Ruby For Programmers

This cheat-sheet accompanies the Bitwise Courses course on *Ruby For Programmers* by Huw Collingbourne.

for

A standard Ruby `for` loop is like a `foreach` loop in other languages – it iterates over the items in a collection.

```
for i in [1,2,3]  do
  puts( i )
end
```

Output:

```
1
2
3
```

```
for s in ['one','two','three'] #do
  puts( s )
end
```

Output:

```
one
two
three
```

each

You can also use the `each` method to iterate over an array.

```
[1,2,3].each  do |i|
   puts( i )
end
```

Output:

```
1
2
3
```

while

A `while` loop executes while some test value remains true. You can put the `while` test at the start of a loop or at the end. When it is placed at the beginning, the code in the loop will execute 0 or more times (it will fail immediately if the test condition is false).

```
puts( "starting 1st while loop" )
i = 10
while i < 10
  puts(i)
  i += 1
end
puts( "1st while loop ended" )
```

Output:

```
starting 1st while loop
1st while loop ended
```

If the `while` test is put at the end of the loop the code will execute 1 or more times (it will execute once *before* the test is evaluated even if the test condition is false).

```
i = 10
puts( "\nstarting 2nd while loop" )
begin
  puts(i)
  i += 1
end while i < 10
puts( "2nd while loop ended" )
```

Output:

```
starting 2nd while loop
10
2nd while loop ended
```

until

Ruby also has an `until` loop which can be thought of as a '*while not*' loop.

```
puts( "starting not while loop" )
i = 0
while i != 10
    print("#{i} ")
    i += 1
end
puts( "\nnot while loop ended" )

puts

puts( "starting until loop" )
i = 0
until i == 10
    print("#{i} ")
    i += 1
end
puts( "\nuntil loop ended" )
```

Output:

```
starting not while loop
0 1 2 3 4 5 6 7 8 9
not while loop ended

starting until loop
0 1 2 3 4 5 6 7 8 9
until loop ended
```