While Loops & User Input

While Loops

Format:

```
while condition:
    statement(s)
```

The condition will be checked before each execution of the statements. If the condition is `True`, the statements will be run, and then the condition will be checked again. If the condition is `False`, the statements will be skipped, and the loop is over.

Example:

```
def IsSquare(n)
# Is the number 'n' a perfect square? (assume n >= 0)
    r = 0
    while r * r < n:
        r += 1
    return r * r == n
```

With while loops you can end up in an infinite loop quite easily, so you have to be careful about your conditions.

Break

You can use the `break` statement to terminate a loop from within:

```
while True:
    n += 5
    if n > 10:
        break
```

The `break` statement in this loop will terminate the loop when n becomes greater than 10.

Continue

The `continue` statement is similar to `break`, but only terminates the current run of the loop:

```
n = 0
for n in range(10):
    if n = 5:
        continue
    print(n)
```

This code will begin printing the numbers from 0 to 9, but once it gets to 5, the `continue` statement will cause it to move on to the next iteration, so it will not print 5.

User Input

To take input from the user you can use the in-built function `input()`:

```
name = input('Please enter your name: ')
```

The program will wait until the user hits enter, and the line they typed will be put into the variable name, as a string.

- To get a number from the user you will need to convert the input:

  `number = int(input('Please enter an integer: '))`

  `number = float(input('Please enter a value: '))`

Handouts & Assignments

- Assignment 9 - The 'while' Statement
- Assignment 11 - Simulating Dice Rolls
- Handout 11 - The 'while' Statement
- Handout 13 - Keyboard Input and the 'break' Statement