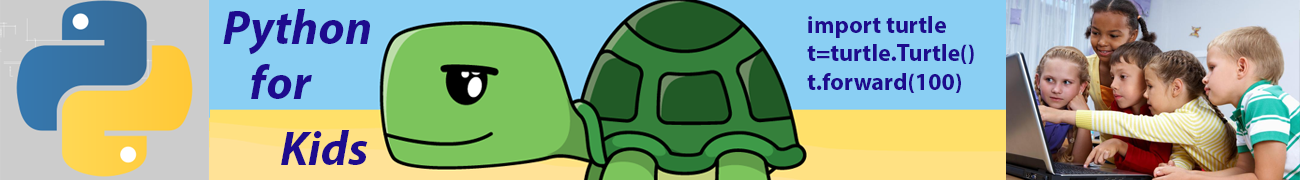
****

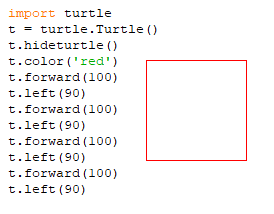
**Lesson 6: Loopy loopsin Python**

In programming, a **loop** means repeating something multiple times. One of the main Python primitive loop commands is *for* loop:

**Summary:**

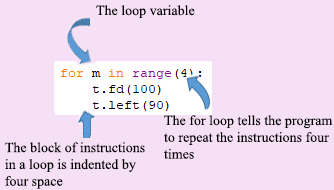
|  |  |
| --- | --- |
| **Code Instruction** | **What it does** |
| Keep in mind that command t.fd(100) is equivalent command t.fd(100),  t.forward is the same as t.fd(100)  **t.left(90) is the same as**  **t.lt(90) and t.right(90) is the same as t.rt(90)** | Here for loop is used for a block of two lines code [t.forward(100) and t.left(90)] which you want to repeat 4 times  This short program is equivalent to the following |
|  | Nested Loop, equivalent to the following block-based code. Nested loop is loop inside a loop, the outer loop only repeats (here 2 times) after inner loop has gone round its required number of times (here 4) |

Let’s learn how to write more efficient Python code by using loop structure Let’s simplify code that draws simple square. Code looks as follows:

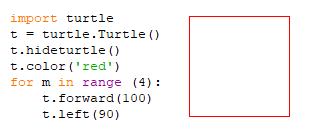


What we can see from the structure of this code. The program above gives the turtle the same commands **t.forward (100)** and **t.left(90)** four times – once for each side of the square. An alternative to this is to use these two commands inside a *for* loop. Whenever we need to do something over and over again in a program, loops allow us to repeat those steps without having to type each one separately.

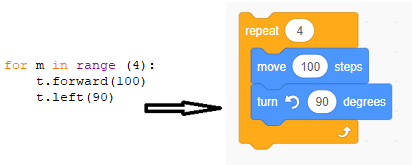
To build our own loop, we first need to identify the repeated steps. The instructions that we’re repeating in the preceding code are t.forward(100) to draw a turtle line with a length of 100 pixels and t.left(90) to turn the turtle left 90 degrees before drawing the next circle. Second, we need to figure out how many times to repeat those steps. We want four circles to build a square, so let’s start with four.



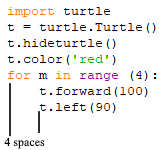
We start with the keyword ***for*** and then we give an integer number ***m*** that will be our counter. Here variable ***m*** (integer number) is something you want the computer to remember while your program is running. We use a name of this variable as an ***m,*** however it could be any name as you want, for example, ***a, b, x, y*** …Name of the variable does not matter, the key point that this variable ***m*** successively takes the following values: first step value ***m*** equal 0, second step ***m***=1, third step ***m***=2, ad fourth step ***m***=3. For each step computer draws line t.forward(100) and turn turtle by 90 degrees t.left(90). So, code with a loop is as following



This *for* loop is equivalent to the block-based code shown below



**Keep in mind that there are four additional spaces at the beginning lines of the loop block as shown below (when you compare with the line** ). **Code that is not indented, which comes after the loop, will be run once just like regular code (Example #4).**



**Python editor will have automatically indented this for you.**

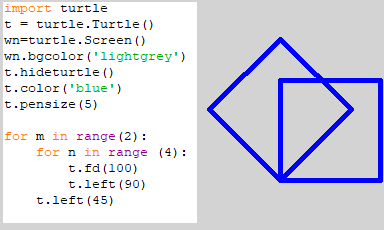
Loop variable (in our code loop variable named as ***m*** counts the number of times a loop has repeated itself. It starts at the first value in the range (0) and stops one before the last value (in our loop-> 3).

