Chapter 30

WHILE LOOPS AND NESTED LOOPS

WHILE LOOPS

WHILE LOOPS repeat as long as the conditional statements within them are True. A *while* loop looks similar to a *for* loop, but it replaces the counting portion with a CONDITIONAL STATEMENT.

> **REMINDER:**
> A conditional statement runs a chunk of code only when a certain condition is met.

While loops always start with the key word "while" followed by a Boolean expression and then a colon (:). The repeated code is indented below the first line.

You could make a password checker that will continue to loop until the user types in the correct password. The Boolean expression checks if the entered password is true. To do this:

Create the variable "password" and assign it the value **None**.

"None" is a Python key word that means "empty."

Create the variable "password" before you use it in the *while* loop.

```
password = None
while password != "myPassword1234":
    password = input("Enter the password: ")
    if password != "myPassword1234":
        print("Your password is incorrect.")
print("Correct password. Welcome.")
```

This *while* loop will continually loop as long as the password entered is NOT the same as "myPassword1234".

This condition will only run the print( ) function if the password variable value is not equal to "myPassword1234".

The print( ) function will run after the password variable value DOES equal "myPassword1234".

Example output:

```
Enter the password: rememberMe
Your password is incorrect.
Enter the password: CantRemember
Your password is incorrect.
Enter the password: OhNowIDo
Your password is incorrect.
Enter the password: myPassword1234
Correct password. Welcome.
```

The program will continue the loop and prompt the user to enter the password until the user enters "myPassword1234". When that happens, the loop ends and the rest of the program (printing "Correct password. Welcome.") is allowed to run.

Infinite Loops

In Python, **INFINITE** loops (forever loops) are *while* loops that use a Boolean statement that can never become false.

These are all infinite loops because there is no way for the Boolean statements in each to become false:

```
while True:
    print("This is the song that never ends.")

while 4 > 3:
    print("This is the song that never ends.")
```

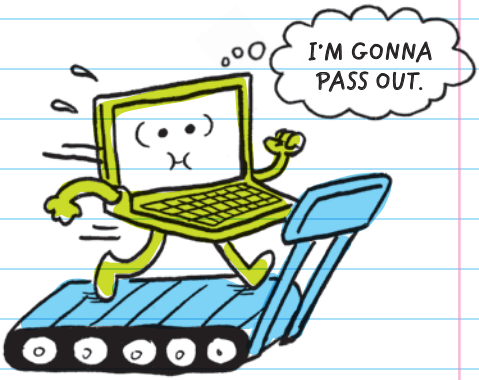
```
while "hello" == "hello":
    print("This is the song that never ends.")

while 5 <= 5:
    print("This is the song that never ends.")
```

None of these loops provide a way for the conditional statement to ever become false, so these loops will continue to loop forever (infinitely). The print( ) statement in each example will continuously loop and print over and over again, forever.

> You can exit an infinite loop by pressing Ctrl + C on Windows, or command ⌘ + C on a Mac.

Sometimes you write an infinite loop as part of your program on purpose. For example, video games usually use an infinite loop to animate the characters, continually updating character movement and interaction as the player plays the game. But other times, an infinite loop may be written by accident, and it may crash your



computer because the program is too large or tries to get the computer to process too much information.

NESTED LOOPS

A **NESTED LOOP** is when one loop is put inside another loop. They help create more complex repeating code. For example, you can use a password program as an outer *while* loop and nest an inner *for* loop to print out all the wrong guesses the user inputs before they guess the correct password:

```
password = None
attempts = []
while password != "myPassword1234":
    password = input("Enter the password:")
    attempts.append(password)
    if password != "myPassword1234":
        print("Your password is incorrect. You have already
        guessed:")
        for i in attempts:
            print(i)
print("Correct password. Welcome.")
```

Creates a new list type variable and assigns it an empty list value

Adds the user's input to the attempts list

This *for* loop prints out all the items in the attempts list (the password attempts the user has entered).

There is an additional indent for the print() function to show that this code is inside the nested *for* loop, not the outer *while* loop.

CHECK YOUR KNOWLEDGE

1. What's the difference between a *while* loop and a *for* loop?

2. A *while* loop that uses a conditional statement that will always be true is called _____.

3. What keys do you press to get out of an infinite loop?

4. Describe how a tab (or 4 spaces) is used in nested loops.

5. Write a loop that will give the following results.

PROGRAM NAME	PROGRAM	RESULT
A		The program will print: Go! Go! Go! as long as the variable "x" is more than 50.
B		The program will print: Good morning, Steve as long as the variable "name" is equal to "Steve".

PROGRAM NAME	PROGRAM	RESULT
C		The program will count from 7 to 11 by twos 3 times.
D		The program will count to 5 over and over again, forever.
E		The program will print: hip hip hooray hip hip hooray hip hip hooray
F		The program will print: 1 2 3 4 5
G		The program will print: hello, friend . . . forever, or until the program is stopped using Ctrl + C.

CHECK YOUR ANSWERS

1. A *for* loop runs a set amount of times, and a *while* loop will run as long as its condition is True.

2. An infinite, or forever, loop

3. Ctrl + C on Windows or command ⌘ + C on a Mac

4. Tabs (or 4 spaces) are used to show which loop is inside the other. An additional tab or 4 spaces shows that the code is nested inside another inner loop.

5. Program answers:

   A.
   ```
   while x > 50:
       print("Go! Go! Go!")
   ```

   B.
   ```
   while name == "Steve":
       print("Good morning, Steve")
   ```

   C.
   ```
   for i in range(3):
       for j in range(7, 12, 2):
           print(j)
   ```

   Variable names may be different.

D.
```
count = True
while count:
    for i in range(1, 6, 1):
        print(i)
```

Variable names may be different.

E.
```
for i in range(3):
    for j in range(2):
        print("hip", end=" ")
    print("hooray")
```

Variable names may be different.

F.
```
num = 0
while num < 5:
    num = num + 1
    print(num)
```

Variable name may be different.

G.
```
while True:
    print("hello, friend")
```


999, 1000 LOOPS!
YOU KEEP READING AND
I'LL KEEP RUNNING!