Челлендж «Программирование анимированных объектов на языке Питон. Потешные огни»

Учимся работать с циклами

Рассмотрим конструкцию, которая позволяет повторять некоторый набор действий заданное количество раз. Такая конструкция называется циклом со счетчиком. В цикле задается счетчик, и цикл работает столько раз, сколько изменяется счетчик.

Цикл состоит из заголовка и тела.

```
for счетчик in range(A):   <-- заголовок
    оператор1
    ...                    } тело
    операторN
```

Счетчик будет последовательно перебирать значения от *0* до *A–1*

Все операторы от *1-го* до *N-го* будут выполняться *A* раз

Счетчик – это целочисленная переменная. Традиционно счетчики называют *i, j, k*.

Тело цикла – это те действия – операторы, которые будут повторяться.

Одно повторение тела цикла называется *итерацией*.

Обратите внимание, что операторы в теле цикла набраны с отступом. Отступ следует делать пробелами в количестве не менее двух штук, а не табуляцией.

Действие, следующее после цикла, набирается на том же уровне, что и сам цикл, без отступов – здесь это *операторК*.

```
for счетчик in range(A):
    оператор1
    ...
    операторN
операторК
```

Все операторы от *1-го* до *N-го* будут выполняться *A* раз.

После всех этих операторов выполнится *К*-й оператор.

Давайте изучим, как работает этот вид цикла. Будем последовательно набирать и выполнять различные программы.

Наберите в отдельном файле следующий текст и запустите его на выполнение:

```
for i in range(6):
  print("*")
```

На экране появится

```
>>>
========== RESTART: C:/!D/!Петровские асссамблеи/Фейерверки/1111.py ==========
*
*
*
*
*
*
>>>
```

Добавьте к предыдущем тексту текст, выделенный красным, и запустите программу:

```
for i in range(6):
  print("*",end="")
```

```
>>>
========== RESTART: C:/!D/!Петровские асссамблеи/Фейерверки/1111.py ==========
******
>>>
```

Замените текст и запустите программу:

```
for i in range(6):
  print(i)
```

```
>>>
========== RESTART: C:/!D/!Петровские асссамблеи/Фейерверки/1111.py ==========
0
1
2
3
4
5
>>>
```

Измените текст и запустите программу:

```
for i in range(4):
  print(i)
  print("*")
print("!!!")
```

```
>>>
========== RESTART: C:/!D/!Петровские асссамблеи/Фейерверки/1111.py ==========
0
*
1
*
2
*
3
*
!!!
>>>
```

Вторая разновидность цикла – это указание после слова *range* двух целых чисел через запятую. Первое число должно быть меньше второго, иначе цикл не отработает ни разу.

```
for счетчик in range(A, B):
  оператор1
  ...
  операторN
```

Счетчик будет последовательно перебирать значения от *A* до *B–1*.

Все операторы от *1-го* до *N-го* будут выполняться *(B–A)* раз. Если $A \geq B$, то цикл не выполнится ни разу.

Наберите в отдельном файле следующий текст и запустите его на выполнение:

```
s=0
for i in range(2, 6):
  s=s+i
print(s)
```

На экране появится

```
>>> ========== RESTART: C:/!D/!Петровские асссамблеи/Фейерверки/1111.py ==========
14
>>>
```

В этом примере подсчитывается сумма чисел от 2 до 5 – сначала переменной *s* присваивается значение нуля. Затем выполняется цикл, где счетчик меняется от 1 до 5 – в цикле повторяется действие, которое увеличивает значение переменной *s* на значение счетчика – из переменной *s* извлекается значение, складывается со значением счетчика, и опять помещается в переменную *s*. Затем содержимое переменной *s* печатается.

Замените в программе 6 на 2 (выделено красным) и запустите программу на выполнение:

```
s=0
for i in range(2, 2):
  s=s+i
print(s)
```

На экране появится

```
>>> ========== RESTART: C:/!D/!Петровские асссамблеи/Фейерверки/1111.py ==========
0
>>>
```

Здесь счетчик меняется от двух до двух – такой цикл не будет выполнен ни разу, поэтому будет выведено исходное значение переменной *s* – ноль.

Последний вариант цикла со счетчиком – задание во фразе *range* трех чисел.

```
for счетчик in range(A, B, S):
  оператор1
  ...
  операторN
```

Счетчик будет перебирать значения от *A* до *B–1*
с шагом *S*

Например, при задании *range(5, 16, 3)* счетчик будет увеличиваться на три и принимать значения 5, 8, 11, 14, а цикл будет выполнен 4 раза.

При задании *range(20, 5, –4)* счетчик будет уменьшаться на четыре и принимать значения 20, 16, 12, 8, а цикл будет выполнен тоже четыре раза.

