами Фибоначчи. Реализуем вывод чисел Фибоначчи меньше определенного значения. Сделать это можно, например, вот так:



```
*while.py - O:\!\СКФ МТУСИ\Методички+!\Python\ЛР №3 - Циклы\while.py (3.5.0)*  
File Edit Format Run Options Window Help  
n = int(input('Введите верхнюю границу выводимых чисел фибоначчи '))  
f1=0 #на самом деле это f0  
f2=1 #на самом деле это f1  
print(f1)  
while f2 <= n:  
    print(f2)  
    f2=f2+f1  
    f1=f2-f1
```

Ln: 8 Col: 12

Результат выполнения данной программы будет следующим:



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
13
21
34
55
89
>>>
===== RESTART: O:/!!СКФ МТУСИ/Методички+!/Python/ЛР №3 - Циклы/while.py =====
Введите верхнюю границу выводимых чисел Фибоначчи 100
0
1
1
2
3
5
8
13
21
34
55
89
>>>
```

Ln: 59 Col: 4

Итак, цикл `while` есть смысл использовать тогда, когда выполнение цикла должно остановиться в тот момент, когда одна из переменных примет определенное значение, но мы не можем спрогнозировать, на какой итерации цикла это произойдет.

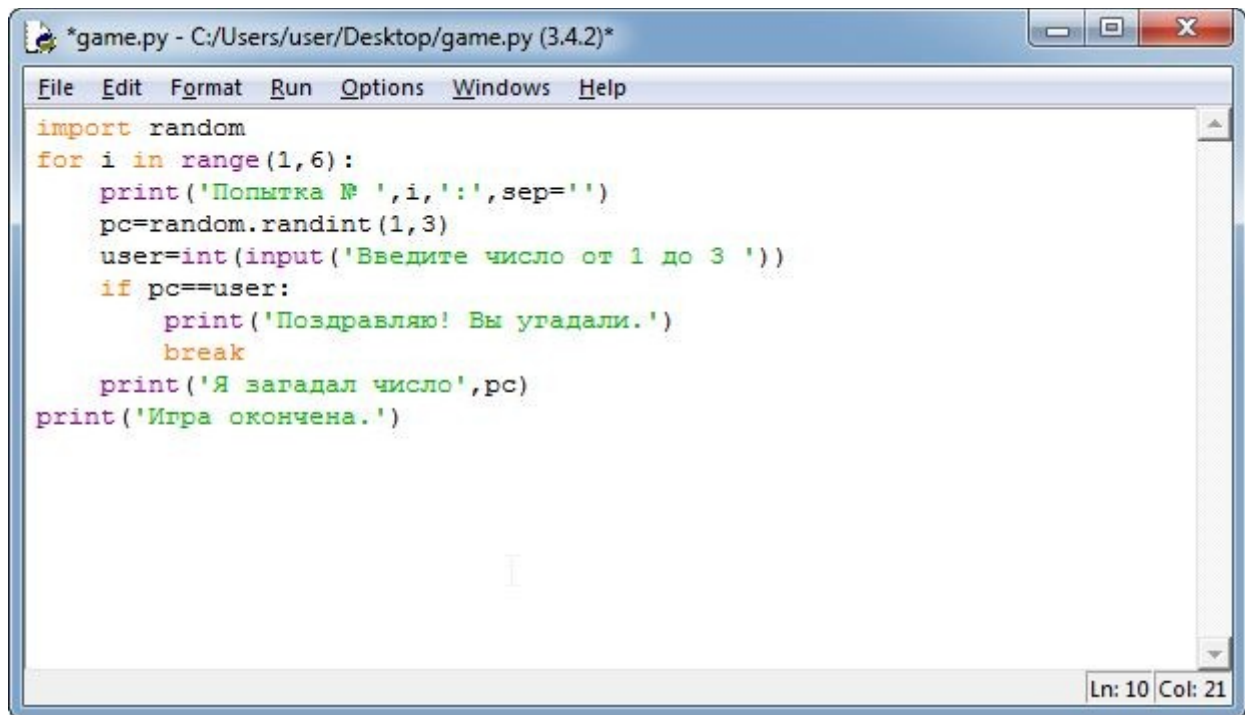
6.4 Управление циклом: *continue* и *break*

Бывают случаи, когда нужно перейти к следующей итерации цикла или прервать выполнение цикла. Для выполнения данных действий используются управляющие конструкции *continue* и *break*. Рассмотрим их действие более подробно.

continue используется для перехода к следующей итерации цикла.

break используется для завершения цикла.

Рассмотрим пример такой программы. Компьютер при помощи функции *random* «загадывает» числа от 1 до 3. Пользователю предлагается ввести с клавиатуры число от 1 до 3. Если пользователь угадал число, компьютер выводит соответствующее сообщение. Если нет — выводит число пользователя и «загаданное число». Каждый раз компьютер выводит номер попытки. После 5 попыток игра заканчивается в любом случае.

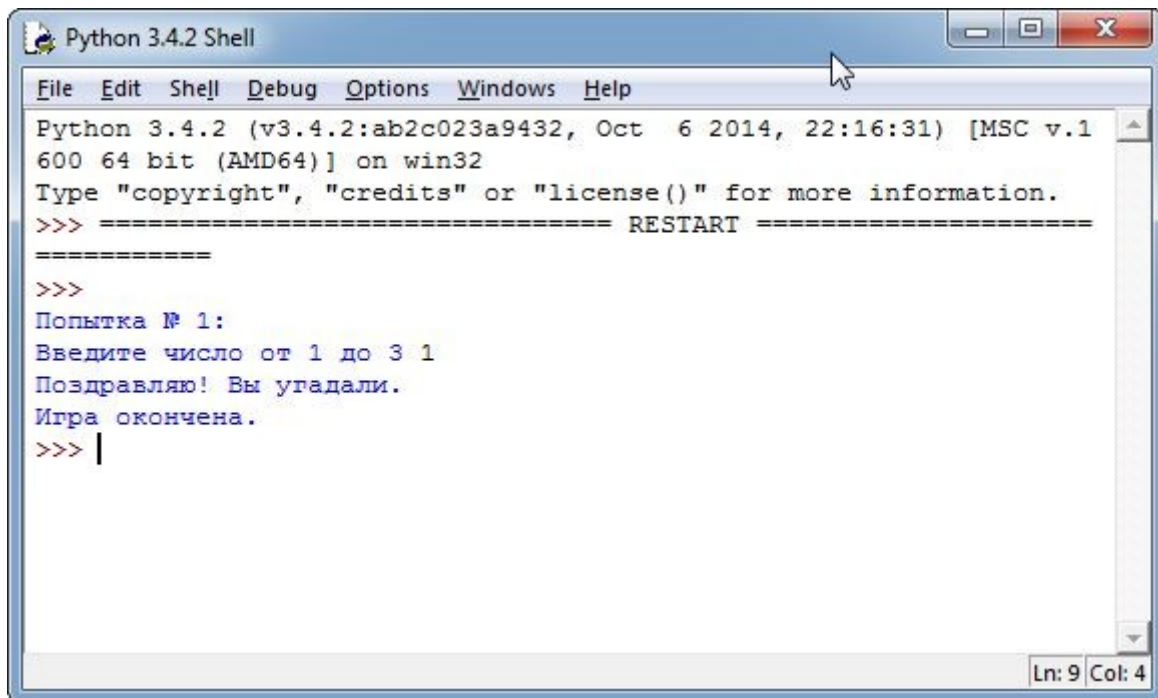
A screenshot of a Python IDE window titled '*game.py - C:/Users/user/Desktop/game.py (3.4.2)*'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Windows', and 'Help'. The code editor contains the following Python code:

