

LOOPS + ITERATIONS

WHAT IS THE CORE ELEMENT OF ANY PATTERN?

WHAT IS THE CORE ELEMENT OF ANY PATTERN?

REPETITION

YOU KNOW HOW TO DRAW A CIRCLE.
BUT IF YOU WANT 50 CIRCLES...

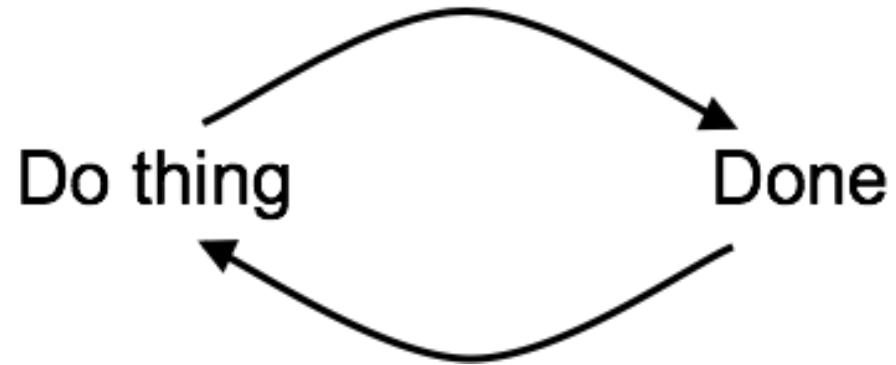
```
ellipse(10, 10, 10, 10);  
ellipse(10, 20, 10, 10);  
ellipse(10, 30, 10, 10);  
ellipse(10, 40, 10, 10);  
ellipse(10, 50, 10, 10);  
ellipse(10, 60, 10, 10);  
ellipse(10, 70, 10, 10);  
ellipse(10, 80, 10, 10);  
ellipse(10, 90, 10, 10);  
ellipse(10, 100, 10, 10);  
ellipse(10, 110, 10, 10);  
ellipse(10, 120, 10, 10);  
ellipse(10, 130, 10, 10);  
ellipse(10, 140, 10, 10);
```

**YOU GET THE POINT.
IT IS SUPER BORING AND ANNOYING.:)**

ADDING REPETITION WITH LOOPS

IF YOU WANT TO ADD REPETITION, USE LOOPS.

WE WANT THE COMPUTER TO DO THIS:



Conditionals

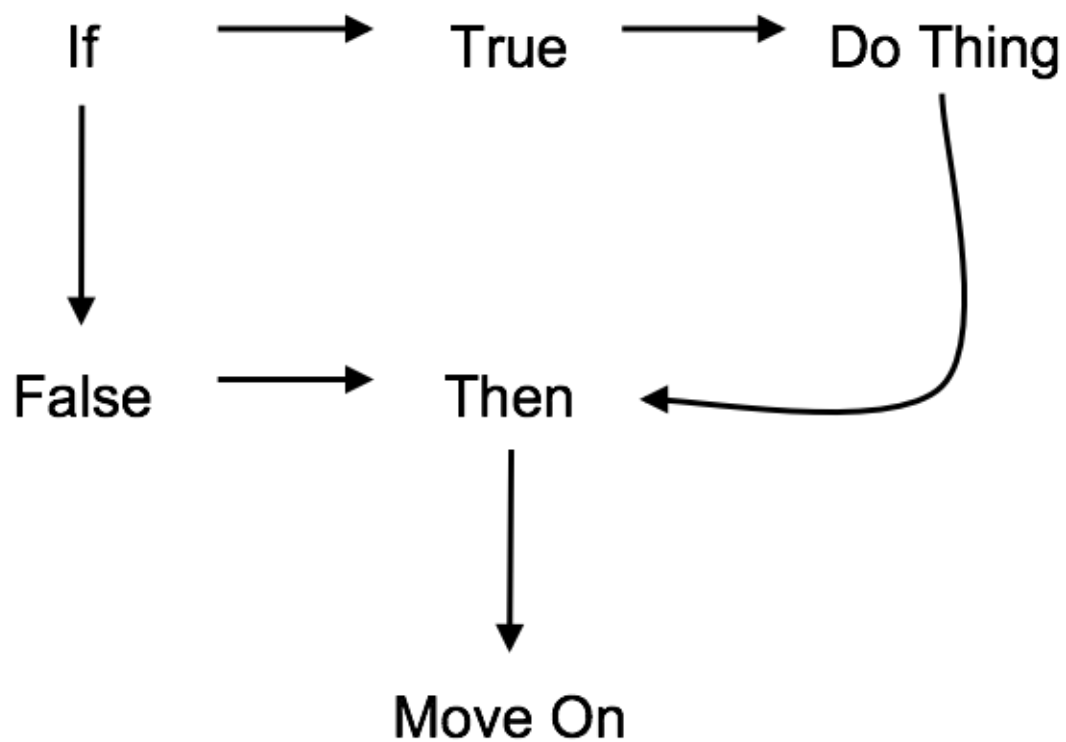
NEW SYNTAX

```
If ( conditional ) {  
    // commands ...  
}
```

