Introduction to Python

Indirect Loops: While Loops

Indefinite Loop

A **for** loop, we discussed earlier is an example of a **definite loop**, the number of iterations can be specified ahead of time by the programmer.

In some cases, however, the number of iterations can be unknown. This type of loop is called an **indefinite loop.** For example, a user is asked to enter a set of inputs. The number of inputs the user enter is not known in advance. The program's input loop accepts these values until the user enters a special value or **sentinel** that terminates the input.

In this section, we explore the use of the **while** loop to describe conditional iteration: iteration that repeats as long as a condition is true.

While Loop

The **while loop** has the syntax:

```
while <condition>:
      block
```

The loop **r**epeatedly executes its block of code as long as the condition is true.

At least one statement in the block of the loop must update a variable that affects the value of the condition. Otherwise, the loop will continue forever, an error known as an **infinite loop**.

While vs For Loops 1

The following two segments of code are equivalent.

```
# Counting down with a for loop from 10 to 1.
for count in range(10, 0, -1):
      print(count, end = " ")


# Counting down with a while loop from 10 to 1.
count = 10
while count >= 1:
      print(count, end = " ")
      count -= 1
```

While vs For Loops 2

The following two segments of code are equivalent.

```
# Summation with a for loop
theSum = 0
for count in range(1, 101):
      theSum += count
print(theSum)
# Summation with a while loop
theSum = 0
count = 1
while count <= 100:
      theSum += count
      count += 1
print(theSum)
```

While Loop

Here's an example of a loop where the number of iterations is unknown in advance. Suppose a program computes a sum of a set of inputs. We don't know when the user finishes entering the inputs. Instead of a for loop, a while loop is more appropriate.

```
s = 0.0
data = input("Enter a number or just enter to quit: ")
while data != "":
    number = float(data)
    s += number
    data = input("Enter a number or just enter to quit: ")
print("The sum is", s)
```

While Loop

```
s = 0.0
data = input("Enter a number or just enter to quit: ")
while data != "":
    number = float(data)
    s += number
    data = input("Enter a number or just enter to quit: ")
print("The sum is", s)
```

Sample Run:

Enter a number or just enter to quit: 3

Enter a number or just enter to quit: 4

Enter a number or just enter to quit: 5

Enter a number or just enter to quit:

The sum is 12.0

While Loop

Alternatively, the previous program can be simplified so that only one input is used. The program uses the **break** statement to exit the loop if the input is an empty string.

```
s = 0.0
while True:
    data = input("Enter a number or just enter to quit: ")
    if data == "":
        break
    number = float(data)
    s += number
print("The sum is", s)
```

While Loop

As another example of an indefinite loop, suppose the user is asked to enter a numeric grade. The program will keep asking the user until he enters a value in the appropriate range(0 – 100).

```
while True:
      number = int(input("Enter the numeric grade: "))
      if number >= 0 and number <= 100:
            break
      else:
            print("Error: grade must be between 100 and 0")
print(number)    # Just echo the valid input
```

While Loop

```
while True:
      number = int(input("Enter the numeric grade: "))
      if number >= 0 and number <= 100:
            break
      else:
            print("Error: grade must be between 100 and 0")
print(number)   # Just echo the valid input
```

Enter the numeric grade: **101**

Error: grade must be between 100 and 0

Enter the numeric grade: **–1**

Error: grade must be between 100 and 0

Enter the numeric grade: **45**

45

Random Numbers

In some situation, we like to be able to simulate randomness. For example, we might toss a coin or roll a die.

The Python's random module contains many functions to do this. The function **randrange()** is easy to use since it is similar to the **range()** function we used in for loops.

randrange(start, stop, step) generates a random integer beginning with start(including) and ending with stop(not including) with step.

```
for i in range(10):
    num = random.randrange(1, 5)
    print(num, end=" ")
```

Output:

3 3 3 2 1 3 2 2 3 4

Lab 1

In this lab, we will write a simple guessing game.

The computer randomly generate a number in some given range. On each pass through the loop, the user enters a number to attempt to guess the number selected by the computer.

The program responds by saying "You've got it," "Too large, try again," or "Too small, try again." When the user finally guesses the correct number, the program congratulates him and tells him the total number of guesses.

Let's look at a sample run. What's the strategy for guessing the number?

Guessing Game

Enter the smaller number: 1

Enter the larger number: 100

Enter your guess: 50

Too small!

Enter your guess: 75

Too large!

Enter your guess: 63

Too small!

Enter your guess: 69

Too large!

Enter your guess: 66 Too large!

Enter your guess: 65

You've got it in 6 tries!

References

1) Vanderplas, Jake, A Whirlwind Tour of Python, O'reilly Media.