INDEX:

``````

=> This function will get two strings and return the location of the second string within the first string.

#!/usr/bin/perl -w

use strict;

my $a = "This is sparta";

print index $a, "s";

-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-

SUBSTR:

``````````````````````

=> It is basically the opposite of index(). While index() will tell you where is a given string, substr will give you the substring at a given locations.

=> This fuction required 4 parameters:

+ The first one is the string

+ The second is a 0-based location

+ the third is the length of the substring

+ replace [ if required ]

Q. Print the substring “is” from the string “This is sparta”

# vim script.pl

#!/usr/bin/perl -w

use strict;

my $a = "This is sparta";

print substr $a, 2, 2;

# perl script.pl

is

++++++++++++++++++++++++++++++++++++++

=> Count 2 from the left, 7 from the right, return what is between.

```````````````````````````````````````````````````````````````````````````````````````

# vim script.pl

#!/usr/bin/perl -w

use strict;

my $a = "This is sparta";

print substr $a, 2, -7;

# perl script.pl

is is

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

=> You can also leave out the 3rd (length) parameter which will mean: return all the characters after the index

`````````````````````````````````````````````````````````

# vim script.pl

#!/usr/bin/perl

use strict;

use warnings;

my $a = "This is sparta";

print substr $a, 1;

# perl script.pl

his is sparta //Here, all the characters at index 1 and after index 1 are printed

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

=> We can also use a negative number in the offset, which will mean: count 2 from the right

# vim script.pl

#!/usr/bin/perl -w

use strict;

my $a = "This is sparta";

print substr $a, -6;

# perl script.pl

sparta

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Replacing part of a string

``````````````````````````````````

#!/usr/bin/perl

use strict;

use warnings;

my $a = "This is sparta";

substr $a, -6, 6, "BABA";

print "$a\n"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SWAP Variables:

```````````````````````

Q. Swap two variables a and b without using any third variable.

#!/usr/bin/perl -w

use strict;

my $a = "a";

my $b = "b";

($a, $b) = ($b, $a);

print "\$a = $a\n";

print "\$b = $b\n";

# perl script.pl

$a = b

$b = a

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Positional Parameters OR Command line Arguments:

```````````````````````````````````````````````````````````````````````

- Unix/Linux Shell programming you will recognize $0 is being the name of the script.

- @ARGV is just a regular array in Perl, so index starts with $0

#!/usr/bin/perl -w

use strict;

print "@ARGV\n";

my ($a, $b) = @ARGV;

my $name = $ARGV[0];

my $number = $ARGV[1];

print "\$name = $name";

print "\$number = $number";

# perl script.pl arg1 arg2

$0 $1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_OR\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q. Create a perl program that prints the two arguments entered to it.

#!/usr/bin/perl -w

use strict;

my ($name, $number) = @ARGV;

print "\$name = $name\n";

print "\$number = $number\n";

# perl script.pl pop1 pop2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CONTROL STRUCTURE:

``````````````````

Operators:

```````````

1) Relational Operator:

```````````````````````

Numeric comparisons: ==, !=, <, <=, >, >=

String Comparisons: eq, ne, lt, le, gt, ge

RegEx Match/not-match: =~, !~

2) Logical Operators:

``````````````````````

||, &&, !

or, and, not, defined, not defined

3) String Operator:

```````````````````

. (dot) is the string concatenation operator

x is repetition operator

4) File I/O Operators:

``````````````````````

< : Read Operator

> : write Operator

>> : Append Operator

5) File Test Conditions:

````````````````````````

-r : File or Dir is readable

-w : File or Dir is writable

-x : File or Dir is executable

-o : File or Dir is owned by user

-e : File or Dir exists

-z : File exists and has zero size

-s : File exists and has non-zero size

-f : File

-d : Dir

-l : Symbolic Link [shortcut in Linux]

-T : ASCII Text file

-B : File is binary

if...elsif...else:

``````````````````

if (condition){

statement

statement

...

} elsif (condition){

statement

statement

...

} else{

statement

statement

...

}

Q. Create a perl program that prints “one” when 1 is entered, prints “Two” when 2 is inserted and prints “None” when any other number is entered.

# vim script.pl

#!/usr/bin/perl -w

use strict;

print "Enter a number: ";

chomp(my $a = <>);

if ($a == "1"){

print "One\n";

} elsif ($a == "2"){

print "Two\n";

} else {

print "None\n";

}

----------------

Q. Create a perl program that prints “First statement” when argument is between 1 and 5, prints “Second statement” when argument is 10 and prints “None” when any other argument is given.

# vim script.pl

#!/usr/bin/perl -w

use strict;

my $a = $ARGV[0];

if ($a == "1" || $a <= "5" ){

print "First statement\n";

} elsif ($a == "10"){

print "Second Statement\n";

} else {

print "None\n";

}

# perl script.pl 1

First statement

# perl script.pl 10

Second Statement

Why?????????