CONDITIONAL // IS ANY BOOLEAN EXPRESSION
COMMANDS // ARE ANY ARBITRARY COMMANDS

WHAT THIS WILL DO IS EXECUTE ALL THE COMMANDS, IN ORDER, OVER AND OVER AGAIN, AS LONG AS THE CONDITIONAL IS TRUE -- AKA, "IF" IT IS TRUE, HENCE THE NAME.

“if ()”



If I am bored, then I watch Netflix.
Otherwise, I will not watch.

```
if (bored) {  
    WATCH NETFLIX;  
}  
else {  
    DO NOT WATCH NETFLIX;  
}
```

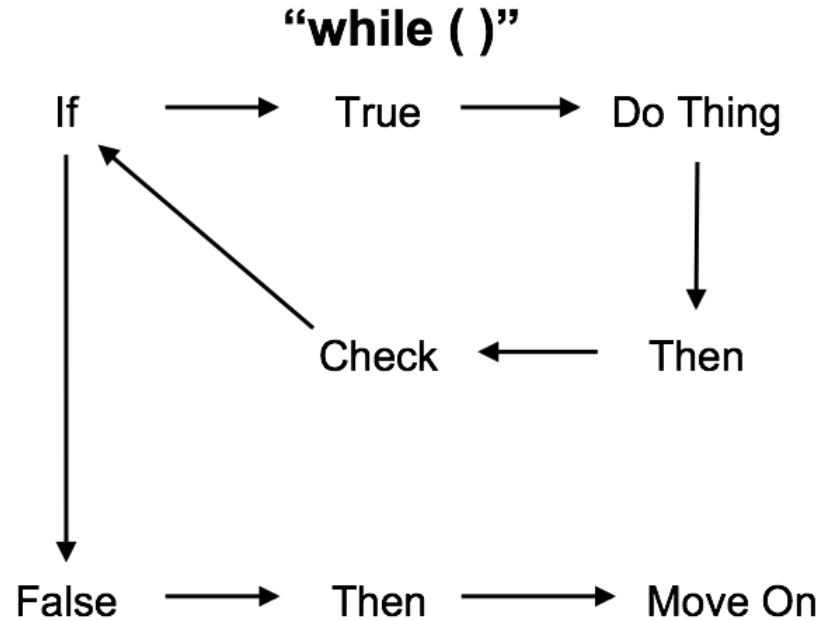
If I am bored **and** do not have money to go
out, then I watch Netflix.
Otherwise, I will not watch.

```
if (bored && no money ) {  
    WATCH NETFLIX;  
}  
else {  
    DO NOT WATCH NETFLIX;  
}
```

If I am bored **or** have time to waste,
then I watch Netflix.
Otherwise, I will not watch.


```
if (bored || have time to waste) {  
    WATCH NETFLIX;  
}  
else {  
    DO NOT WATCH NETFLIX;  
}
```


LET US MIX THE IF () CONDITIONAL WITH OUR EARLIER LOOP.




“while()” vs “if()”

```
if (test) {  
    stuff to do if true;  
}
```



Happens only once, and
then exits.

```
while (test) {  
    stuff to do if true;  
}
```




Repeats for the entire duration
that the **test** is true.

“while()” vs “if()”

If **test** remains unchanged,
program continues


```
if (test) {  
    stuff to do if true;  
}
```



Happens only once, and
then exits.

If **test** remains unchanged, program
loops until a change occurs.

```
while (test) {  
    stuff to do if true;  
}
```



Repeats for the entire duration
that the **test** is true.