```
import random
for i in range(1,6):
    print('Попытка № ',i,':',sep='')
    pc=random.randint(1,3)
    user=int(input('Введите число от 1 до 3 '))
    if pc==user:
        print('Поздравляю! Вы угадали.')
        break
    print('Я загадал число',pc)
print('Игра окончена.')
```

The status bar at the bottom right shows 'Ln: 10 Col: 21'.

И несколько примеров запуска программы.

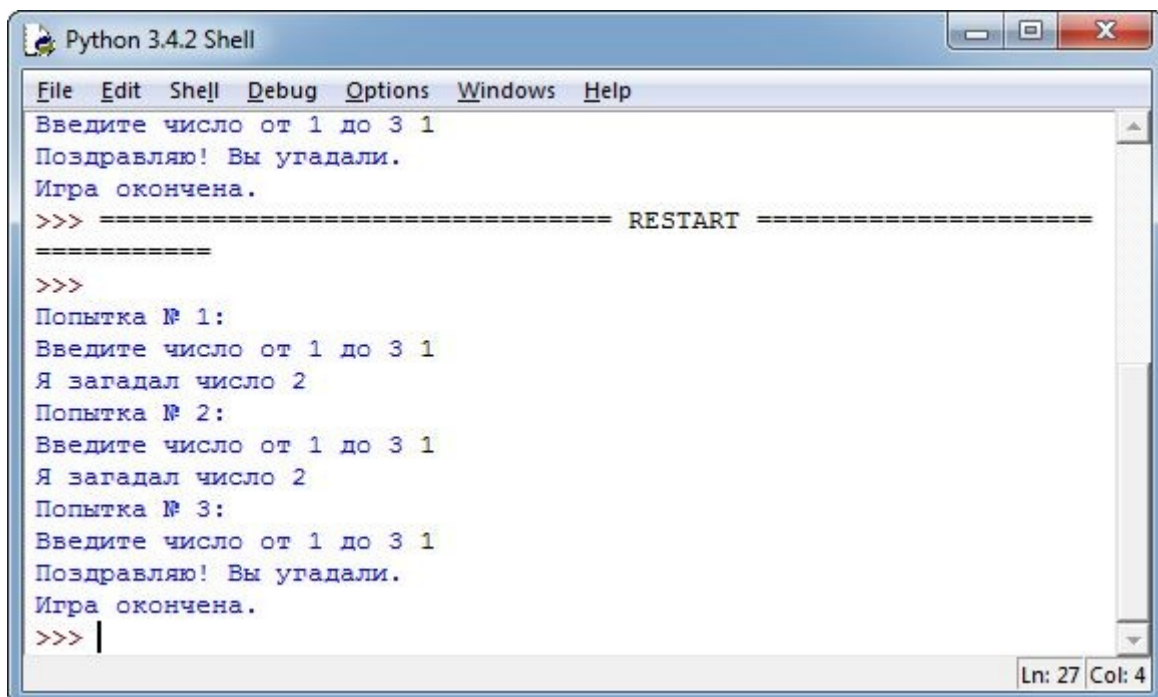
Угадано с первой попытки:



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct 6 2014, 22:16:31) [MSC v.1
600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Попытка № 1:
Введите число от 1 до 3 1
Поздравляю! Вы угадали.
Игра окончена.
>>> |
```

