Ruby Loops

For instance, you want to print a string ten times. You can type ten print statement, but it is easier to use a loop. The only thing you have to do is to setup a loop to execute the same block of code a specified number of times. Here we have discussed the loop statements supported by Ruby.

**Ruby while Statement:**

The while statement is simple, it executes code repeatedly as long as the condition is true. A while loop's conditional is separated from code by the reserved word 'do', a newline, backslash \, or a semicolon.

**Syntax:**

while conditional [do]

code

end

**Example:**

The following codes print the numbers 0 through 10. The condition a < 10 is checked before the loop is entered, then the body executes, then the condition is checked again. When the condition results in false the loop is terminated.

x = 1

y = 11

while x < y do

print x ,". Ruby while loop.\n"

x +=1

end

Copy

Output:

1. Ruby while loop.

2. Ruby while loop.

3. Ruby while loop.

4. Ruby while loop.

5. Ruby while loop.

6. Ruby while loop.

7. Ruby while loop.

8. Ruby while loop.

9. Ruby while loop.

10. Ruby while loop.

Within the while statement, the 'do' keyword is optional. The following loop is equivalent to the loop above:

x = 1

y = 11

while x < y

print x ,". Ruby while loop.\n"

x +=1

end

Copy

**Ruby while modifier:**

Like if and unless, while can be used as modifiers.

x = 0

x += 1 while x < 10

p x # prints 10

Copy

You can use begin and end to create a while loop that runs the body once before the condition:

x = 0

begin

x += 1

end while x <10

p x # prints 10

Copy

**Ruby until Statement:**

The until loop executes while a condition is false. An until loop's conditional is separated from code by the reserved word 'do', a newline, backslash \, or a semicolon. Like a while loop, the do is optional.

**Syntax:**

until conditional [do]

code

end

**Example:**

The following script prints the numbers 1 through 10. Like a while loop the condition x > 11 is checked when entering the loop and each time the loop body executes. If the condition is false the loop will continue to execute.

x = 1

y = 11

until x > y do

print x ,". Ruby while loop.\n"

x +=1

end

Copy

Output:

1. Ruby while loop.

2. Ruby while loop.

3. Ruby while loop.

4. Ruby while loop.

5. Ruby while loop.

6. Ruby while loop.

7. Ruby while loop.

8. Ruby while loop.

9. Ruby while loop.

10. Ruby while loop.

**Ruby until modifier:**

Like if and unless, until can be used as modifiers.

**Syntax:**

code until conditional

OR

begin

code

end until conditional

**Example:**

x = 0

x += 1 until x > 10

p x # prints 11

Copy

This will produce the following result:

You can use begin and end to create an until loop that runs the body once before the condition:

x = 0

begin

x += 1

end until x <10

p x # prints 1

Copy

**Ruby for Statement:**

Like most other languages, Python has for loops, The for loop consists of for followed by a variable to contain the iteration argument followed by in and the value to iterate over using each.

* Like while and until, the do is optional.
* The for loop is similar to using each but does not create a new variable scope.
* The result value of a for loop is the value iterated over unless break is used.
* The for loop is rarely used in modern ruby programs.

**Syntax:**

for variable [, variable ...] in expression [do]

code

end

**Example:**

for x in [1, 2, 3, 4] do

puts x

end

for x in 0..4

puts "Value of x is : #{x}"

end

Copy

Output:

1

2

3

4

Value of x is : 0

Value of x is : 1

Value of x is : 2

Value of x is : 3

Value of x is : 4

**Ruby break Statement:**

The break statement is used to terminate a block early. You can also terminate from a while, for loops using a break.

**Syntax :**

break Statement

**Example:**

Following example break a while loop:

x = 0

while true do

puts x

x += 1

break if x > 5

end

Copy

Output:

0

1

2

3

4

5

**Example:**

Following example break a for loop:

**Example:**

Following example break a while loop:

for x in 0..5

if x > 2 then

break

end

puts "Value of x is #{x}"

end

Copy

Output:

Value of x is 0

Value of x is 1

Value of x is 2

**Ruby next Statement:**

The next statement is used to skip the rest of the current iteration. Terminates execution of a block if called within a block.

**Syntax:**

next Statement

**Example:**

for x in 0..6

if x < 3 then

next

end

puts "Value of x is : #{x}"

end

Copy

Output:

Value of x is : 3

Value of x is : 4

Value of x is : 5

Value of x is : 6

**Ruby redo Statement:**

The redo statement is used to redo the current iteration:

**Syntax:**

redo Statement

**Example:**

restart = false

for x in 1..15

if x == 10

if restart == false

puts "Re-doing when x = " + x.to\_s

restart = true

redo

end

end

puts x

end

Copy

Output:

1

2

3

4

5

6

7

8

9

Re-doing when x = 10

10

11

12

13

14

15