Nested Loops

Can the body of a loop have another loop within it? Yes!! This is called a nested loop. It’s like Russian dolls, where each doll fits inside a larger doll. In a nested loop, an inner loop runs inside an outer loop.



Let's give an example. We will draw several squares, each of which is rotated 45 degrees from the previous one.



In this example the inner loop (variable ***n*** goes around 4 times (we draw square). Code t.left(45) is a body of outer loop (variable ***n***) and turns turtle by 45 degrees, and then we again go to the inner loop. Let’s detail an operation of the code.

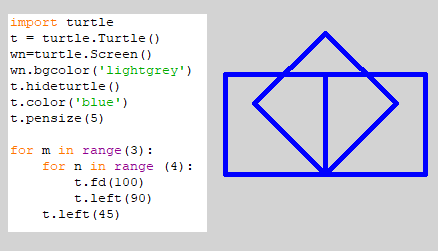
Step #1: The first time through the outer loop, counter ***m*** has a starting value of 0, turtle goes to the inner loop, variable ***n*** takes value equal 0, turtle draws first line 100 px, then turns left by 90 degrees.

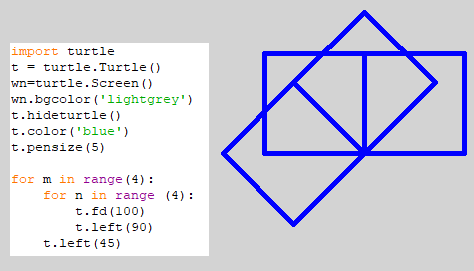
Step #2 Python goes back to the beginning of the inner loop and sets ***n*** to 1. Then it draws second line and turns left by 90 degrees.

Step#3 Python goes back through the loop again, increasing ***n*** to 2. It draws third line and turn by 90 degrees.

Step #4 Python goes back through the loop again, increasing ***n*** to 2. It draws third line and turn by 90 degrees.

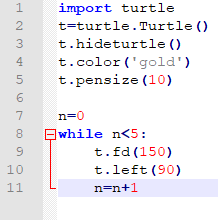
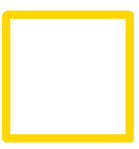
Step#5 Python goes out of inner loop, turns turtle to 45 degrees, go back to the beginning of the outer loop and sets outer variable ***m*** to 1. Then Python enters to the inner loop and repeats all steps #1 to #4. Because, outer variable changes in the range from 0 to 1Pyrho draws only 2 squares. Let’s see what will happen, if outer loop looks like: or 





*While* loops

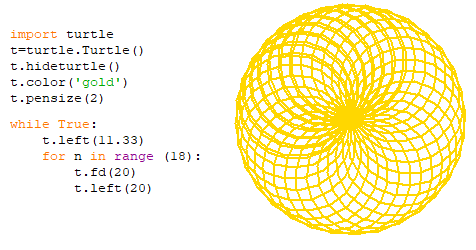
For loops are useful when you know how many times a task needs to be repeated. But sometimes you need a loop to keep repeating until something changes. A while loop keeps on going around as many times as it needs to. A *while* loop keeps repeating as long as certain condition is true. This condition is called the loop condition and is either true or false.