Ln: 9 Col: 4

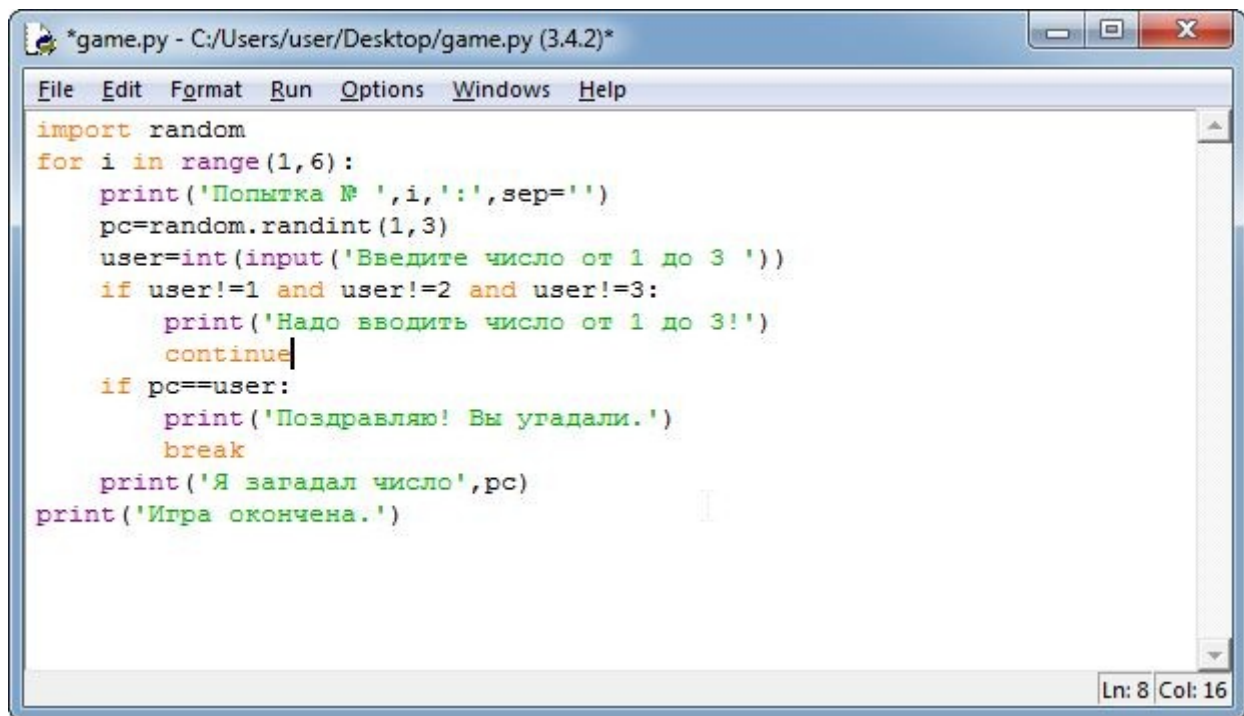
Или с третьей:



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Введите число от 1 до 3 1
Поздравляю! Вы угадали.
Игра окончена.
>>> ===== RESTART =====
>>>
Попытка № 1:
Введите число от 1 до 3 1
Я загадал число 2
Попытка № 2:
Введите число от 1 до 3 1
Я загадал число 2
Попытка № 3:
Введите число от 1 до 3 1
Поздравляю! Вы угадали.
Игра окончена.
>>> |
```

Ln: 27 Col: 4

Недостатком является то, что пользователь может ввести любое число (или даже вовсе не число). С учетом того, что мы имеем всего 3 «разрешенных» варианта ответа, не сложно устроить проверку. Логично, что в случае, если пользователь ввел недопустимую строку, нет смысла выводить для него число, загаданное компьютером. Попробуем реализовать данную проверку:



The image shows a screenshot of a Python IDE window titled "*game.py - C:/Users/user/Desktop/game.py (3.4.2)*". The window has a menu bar with File, Edit, Format, Run, Options, Windows, and Help. The code editor contains the following Python code:

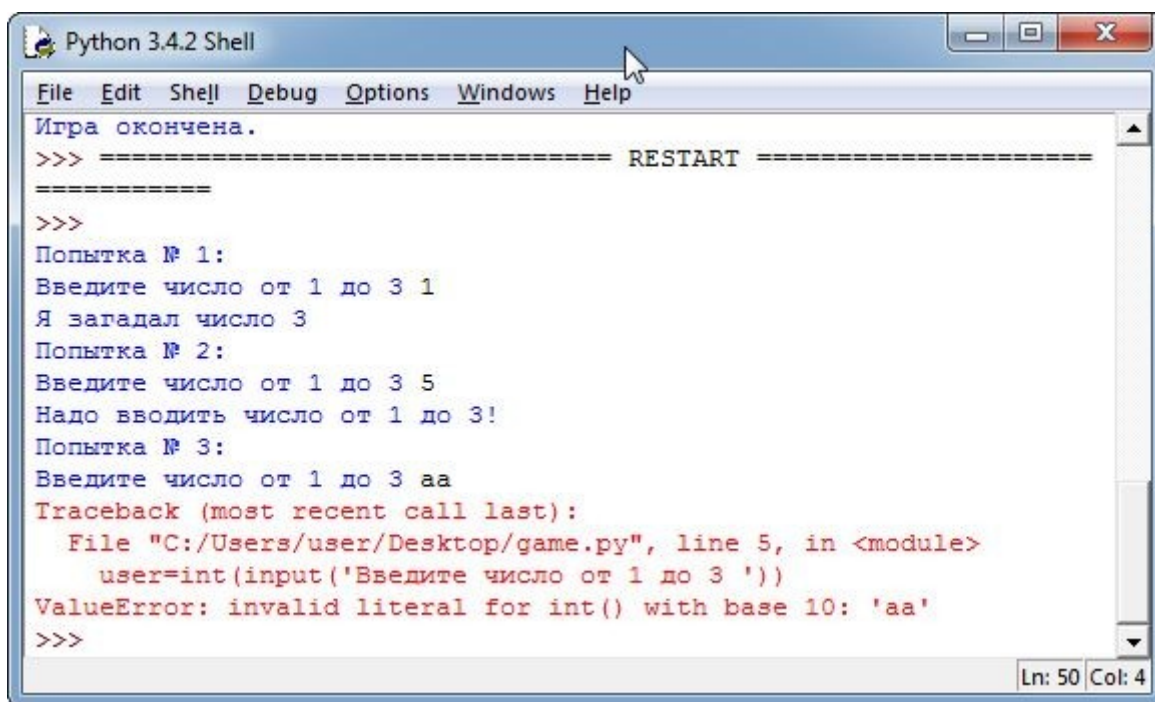
```
import random
for i in range(1, 6):
    print('Попытка № ', i, ': ', sep='')
    pc=random.randint(1, 3)
