**The While Loop**

In Python, as in most programming languages, there are two kinds of loops. We have seen the ***for*** loop. The **for loop is designed to loop a specific number of times**.

Now we look at the ***while*** loop. A **while loop will continue looping while (as long as) a condition is still true**. Here’s an example:

age = int(input ("Enter your age:"))

while age <= 0: # check for negative age

print ("Error! Age must be more than 0! ")

age = int(input("Enter your age:")) # let user try again

print ("thank you!") # this is not part of loop

Try entering a negative value. This program will keep looping as long as the user enters an age less than or equal to zero.

Actually, there are two ways to look at it. This loop:

* keeps asking for age WHILE user enters a NEGATIVE number
* which is the same as saying it -
* keeps asking for age UNTIL user enters a POSITIVE number

Notice that only the indented lines get repeated. The print statement at the bottom only gets executed after the while loop is completed.

What will this code do?

x = int(input("What is 3 + 4 ?"))

while x != 7: # while x is NOT equal to 7

print ("Try Again!")

x = int(input("What is 3 + 4 ?"))

print ("You are correct!")

The above example will loop WHILE x \_\_\_\_\_\_\_\_\_\_\_\_\_\_ seven.

The above example will loop UNTIL x \_\_\_\_\_\_\_\_\_\_\_\_\_\_ seven.

A while loop may not loop at all:

x = 20

while x == 5:

print ("x is 5!")

print ("Good bye")

This just prints goodbye. Why does it never loop?

**Assignment Statements vs Conditional Statements**

Just a reminder that the statement:

x = 5

is an ***assignment statement***. It assigns the variable x a value of 5. It means that “x is five”. But this:

x == 5

is a ***conditional statement***. It simply checks whether x equals 5. It asks “*IS x five*?”. Conditional statements go inside IF statements and WHILE loops.

To see the difference between an assignment statement and a conditional statement, try this in the console (black window):

>>> x = 5

>>> x == 5

The first is an assignment statement, the second is a conditional statement. What is the result? The first line assigns x a value of 5. The second line asks if x is 5. Now try this:

>>> x == 7

Forgetting to put the “double-equals” sign in conditional statements is a common mistake in programming.

**Counting**

A for loop is ideal for counting. Here is an example:

for i in range(10):

print (i)

This counts from 0 to 9

While loops can also be made to count, but we must create a counter variable, and make it go up by 1. Consider this code:

x = 5

print (x) # what is x now?

x = x + 1

print (x) # what is x now?

x = x + 1

print (x) # what is x now?

The output to this code is

5

6

7

What is the meaning of this line?

x = x + 1

Suppose x is already 5. That means x + 1 is 6. So x is assigned a new value of 6.

Now suppose x is 6. That means that x + 1 is 7. So x is assigned a value of 7.

And so on.

The statement x = x + 1 is a way to ***increment*** the value of x by 1.

You can also ***decrement*** by 1:

x = x – 1

**Counting in a Loop**

What does this do:

x = 4

while x < 10:

print ("x is", x)

x = x + 1

print ("Good bye")

What happens when we change the increment:

x = 4

while x < 10:

print ("x is", x)

x = x + 2 # notice the change?

print ("Good bye")

This outputs 4, 6, 8. Why does this not print 10?

**Looping Forever**

A while loop continues looping as long as the condition is true. Consider this:

x = 4

while x == 4:

print ("x is", x)

print ("Good bye")

What does this do? Why?

Sometimes you want a loop to go forever. In that case, you just need to put a condition that is always true. The one thing that is definitely always true is True. True is true. No?

while True:

print ("hello!")

print ("Good bye")

This is a good trick if you want code to loop indefinitely.

It is always good to have a way out of such a loop. The best way is to have a break statement in the loop, to get you out when you need to.

Exercises

9. Create a while loop that produces exactly the following output:

a) 1 2 3 4 5 6 7

b) 0 5 10 15 20

c) 100 90 80 70 60 50 40 30 20 10 0

d) hello 1 hello 3 hello 5 hello