

# THE WHILE STATEMENT

The `while` statement allows you to repeatedly execute a block of statements as long as a condition is true. A `while` statement is an example of what is called a **looping** statement. A `while` statement can have an optional `else` clause.

Example (save as `while.py`):

```
number = 23
running = True

while running:

    guess = int(raw_input('Enter an integer : '))

    if guess == number:

        print 'Congratulations, you guessed it.'

        # this causes the while loop to stop

        running = False

    elif guess < number:

        print 'No, it is a little higher than that.'
```

```
else:
```

```
    print 'No, it is a little lower than that.'
```

```
else:
```

```
    print 'The while loop is over.'
```

```
    # Do anything else you want to do here
```

```
print 'Done'
```

Output:

```
$ python while.py
```

```
Enter an integer : 50
```

```
No, it is a little lower than that.
```

```
Enter an integer : 22
```

```
No, it is a little higher than that.
```

```
Enter an integer : 23
```

```
Congratulations, you guessed it.
```

```
The while loop is over.
```

```
Done
```

## *How It Works*

In this program, we are still playing the guessing game, but the advantage is that the user is allowed to keep guessing until he guesses correctly - there is no need to repeatedly run the program for each guess, as we have done in the previous section. This aptly demonstrates the use of the `while` statement.

We move the `raw_input` and `if` statements to inside the `while` loop and set the variable `running` to `True` before the `while` loop. First, we check if the variable `running` is `True` and then proceed to execute the corresponding **while-block**. After this block is executed, the condition is again checked which in this case is the `running` variable. If it is true, we execute the `while-block` again, else we continue to execute the optional `else-block` and then continue to the next statement.

The `else` block is executed when the `while` loop condition becomes `False` - this may even be the first time that the condition is checked. If there is an `else` clause for a `while` loop, it is always executed unless you break out of the loop with a `break` statement.

The `True` and `False` are called Boolean types and you can consider them to be equivalent to the value `1` and `0` respectively.

Source: <http://www.swaroopch.com/notes/python/>