You are logged in as C. O'Keefe (Logout)

Core Ruby 41st Batch

Home ▶ PORPC101-41C ▶ 5 April - 11 April ▶ Some useful Ruby methods

Navigation

Home

- My home

Site pages

My profile

Current course

PORPC101-41C

Participants

General

5 April - 11 April

Read: The Philosophy of Ruby

Ruby Coding Style / Convention

How to use IRB

Course Ruby Programs

Week 1: Tutorial

Some useful Ruby methods

Week 1: Exercises

Week 1: Forum

PARALLEL TRACK Start using Git and GitHub

Git/GitHub Discussions

Week 1: Quiz

12 April - 18 April

19 April - 25 April

Some useful Ruby methods

printf

Another output method we use a lot is **printf**, which prints its arguments under the control of a format string (just like printf in C or Perl).

```
printf("Number: %5.2f,\nString: %s\n", 7.18, "FOR
```

The output is:

```
Number:  7.18,
String: FORPC101
```

In this example, the format string "Number: %5.2f,\nString: %s\n" tells **printf** to substitute in a floating-point number (allowing five characters in total, with two after the decimal point) and a string. Notice the newlines (\n) embedded in the string; each moves the output onto the next line.

step

Class **Numeric** also provides the more general method **step**, which is
more like a traditional for loop.

```
8.step(40, 5) {|i| print i, " " }
```

This produces an output as:

```
>ruby tmp2.rb

8 13 18 23 28 33 38 >Exit code: 0
```

exit(true | false | status=1)

26 April - 2 May
3 May - 9 May
10 May - 16 May
17 May - 23 May
24 May - 30 May
31 May - 6 June
7 June - 13 June
My courses

Settings

Course administration

My profile settings

The method **exit** terminates your program, returning a status value to the operating system. The optional parameter is used to return a status code to the invoking environment. With an argument of **true**, exits with a status of zero. With an argument that is **false** (or no argument), exits with a status of 1, otherwise exits with the given status.

For example:

```
exit 99
```

will exit with a status of 99.

local_variables returns the names of the current local variables in a particular context. It is a reflective method and can be placed anywhere in the program ie. even on line 1. In the example:

```
a = 10

def mtd

 b = 20

end

puts local_variables
```

The output is:

```
>ruby tmp2.rb

a

>Exit code: 0
```

Last modified: Friday, 25 May 2012, 11:07 PM

You are logged in as C. O'Keefe (Logout)