    user=int(input('Введите число от 1 до 3 '))
    if user!=1 and user!=2 and user!=3:
        print('Надо вводить число от 1 до 3!')
        continue
    if pc==user:
        print('Поздравляю! Вы угадали.')
        break
    print('Я загадал число', pc)
print('Игра окончена.')
```

The status bar at the bottom right indicates "Ln: 8 Col: 16".

Теперь если пользователь введет число, которое не равно ни 1, ни 2, ни 3, компьютер не будет показывать свое число, а «напомнит», какие числа он принимает. Попытка, при этом, будет использована.

Однако, если ввести вместо числа строку, программа выдаст ошибку.

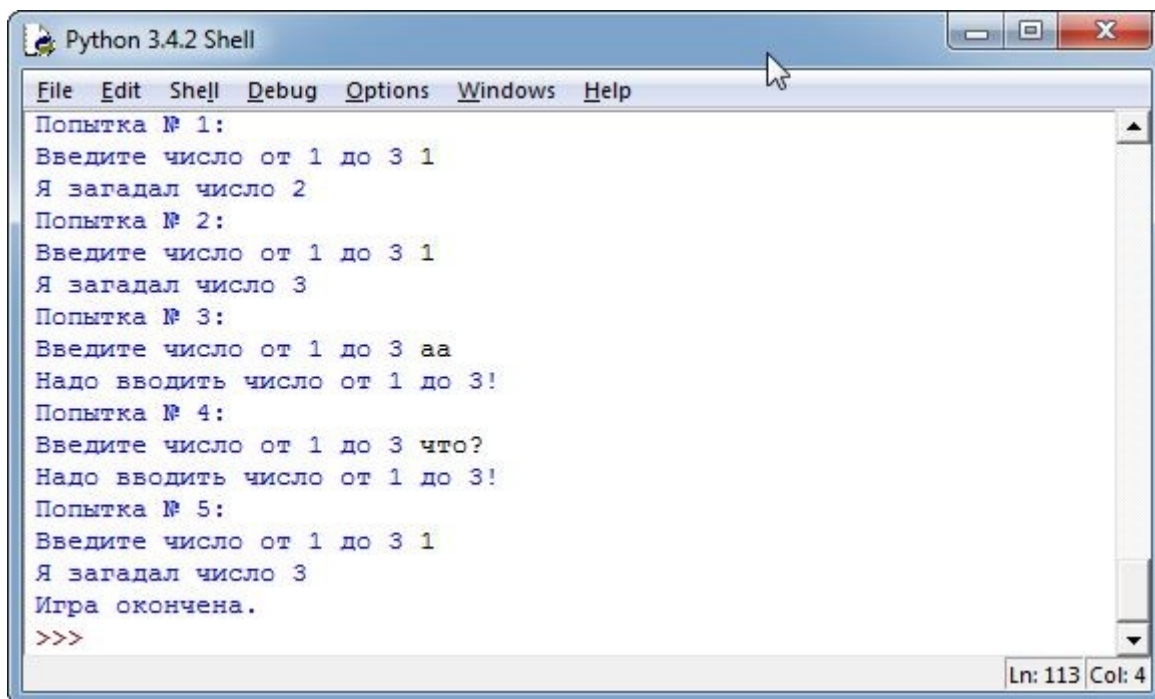
Рассмотрим оба случая:



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Игра окончена.
>>> ===== RESTART =====
>>>
Попытка № 1:
Введите число от 1 до 3 1
Я загадал число 3
Попытка № 2:
Введите число от 1 до 3 5
Надо вводить число от 1 до 3!
Попытка № 3:
Введите число от 1 до 3 aa
Traceback (most recent call last):
  File "C:/Users/user/Desktop/game.py", line 5, in <module>
    user=int(input('Введите число от 1 до 3 '))
ValueError: invalid literal for int() with base 10: 'aa'
>>>
```