START WITH THE FOLLOWING CODE:

```
rect(200,200,200,20);
```

THAT'S NICE, BUT SAY WE WANT TO ADD 2 MORE.
WE COULD COPY/PASTE AND CHANGE SOME VALUES LIKE THIS:

```
rect(200,100, 200,20);  
rect(200,200, 200,20);  
rect(200,300, 200,20);
```

...WHAT IF NOW WE WANT TO ADD 10 MORE. THIS COULD GET
TEDIOUS REALLY QUICKLY. SO LET'S SEE HOW WE COULD DO
IT WITH A WHILE LOOP:

```
int i = 100;  
while ( i <= 300 ) {  
    rect(200,i, 200,20);  
    i = i + 100;  
}
```

**OR WE COULD MAKE THEM EXTEND FURTHER
DOWN IN THE WINDOW:**

```
int i = 100;
while ( i <= 600 ) {
    rect(200,i, 200,20);
    i = i + 100;
}
```

EXPERIMENT WITH THIS AND SEE WHAT YOU GET.

**...WHAT IF YOU WANTED TO MODIFY WHAT YOU WERE DOING
INSIDE THE LOOP, EACH TIME YOU DID IT. TAKE A LOOK AT
THIS:**

```
int i = 100;
while ( i <= 250 ) {
    fill(i,100,200);
    rect(200,i, 200,20);
    i = i + 50;
}
```


The arguments to `fill()` will be:

100,100,200 blue

150,100,200 more red

200,100,200 even more red

250,100,200 purplish

**SO FAR WE HAVE ONLY BEEN LOOKING AT ONE LOOP.
BUT YOU CAN COMBINE THEM IN INTERESTING WAYS. THINK
ABOUT A LOOP OF LOOPS!**

```
int i = 100;
while ( i <= 300 ) { // outter
    int j = 100;
    while ( j <= 300 ) { // inner
        fill(i,100,250,100);
        rect(j,i, 20,20);
        j = j + 50;
    }
    i = i + 50;
}
```

SO THE "OUTTER" LOOP IS GOING TO REPEAT 5 TIMES
(I=100,150,200,250,300). BUT EACH TIME IT DOES THAT, THE
"INNER LOOP" IS ALSO GOING TO REPEAT 5 TIMES
(J=100,150,200,250,300). SO HOW MANY RECTANGLES WILL GET
DRAWN ON THE SCREEN? (REPHRASING HINT: ONE LOOP REPEATS 5
TIMES, AND EACH TIME IT DOES, THE INNER LOOP REPEATS 5
TIMES.)

THIS IS CALLED A **NESTED LOOP** AND IS A VERY POWERFUL IDEA.

FOR LOOPS: THE SAME BUT DIFFERENT

ALL THE ABOVE DISCUSSION IS ABOUT WHILE LOOPS. BUT THERE IS ANOTHER KIND OF SYNTAX FOR CREATING THIS SAME EXACT BEHAVIOR: THE FOR LOOP.

YOU CAN DO EXACTLY THE SAME THING WITH FOR AND WHILE LOOPS. THE ONLY DIFFERENCE IS THE SYNTAX USED TO WRITE THEM.

CONSIDER TWO EXAMPLES. THE FOLLOWING TWO CODE SNIPPETS ARE COMPLETELY EQUIVALENT, JUST WRITTEN DIFFERENTLY:

```
for (int i = 10; i < 50; i = i + 10) {  
    println("i = " + i);  
    rect(i,i,5,5);  
}
```

```
int i = 10;  
while ( i < 50 ) {  
    println("i = " + i);  
    rect(i,i,5,5);  
    i = i + 10;  
}
```

BOTH EXAMPLES CONTAIN: A **VARIABLE DECLARATION**, A **BOOLEAN EXPRESSION** AND A **VARIABLE ASSIGNMENT** THAT INCREMENTS THE VARIABLE. THEY ARE ALL THERE, JUST IN DIFFERENT PLACES.

USE WHICHEVER ONE SEEMS MORE CLEAR TO YOU. IN SOME SITUATIONS, ONE MIGHT SEEM MORE APPROPRIATE THAN THE OTHER. BUT IT IS REALLY JUST AN ISSUE OF CLARITY AND READABILITY.

Yaay!
Time to Code More



Inspirational Examples

Assignment

ALGORITHMIC QUILTS

Encyclopedia Of Pieced Quilt Patterns,
Barbara Brackman