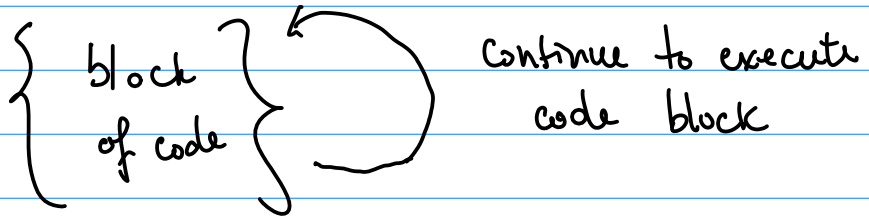
Loops:

- for loops
- while loops



For for loops:

SYNTAX ~

```
for var = some_array
    (block of code)
end
```

Ex:

```
for k = 1:3
    display("ho")
end
```

ho
ho
ho

Same equivalent code

```
k=1;
display("ho")
k=2;
display("ho")
k=3;
display("ho")
```

Ex:

```
total = 0
for k=1:100
  total = total+k;
end
total
```

What does it do?

1:100 = [1 2 3 4 ... 100]

k=1;

total = total + k; (total = 0+1 = 1)

k=2;

total = total + k; (total = 1+2 = 3)

k=3;

total = total + k; (total = 3+3 = 6)

⋮

k=100

total = total + k; (total = 1+2+3+...+99+100)

For while loops:

SYNTAX:

```
while some_statement
  (code block)
end
```

Ex:

```
x=1;
while x < 100
  x = x*2;
end
```

x = 128

This loop gets the first power of 2 larger than 100.

x = 1 < 100 ✓

x = 1·2

x = 2 < 100 ✓

x = 2·2

x = 4 < 100 ✓

x = 4·2

x = 8 < 100 ✓

x = 8·2

x = 16 < 100 ✓

x = 16·2

x = 32 < 100

x = 32·2

x = 64 < 100

x = 64·2

x = 128 ✗ 100

STOP

Ex:

```
x = 0
while x < 100
    x = x * 2;
end
```

$x = 0 < 100$

$x = 0.2 = 0$

$x = 0 < 100$

$x = 0.2 = 0$

$x = 0 < 100$

RUNS FOREVER!

Ex: Back in Calculus I

Newton's method to find a solution of $f(x) = 0$.

x_0 = some initial guess

$x_{n+1} = x_n - f(x_n) / f'(x_n)$

Special case: $f(x) = x^2 - 2$ (so $f(x) = 0$ when $x = \sqrt{2}$)

$f'(x) = 2x$

$x_0 = 1$

$x_{n+1} = x_n - f(x_n) / f'(x_n)$

$$\boxed{x_{n+1} = x_n - (x_n^2 - 2)/(2x_n)}$$

Stop when error $= |x_{n+1} - x_n| < 0.0000001$ (1e-7)

```
x0 = 1
error = 1
while (error > 1e-7)
    x1 = x0 - (x0^2 - 2)/(2*x0);
    error = abs(x1 - x0);
    x0 = x1;
end
```

- Use for loops when you know how many times to execute a code block, or you're performing an operation over an array
- Use while loops when you need to perform a code block until some condition holds

Advice: avoid using while, when you could get away with using a for loop.