Loops
The for statement

The general form of a for statement is:

```
for <var> in <some kind of series>:
```

The easiest way to explain this is with an example:

```
for i in [1, 2, 3]:
    print(i)
```

Produces the output:

```
1
2
3
```

- The thing in [ ] is called a list
- The number of times you go through the loop = the number of items in the list
- Each time you go through the loop, you assign the value of the next item to the variable in the for statement
- Don’t forget the colon
- Indentation is important!
The **range** function

`range(<number>)` produces a sequence of ints counting from zero up to `<number>`-1.

For example, `range(5)` gives you the sequence 0, 1, 2, 3, 4.

Example:

```python
for number in range(3):
    print(number)
```

produces the output:

```
0
1
2
```
The range function: more features

```python
range(<num1, num2>)  produces a sequence of ints counting from
<num1> up to <num2>-1.
```

For example, `range(3, 7)` gives you the sequence 3, 4, 5, 6.

Finally, `range(<num1, num2, num3>)` produces a sequence of ints counting from `<num1>` up to `<num2>-1` counting by `num3`'s.

For example, `range(4, 36, 5)` gives you the sequence 4, 9, 14, 19, 24, 29, 34.
Quick check:

What is the output?

def main():
    for i in [2, 4, 6]:
        print(i)
main()

What is the output?

def main():
    for i in range(3,22,3):
        print(i+1)
main()
Another way to iterate: the `while` statement

The general form of a `while` statement is:

```python
while <boolean expression>:
```

Again, the easiest way to explain this is with an example:

```python
powerOf2 = 1
while (powerOf2 < 100):
    print(powerOf2)
    powerOf2 = powerOf2 * 2
```

Produces the output:

```
1
2
4
8
16
32
64
```