

Part 04 - Flow Control - Loops

Part 04 - Flow Control - Loops

For Loop

i Definition: For loop

A loop whose body gets obeyed once for each item in a sequence.

A `for` loop in Boo is not like the `for` loop in languages like C and C#. It is more similar to a `foreach`.

The most common usage for a `for` loop is in conjunction with the `range` function.

The `range` function creates an enumerator which yields numbers.

The `join` function in this case, will create a string from an enumerator.

join and range example

```
join(range(5))  
join(range(3, 7))  
join(range(0, 10, 2))
```

Output

```
0 1 2 3 4  
3 4 5 6  
0 2 4 6 8
```

`range` can be called 3 ways:

```
range(end)
```

```
range(start, end)
```

```
range(start, end, step)
```

To be used in a `for` loop is quite easy.

for loop

```
for i in range(5):  
    print i
```

Output

```
0  
1  
2  
3  
4
```

i Practically as fast as C#'s

The `range` function does not create an array holding all the values called, instead it is an `IEnumerator`, that will quickly generate the numbers you need.

While Loop

i Definition: While loop

A structure in a computer program that allows a sequence of instructions to be repeated while some condition remains true.

The `while` loop is very similar to an `if` statement, except that it will repeat itself as long as its condition is true.

while loop

```
i = 0  
while i < 5:  
    print i  
    i += 1
```

Output

0
1
2
3
4

In case you didn't guess, `i += 1` adds 1 to `i`.

Continue Keyword

i Continue keyword

A keyword used to resume program execution at the end of the current loop.

The `continue` keyword is used when looping. It will cause the position of the code to return to the start of the loop (as long as the condition still holds).

continue statement

```
for i in range(10):  
    continue if i % 2 == 0  
    print i
```

Output

1
3
5
7
9

This skips the `print` part of this loop whenever `i` is even, causing only the odds to be printed out. The `i % 2` actually takes the remainder of `i / 2`, and checks it against 0.

While-Break-Unless Loop

the `while-break-unless` loop is very similar to other languages `do-while` statement.

while-break-unless loop

```
i = 10
while true:
    print i
    i -= 1
    break unless i < 10 and i > 5
```

Output

```
10
9
8
7
6
5
```

Normally, this would be a simple `while` loop.

This is a good method of doing things if you want to accomplish something at least once or have the loop set itself up.

Pass Keyword

The `pass` keyword is useful if you don't want to accomplish anything when defining a code block.

while-break-unless loop

```
while true:  
    pass //Wait for keyboard interrupt  
    (ctrl-C) to close program.
```

Exercises

1. print out all the numbers from 10 to 1.
2. print out all the squares from 1 to 100.

Go on to [Part 05 - Containers and Casting](#)