**For Loops**

A ***loop*** is a way of repeating a section of code. A ***for*** ***loop*** is one kind of loop.

Try this:

for x in range (3):

print ("hello")

This will print the word hello three times.

Just as with IF statements, FOR loops require ***indenting*** to determine what gets repeated. Try these two examples:

for x in range (3):

print ("hello")

print ("goodbye") # indented

and

for x in range (3):

print ("hello")

print ("goodbye") #not indented

In the second example, the *goodbye* statement is only printed once, because it is not part of the FOR loop.

**Counting with For Loops**

The variable in our examples is x, but you can use any variable. By convention, programmers use the variable i because it is an ***index***. We will see more about the uses of the index in future chapters. We can print the index:

for i in range (3):

print (i) # prints the index

The code above allows us to see the index counting each loop. Note that the index starts at a value of zero.

Because the variable in a for loop counts, we sometimes call it a **counter**. In the above example, the variable i is a counter.

Counting is something computers are very good at:

print ("Count to 10: ")

for i in range (10):

print (i+1) # why i+1 ?

print ("goodbye! ")

Do you see why we have to add 1?

There are several versions of the range function. Try this:

for i in range (6,15):

print (i)

This example starts at 6 and stops at 15. Because it stops at 15, it doesn’t print the 15. It only prints from 6 to 14.

Basically this means “count from 6 up to 15 but not including 15.”

**Stepping Forward and Backward**

The range function can even take a third parameter. What does this do?

print ("What does this do?")

for i in range (10,100, 4):

print (i)

print ("goodbye! ")

The third parameter (4) in the range statement tells the loop to count in steps of 4. This value is called the ***step*** of the loop.

We can make the loop step backwards too!

print ("What does this do?")

for i in range (100,1, -4): # count down in steps of 4

print (i)

print ("goodbye! ")

**The break and continue statements**

You can quit early from a loop if you wish, by using a ***break*** statement. The break statement is usually part of an IF statement. Here is an example:

for i in range (1,100, 4):

print (i, " ",end="")

if i>50:

break

print ("goodbye! ")

In this case, the execution breaks out of the loop once it gets past 50.

You can also skip the remaining part of the statements in an iteration using the ***continue*** statement. Here is an example. Can you spot what happened?

for i in range (1,10):

if i==5:

continue # skip this iteration

print (i, " ", end="")

print ("goodbye! ")

The continue statement is used only in the special case where you want to skip an iteration.

Exercises:

1. What does the following code do? Predict before writing it out.

a)

for i in range (5,10):

print ("hello", i)

b)

for i in range (0,11, 2):

print (i, end="")

c)

for i in range (10):

print (i,end="")

d)

for i in range (4):

print ("hello")

1. Each of these programs has a pattern that can be written using a loop. Rewrite each of them using a for loop.

print("\*")

print("\*")

print("\*")

print("\*")

print("1")

print("2")

print("3")

print("4")

print("5")

print("a a a a a")

print("4 7 10 13")

print("a")

print("b")

print("a")

print("b")

print("a")

print("b")

print("a a a b a a a")

print("20 15 10 5 0")

1. Create a program that asks the user to guess a number between one and ten. Give the user 3 tries to get it right. If the user gets it right, congratulate them and quit the loop. If the user gets it wrong, inform them that they got it wrong and ask them to try again.
2. Ask the user to enter a number. Inform the user that the number is positive or negative, odd or even. Repeat 100 times unless the user enters a zero, in which case the program ends.

Key Terms: ***loop, for loop, iteration, index, step, break, continue***