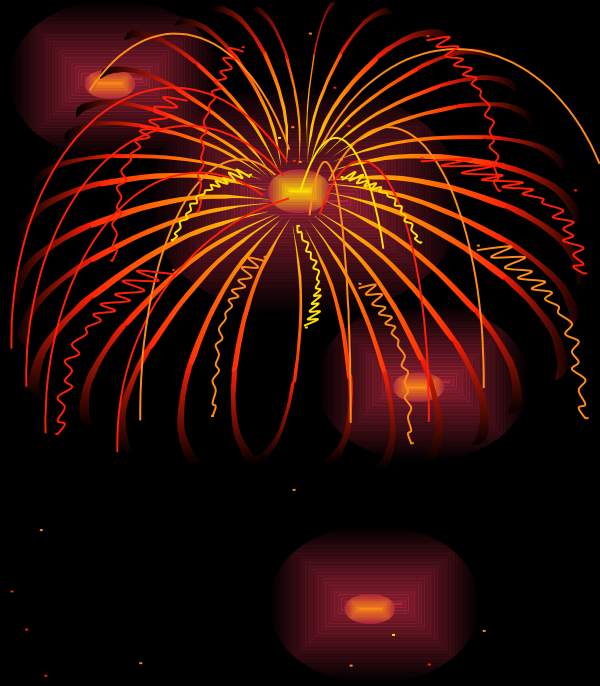
Introduction to Perl & BioPerl

Dr G. P. S. Raghava
Bioinformatics Centre
IMTECH, Chandigarh

Email: raghava@imtech.res.in
Web: http://imtech.res.in/raghava/

Perl Introduction



- Uses
  - Shell scripts
  - CGI, web engines
- Good at
  - Text processing
  - Small/Medium sized projects
  - Quick and dirty solutions
  - Portability (to a certain degree)

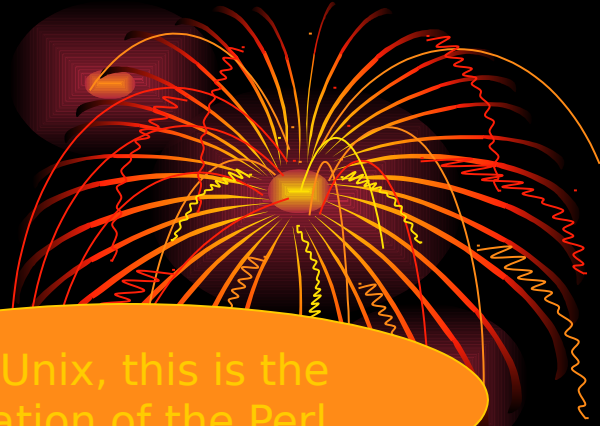
Perl

- Practical Extraction and Report Language
  - Created by Larry Wall
- Runs on just about every platform
  - Most popular on Unix/Linux systems
- Excellent language for file and data processing

Basic Concepts

- Perl files extension .Pl
- Can create self executing scripts
- Advantage of Perl
- Can use system commands
- Comment entry
- Print stuff on screen

Simple Program



On Unix, this is the location of the Perl interpreter

```
#!/usr/local/bin/perl
# This is a comment line. This program prints “Hello World.”
# to the screen.

print “Hello world.\n”;
```

Comments start with # and end with the end of the line

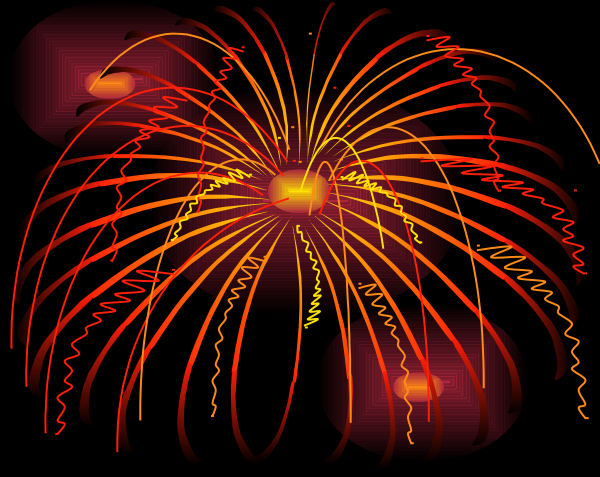
Newline character

Program statements are terminated with semicolons

How to Store Values

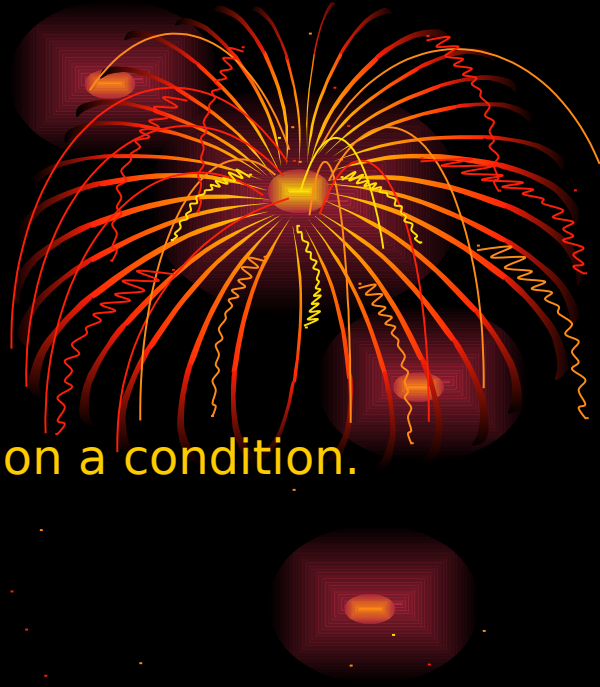
- Scalar variables
- List variables
- Push,pop,shift,unshift,reverse
- Hashes,keys,values,each
- Read from terminal, command line arguments
- Read and write to files

