Blocks In Perl

Talk to a Teacher
http://spoken-tutorial.org
National Mission on Education through ICT
http://sakshat.ac.in

Amol Brahmankar

10 March 2014

Learning Objective

In this tutorial, we will learn

Learning Objective

In this tutorial, we will learn

- Various `BLOCKS` available in Perl

System Requirements

- **Ubuntu Linux 12.04 OS**

System Requirements

- **Ubuntu Linux 12.04 OS**
- **Perl 5.14.2**

System Requirements

- **Ubuntu Linux 12.04 OS**
- **Perl 5.14.2**
- **gedit Text Editor**

Pre-requisites

Basic knowledge of

- **Variables and Comments in Perl**

Pre-requisites

Basic knowledge of

- Variables and Comments in Perl
- Data Structures will be an added advantage

Pre-requisites

Basic knowledge of

- Variables and Comments in Perl
- Data Structures will be an added advantage
- For relevant tutorials, visit http://spoken-tutorial.org

BLOCKS

- `PERL` **provides 5 special blocks**

BLOCKS

- PERL **provides 5 special blocks**
- **These blocks get executed at various stages of a** PERL **program**

BLOCKS In Perl

These `blocks` **are**

BLOCKS In Perl

These `blocks` are

- **BEGIN**

BLOCKS In Perl

These `blocks` are

- **BEGIN**
- **END**

BLOCKS In Perl

These `blocks` are

- **BEGIN**
- **END**
- **UNITCHECK**

BLOCKS In Perl

These `blocks` are

- **BEGIN**
- **END**
- **UNITCHECK**
- **CHECK**

BLOCKS In Perl

These `blocks` are

- **BEGIN**
- **END**
- **UNITCHECK**
- **CHECK**
- **INIT**

BEGIN Block

- `BEGIN` **block gets executed at the time of compilation**

BEGIN Block

- `BEGIN` **block gets executed at the time of compilation**
- **Code written inside this block gets executed first**

BEGIN Block

- `BEGIN` **block gets executed at the time of compilation**
- **Code written inside this block gets executed first**
- **We can have several** `BEGIN` **blocks inside a** `PERL` **script**

Multiple BEGIN Block

- **These blocks will get executed in the order of declaration**

Multiple BEGIN Block

- **These blocks will get executed in the order of declaration**
- **That is in** `First define First execute` **pattern**

BEGIN Block

Syntax:

BEGIN Block

Syntax:

```
BEGIN {
# Piece of code to be executed
at the time of compilation
}
```

BEGIN Block

From the example it is evident that

BEGIN Block

From the example it is evident that

- **The code written inside `BEGIN` block gets executed first**

BEGIN Block

From the example it is evident that

- The code written inside `BEGIN` block gets executed first
- This is irrespective of the location of the `BEGIN` block inside the `PERL` script

BEGIN Block

From the example it is evident that

- **The code written inside** `BEGIN` **block gets executed first**
- **This is irrespective of the location of the** `BEGIN` **block inside the** `PERL` **script**
- `BEGIN` **blocks gets executed in the** `First In First Out` **manner**

BEGIN Block

One of the use of `BEGIN` block is;

- To include files inside a `PERL` script before actual execution starts

END Block

- `END` **block gets executed at the end of a** `PERL` **program**

END Block

- `END` **block gets executed at the end of a** `PERL` **program**
- **Code written inside this block gets executed after** `PERL` **has finished executing the program**

END Block

- `END` **block gets executed at the end of a** `PERL` **program**
- **Code written inside this block gets executed after** `PERL` **has finished executing the program**
- **We can have several** `END` **blocks inside a** `PERL` **script**

Mutiple END Block

- **These blocks will get executed in reverse order of declaration**

Mutiple END Block

- **These blocks will get executed in reverse order of declaration**
- **That is in `Last define First execute` pattern**

END Block

Syntax:

END Block

Syntax:

```
END {
# Piece of code to be executed
at the end of the PERL script
}
```

END Block

From the example it is evident that

END Block

From the example it is evident that

- **The code written inside `END` block gets executed at the end**

END Block

From the example it is evident that

- The code written inside `END` block gets executed at the end
- This is irrespective of the location of the `END` block inside `PERL` script

END Block

From the example it is evident that

- **The code written inside `END` block gets executed at the end**
- **This is irrespective of the location of the `END` block inside `PERL` script**
- `END` **blocks gets executed in the** `Last In First Out` **manner**

END Block

One of the use of `END` block is;

- To destroy the objects, if any are created in the script

UNITCHECK, CHECK, INIT Block

- `PERL` **also has** `UNITCHECK, CHECK` **and** `INIT` **blocks**

UNITCHECK, CHECK, INIT Block

- `PERL` **also has** `UNITCHECK, CHECK` **and** `INIT` **blocks**
- **These blocks are used rarely by developers**

UNITCHECK, CHECK, INIT Block

These blocks are useful

UNITCHECK, CHECK, INIT Block

These blocks are useful

- To catch the transition between compilation and execution phase of the main program

UNITCHECK, CHECK, INIT Block

These blocks are useful

- **To catch the transition between compilation and execution phase of the main program**
- **To perform some checks or initialisation after compilation and before execution**

UNITCHECK, CHECK, INIT Block

- **UNITCHECK and CHECK blocks runs in** `Last in First out` **manner**

UNITCHECK, CHECK, INIT Block

- **UNITCHECK and CHECK blocks runs in** `Last in First out` **manner**
- **INIT block runs in** `First In First Out` **manner**

UNITCHECK Block

Syntax:

UNITCHECK Block

Syntax:

```
UNITCHECK {
# Piece of code to be executed
}
```

CHECK Block

Syntax:

CHECK Block

Syntax:

```
CHECK {
# Piece of code to be executed
}
```

INIT Block

Syntax:

INIT Block

Syntax:

```
INIT {
# Piece of code to be
initialised
}
```

Summary

In this tutorial we have learnt-

- `BEGIN` **and** `END` **blocks in details**

Summary

In this tutorial we have learnt-

- `BEGIN` **and** `END` **blocks in details**
- **Introduction to** `UNITCHECK`, `CHECK` **and** `INIT` **blocks**

Summary

In this tutorial we have learnt-

- `BEGIN` **and** `END` **blocks in details**
- **Introduction to** `UNITCHECK, CHECK` **and** `INIT` **blocks**
- **Using sample programs**

Assignment

```
#!/usr/bin/perl
END {
     @array = ();
     print “Length of an array inside END
block: ”, $#array + 1, “\n”; }

print “Length of an array: ”, $#array +
1,“\n”;

BEGIN {@array = (1, 2, 3); }
```

About the Spoken Tutorial Project

- **Watch the video available at http://spoken-tutorial.org/What_is_a_Spoken_Tutorial**
- **It summarises the Spoken Tutorial project**

Talk to a Teacher

About the Spoken Tutorial Project

- **Watch the video available at http://spoken-tutorial.org /What_is_a_Spoken_Tutorial**
- **It summarises the Spoken Tutorial project**
- **If you do not have good bandwidth, you can download and watch it**

Talk to a Teacher

Spoken Tutorial Workshops

The Spoken Tutorial Project Team

- Conducts workshops using spoken tutorials
- Gives certificates to those who pass an online test
- For more details, please write to contact@spoken-tutorial.org

Talk to a Teacher

Acknowledgements

- Spoken Tutorial Project is a part of the Talk to a Teacher project
- It is supported by the National Mission on Education through ICT, MHRD, Government of India
- More information on this Mission is available at http://spoken-tutorial.org/NMEICT-Intro