В двух следующих примерах подсчитывается сумма значений счетчика, а потом выводится на экран.

Наберите в отдельном файле следующий текст и запустите его на выполнение:

```
s=0
for i in range(2, 8, 2):
  s=s+i
print(s)
```

На экране появится

Счетчик будет принимать значения 2, 4, 6, а сумма будет равна 12.

Измените текст и запустите его на выполнение:

```
s=0
for i in range(8, 2, -2):
  s=s+i
print(s)
```

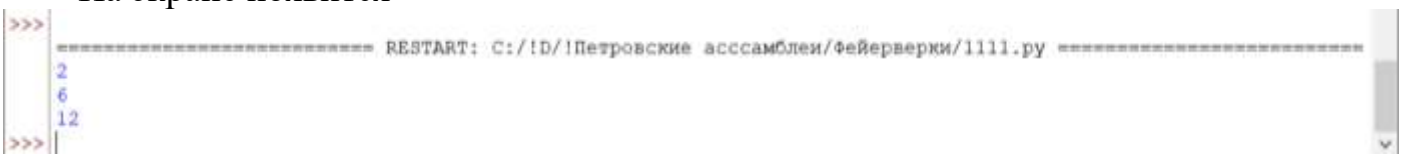
На экране появится

Обратите внимание, что здесь счетчик меняется от большего значения к меньшему с шагом –2. Счетчик принимает значения 8, 6, 4, а сумма будет равна 18. Если же указать шаг, равный 2, то цикл не отработает ни разу.

Следующие два примера отличаются от предыдущих тем, что при выполнении каждой итерации цикла не только добавляется к переменной суммы значение счетчика, но и каждое такое промежуточное значение суммы выводится на экран. Наберите в отдельном файле следующий текст и запустите его на выполнение:

```
s=0
for i in range(2, 8, 2):
  s=s+i
  print(s)
```

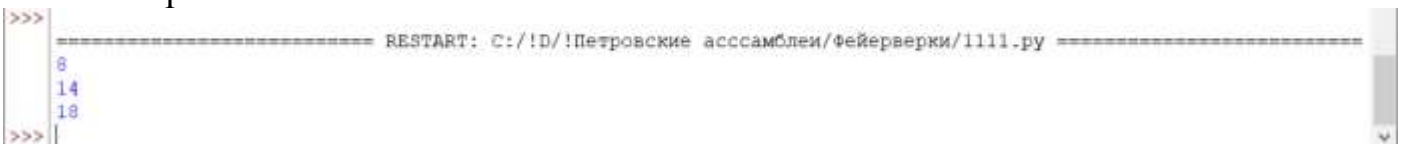
На экране появится



Измените в заголовке текст и запустите его на выполнение:

```
s=0
for i in range(8, 2, -2):
  s=s+i
  print(s)
```

На экране появится



Попробуем использовать циклы для анимации рисунков. Напомним, что в начале программы должен быть подобный текст (полностью пример приведен в файле *PA_items2.py*).

После этого создадим красный круг:

```
f1=c.create_oval((50,50),(100,100), width=0, fill='#cc3333')
time.sleep(ts)
tk.update()
```

Затем наберем следующий текст:

```
for i in range(90):
  c.move(f1, 10,5)
  time.sleep(0.1)
  tk.update()
```

После запуска программы вы увидите, что красный круг будет плавно перемещаться вправо вниз. Вы можете управлять количеством перемещений – это число 90, скоростью перемещения – это задержка в методе *sleep* (здесь она равна 0.1), плавностью перемещения – это смещения в методе *move* по оси X и оси Y (здесь это 10 и 5). Поэкспериментируйте с этими вариантами. Обратите внимание, что необходимо так рассчитать количество перемещений, чтобы объект не вылетел за пределы окна.

Теперь создадим желтый круг:

```
f2=c.create_oval((100,100),(300,300), width=0, fill='#ffff00')
time.sleep(ts)
tk.update()

for i in range(10):
  c.itemconfig(f2, fill='#ff00ff')
  time.sleep(0.2)
  tk.update()
  c.itemconfig(f2, fill='#ffff00')
  time.sleep(0.2)
  tk.update()
```

В результате работы программы вы увидите, что круг 10 раз поменяет цвет на маджентовый и снова на желтый.

И, наконец, поэкспериментируем с размером объекта. Создадим циановый квадрат:

```
f3=c.create_rectangle((500,100),(900,500), width=0, fill='#0099ff')
time.sleep(ts)
tk.update()

for i in range(5):
   c.coords(f3, 600,200,800,400)
```

```
    time.sleep(ts)
    tk.update()
    c.coords(f3, 500,100,900,500)
    time.sleep(ts)
    tk.update()
```

В результате работы программы вы увидите, что 5 раз квадрат уменьшится вдвое и снова увеличится.