

while Loops

CSCI20


Mark Kazakevich

while loops

- A **while** loop is a statement that allows us to repeat code while some condition is **True**
- The number of times it repeats depends on this condition, which must be **True** for the loop to continue iterating
 - When the condition is **False**, the loop ends
- Let's take a closer look

while loop Format

```
while condition:  
    # loop body
```



This block is
considered one
while loop

Let's talk about what these words all mean

while loop Format

```
while condition:  
    # loop body
```

`while`

Indicates that this is a `while` loop statement

while loop Format

```
while condition:  
    # loop body
```

condition

condition is an expression that evaluates to a **boolean value**.

We continue executing the loop as long as condition is True.
“While condition is True, keep repeating the loop.”

while loop Format

```
while condition:  
    # loop body
```

loop body

- These lines of code (which are indented in the for loop), will repeat as long as condition is True.
- Unlike a for loop, there is no variable that changes at every iteration of the loop.
- But, we can change variables involved in the condition

Let's see an example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

Let's see an example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

while

Indicates that this is a while loop statement

Let's see an example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

$n > 0$

We check if $n > 0$ evaluates to **True**.

If it does, then we run the loop body.

Notice that n was defined **before** the loop.

Let's see an example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

In the loop body, we output the value of `n` to the shell.

Let's see an example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

In the loop body, we output the value of `n` to the shell. Notice how we also decrement the value of `n` by 1. By subtracting `n` by 1, we are changing a variable used in the `while` loop condition.

Let's see an example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

Increment: Increase the value of a numeric variable

Decrement: Decrease the value of a numeric variable

In the loop body, we output the value of `n` to the shell. Notice how we also decrement the value of `n` by 1. By subtracting `n` by 1, we are changing a variable used in the `while` loop condition.

Running the example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

notice we decrement n by 1 at the end of the loop

Python shell output after running loop body:

1st iteration of loop:

Current value of n is 3

n > 0 is True

So we run the loop body

3

Running the example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

Python shell output after
running loop body:

2nd iteration of loop:

Current value of n is **2**

n > 0 is True

So we run the loop body

3

2

Running the example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

Python shell output after
running loop body:

3rd iteration of loop:
Current value of n is 1
n > 0 is True
So we run the loop body

```
3
2
1
```

Running the example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

Python shell:

4th iteration of loop:

Current value of n is **0**

n > 0 is **False**

The while condition is no longer true - we do not run the loop body

```
3
2
1
```


Running the example

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```

-
-

```
# program continues
```

-
-
-

The `while` condition is no longer true.

We're done! We now move on to the statements after the `while` loop

```
3
2
1
```

Something to be careful about

```
n = 3
while n > 0:
    print(n)
    n = n + 1
```

If we always add 1, we will never fail the condition, so the loop will keep going forever

- **Be careful** with what what you do to variables involved in your while condition.
- Assigning the wrong thing can lead to an incorrect number of iterations, or..
- **Infinite loops**

Examples in Wing