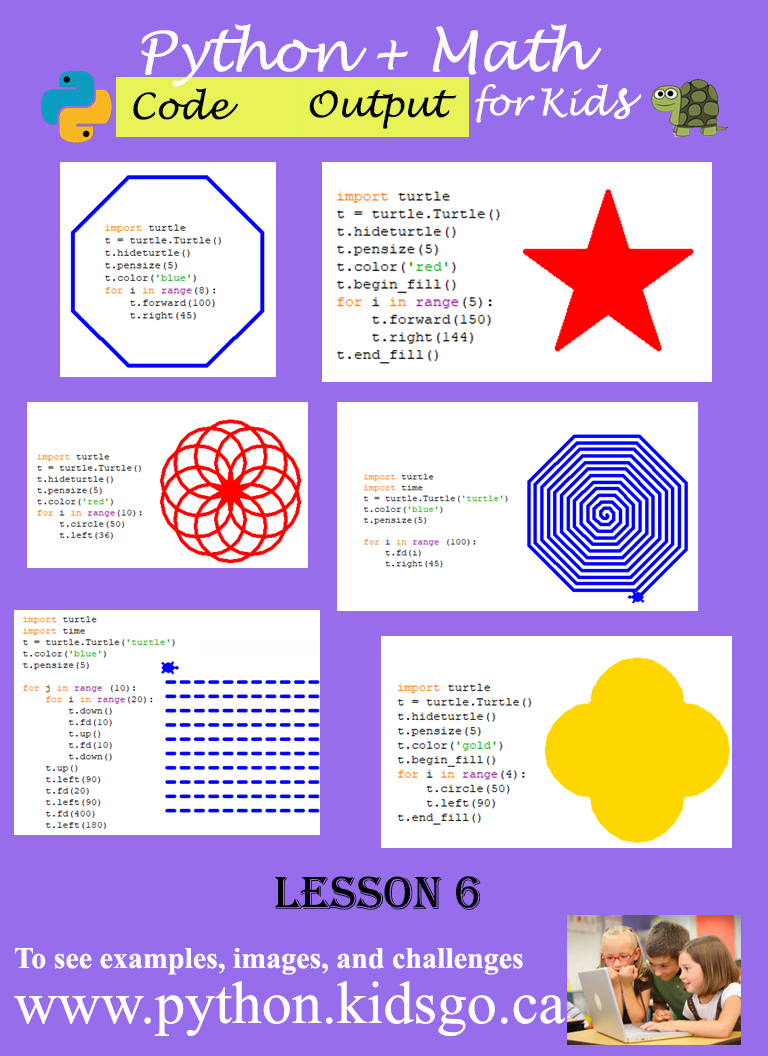
 ****

Code (Example #6) creates square using *while* loop. We start by creating a variable (line 7) named n which is set to 0. Code line 8: While loop checks for condition n<5. The current value of n is 0. Condition is true. Flow of control enters into while loop. Turtle draws horizontal line and turns heading by 90 degrees. Code line 11: n is incremented by1. Flow of control goes back to line 8. Now the value of n is 1 which is less then 5. The condition is true, and again the while loop is executed. This continues till n becomes 5, and the while condition becomes false. Execution of this code is completed.

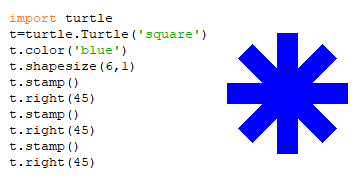
*while* *True*: This loop runs forever. If you set the condition in a while loop to be True, it can never be false and the loop will never end. In some situations, it can be very useful.

This program will never end

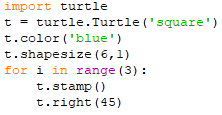
****

****

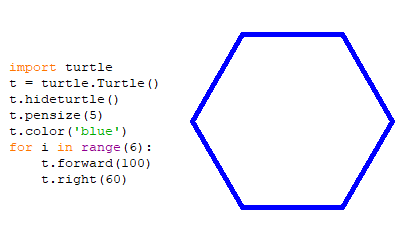
1. **Example #1** (Draw snowflake without and with loop)
2. (No loop)



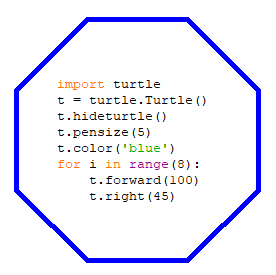
1. (With loop)



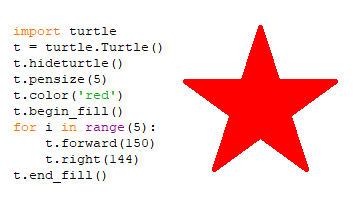
1. **Example #2** (Hexagon)



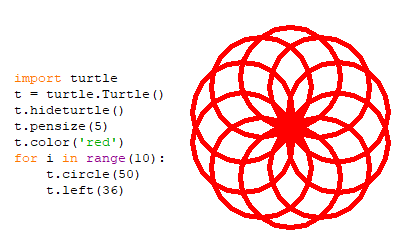
1. **Example #3** (Octagon)