```````````````````````

# perl script.pl

Enter : ss

Argument "ss" isn't numeric in numeric eq (==) at 1.pl line 8, <> line 1.

First statement

+++++++++++++++++++++++++++++++++++++++++++++++++++++

"defined" or "not defined" in if control structure:

```````````````````````````````````````````````````````````````

Q. Create a perl script that prints “cool” if two arguments are given, prints “Not so cool” if only one argument is given and “Define something” if no arguments are given.

#!/usr/bin/perl -w

use strict;

my ($a, $b) = @ARGV;

if (defined $a && defined $b){

print "Cool\n";

} elsif (not defined $a && not defined $b){

print "Not so cool \n"

} else {

print "Define something\n"

}

-------------------------------------

+++++++++++++++++++++++++++++++++++++

with Array:

```````````

if (@names) {

# do something

}

With HASH

`````````

if (%names) {

# do something

}

-------------------------------------

Q. Create a script to check that command/utility/tool exists or not.

RUN :

# perl script.pl -cmd pwd

Command exists.

# perl script.pl -cmd pwdss

Command not found.

# perl script.pl

Error: arg required

exit 127

--------------------------------------

Q. Create a script which run with an argument, like:

# perl script.pl ARGUMENT

With conditions as if no parameter pass then it should print "[ERROR] No parameter provided. Usage: perl $0 PARAMETER" otherwise print the parameter as it is (given as argument)

Q. Create a perl program that prints the output of command given as argument

#!/usr/bin/perl -w

system($ARGV[0]);

+++++++++++++++++++++++++++++++++++++

SPLIT:

``````

=> split REGEX, STRING will split the STRING at every match of the REGEX.

eg:

split /WORD/, $str;

=> split REGEX, STRING, LIMIT where LIMIT is a positive number. This will split the STRING at every match of the REGEX, but will stop after it found LIMIT matches.

------------------------------

Q. Create a perl program that prints the output of ‘id root’ line by line

#!/usr/bin/perl -w

use strict;

my @a = `id root`;

my ($d, $e, $f) = split / /, "@a";

print "$d\n";

print "$e\n";

print "$f\n";

----------------------------

Q. Create a perl program that prints “username”, “/home” and “shell” of the first line of /etc/passwd file: Example of root:x:0:0:root:/root:/bin/bash

#!/usr/bin/perl -w

use strict;

my $a = `head -1 /etc/passwd`;

my @out = split /:/, $a;

print "$out[0],$out[5],$out[6]"

LOOPs:

``````````````

For Loop:

``````````

for (condition){

statement

}

--------------

Q. for loop example easy perl program

#!/usr/bin/perl -w

use strict;

for (1..10){

print "Ulalala\n"

}

----------------------------------

Q. Create a perl program that prints 1 .. 50

#!/usr/bin/perl -w

use strict;

my $i;

for ($i=1; $i<=50; $i++){

print "$i\n";

sleep(1);

}

# perl script.pl

------------------------------------

Q. Create a perl program that prints 10, 9, 8, 7 … 1, if the argument given is 10.

#!/usr/bin/perl -w

use strict;

my $i;

for ($i=$ARGV[0]; $i>=1;$i--){

print "$i\n";

sleep(1)

}

-------------------------------

Q. Create a perl program that prints arguments in reverse order however number of arguments are given.

#!/usr/bin/perl -w

use strict;

my $i;

for ($i=$#ARGV; $i>=0; $i--){

print "$ARGV[$i]\n";

sleep(1);

}

-------------------------------

#!/usr/bin/perl -w

use strict;

my @arr1 = ("one", "two", "three");

for my $i (@arr1){

print "$i\n";

}

# perl script.pl