Задание

Исправить программу таким образом, чтобы она корректно обрабатывала ввод строки, например:



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Попытка № 1:
Введите число от 1 до 3 1
Я загадал число 2
Попытка № 2:
Введите число от 1 до 3 1
Я загадал число 3
Попытка № 3:
Введите число от 1 до 3 aa
Надо вводить число от 1 до 3!
Попытка № 4:
Введите число от 1 до 3 что?
Надо вводить число от 1 до 3!
Попытка № 5:
Введите число от 1 до 3 1
Я загадал число 3
Игра окончена.
>>>
```

Прокомментировать текст программы.

6.5 Самостоятельная работа

1. Используя функцию `random.randint(0,4)` сгенерировать случайное число t от 0 до 4. Сделать скриншот запуска программы.

2. Написать на языке Python программы для решения задачи. Предусмотреть ввод и вывод данных.

Необходимо решить 3 примера. Номера вариантов заданий n выбираются по правилу:

$$n_i = 5 \cdot (i-1) + t,$$

где t – результат работы генератора в предыдущем задании,

i – порядковый номер выбираемого задания (1,2,3).

Т.е. выполняется условие $n \bmod 4 = t$.

В отчет по каждой программе включить:

- текст программы
- скриншот окна редактора с готовой программой,
- скриншот окна сразу после выполнения программы.

Варианты заданий:

0. С клавиатуры вводятся целые числа K и N ($N > 0$). Вывести N раз число K .

1. С клавиатуры вводятся два целых числа A и B ($A < B$). Вывести в порядке возрастания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество N этих чисел.

2. С клавиатуры вводятся два целых числа A и B ($A < B$). Вывести в порядке убывания все целые числа, расположенные между A и B (не включая числа A и B), а также количество N этих чисел.

3. С клавиатуры вводится число с плавающей точкой — цена 1 кг конфет. Вывести стоимость 1, 2, ..., 10 кг конфет.

4. С клавиатуры вводится число с плавающей точкой — цена 1 кг конфет. Вывести стоимость 0.1, 0.2, ..., 1 кг конфет.

5. С клавиатуры вводится число с плавающей точкой — цена 1 кг конфет. Вывести стоимость 1.2, 1.4, ..., 2 кг конфет.

6. С клавиатуры вводятся два целых числа A и B ($A < B$). Найти сумму всех целых чисел от A до B включительно.

7. С клавиатуры вводятся два целых числа A и B ($A < B$). Найти произведение всех целых чисел от A до B включительно.

8. С клавиатуры вводятся два целых числа A и B ($A < B$). Найти сумму квадратов всех целых чисел от A до B включительно.

9. С клавиатуры вводятся два целых числа A и B ($A < B$). Вывести в порядке возрастания все четные числа, расположенные между A и B (включая сами числа A и B), а также количество N этих чисел.

10. Для чисел от 1 до 100 методом перебора определить кол-во чисел, которые делятся на 5 или 7 без остатка.

11. Для чисел от 1 до 100 методом перебора определить кол-во чисел, которые делятся на 2 и 5 без остатка.

12. Для чисел от 1 до 100 методом перебора определить кол-во чисел, которые делятся на 2 или 3 или 5 без остатка.

13. Для чисел от 1 до 100 методом перебора определить кол-во чисел, которые делятся на 2 и 3 и 5 без остатка.

14. Для чисел от 1 до 100 методом перебора определить кол-во чисел, которые не делятся без остатка ни на 2 ни на 3.

6.6 Дополнительное задание

Напишите программу, на вход которой даются четыре числа a , b , c и d , каждое в своей строке. Программа должна вывести фрагмент таблицы умножения для всех чисел отрезка $[a;b]$ на все числа отрезка $[c;d]$.

Числа a , b , c и d являются натуральными и не превосходят 10, $a \leq b$, $c \leq d$.

Следуйте формату вывода из примера, для разделения элементов внутри строки используйте '\t' — символ табуляции. Заметьте, что левым столбцом и верхней строкой выводятся сами числа из заданных отрезков — заголовочные столбец и строка таблицы.

Пример ввода 1:

```
7
10
5
6
```

Пример вывода 1:

	5	6
7	35	42
8	40	48
9	45	54
10	50	60

Пример ввода 2:

```
5
5
6
6
```

Пример вывода 2:

	6
5	30

Пример ввода 3:

```
1
3
2
4
```

Пример вывода 3:

	2	3	4
1	2	3	4
2	4	6	8
3	6	9	12