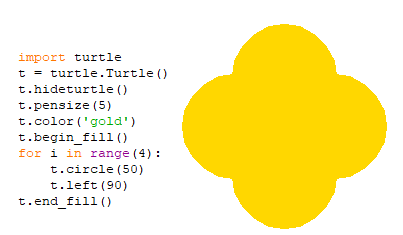


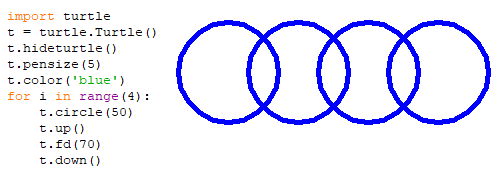
1. **Example #4** (Star)

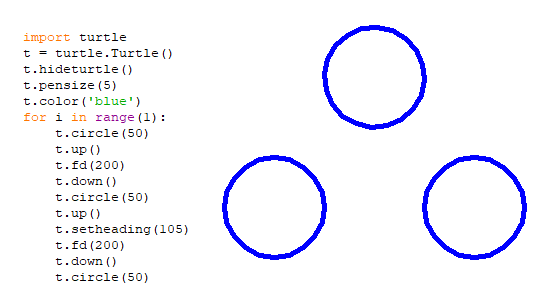


1. **Example #5** (Number of circles)

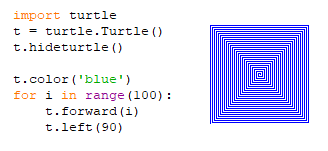






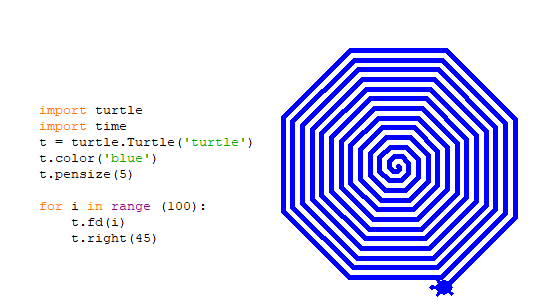


1. **Example #6** (Square Spiral)

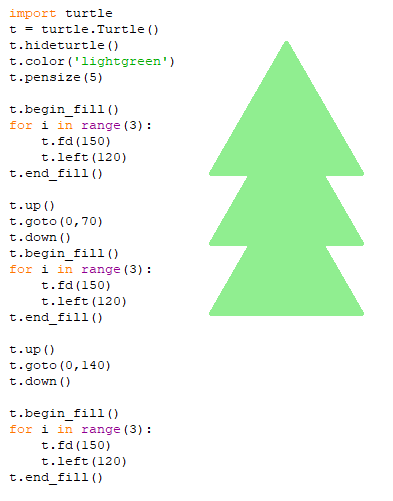


This particular loop sets up a range of 100 numbers from 0 to 99. (0,1,2,3,4... 99 totally 100 values). Because each time through the loop, our counter i changes value by 1, a length of the line also increases by 1

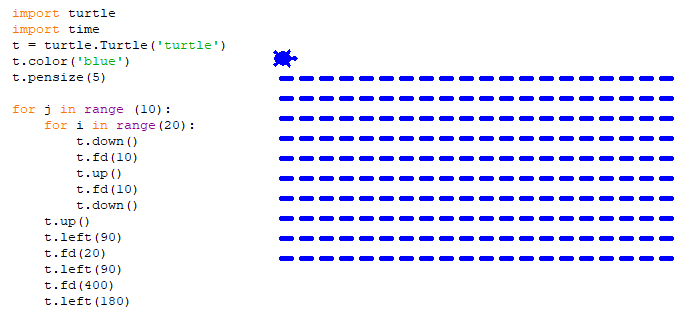
1. **Example #8** (Spiral)



1. **Example #7** (Christmas tree)

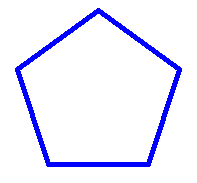


1. **Example #8** (Dashed lines with nested loop)

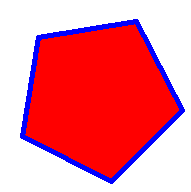


Challenges: write codes to create the following geometry shapes with circle code:

1. Expected output



1. Expected output



1. Expected output