Perl Data Types



- Three Main Datatypes
  - Scalar
    - A single number, string or reference.
    - $society = “Redbrick”;
  - List
    - A collection of scalar datatypes.
    - @societies = (“RB”, “Drama”, “Film”);
    - Length of an array with “scalar @list”
  - Hash
    - Pairs of scalars, accessed by keys.
    - %hash = (“colour” => “red”,
      “make” => “corvette”);

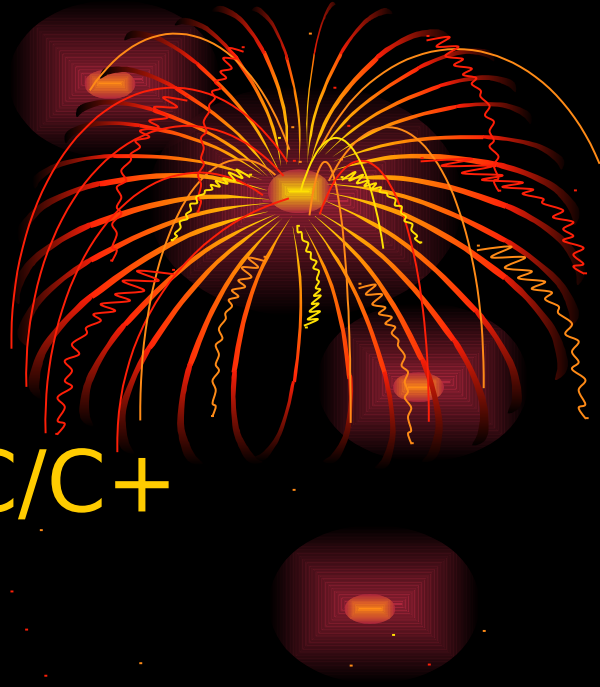
Perl Basics – ‘if’



- Selective evaluate blocks of code depending on a condition.

```
If ($string eq “blah”)
{
	print “String is blah\n”;
}
elsif ($string eq “spoo”)
{
	Print “String is spoo\n”;
}
else
{
	Print “What the hell?\n”;
}
```

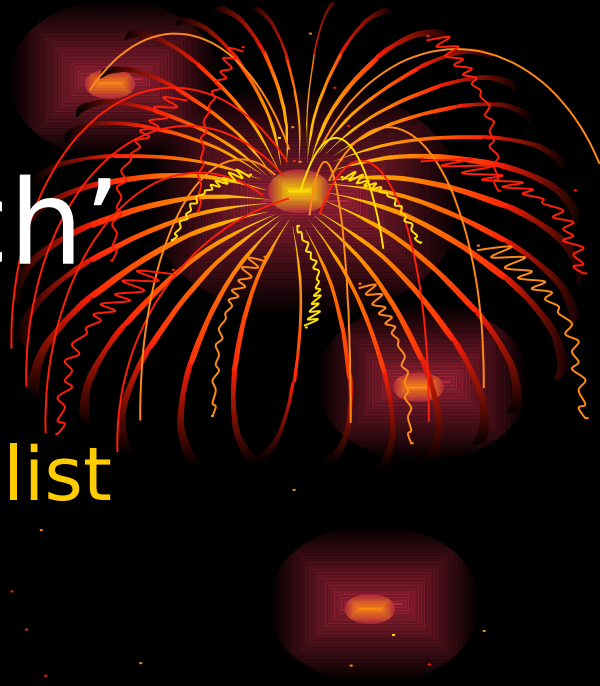
Perl Basics – ‘for’



- Practically the same as C/C++/Java

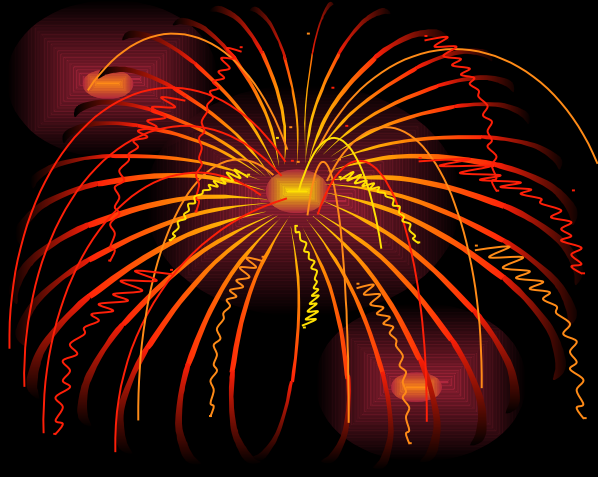
```
for ($i = 0; $i < 10; $i++)
{
    print “i is “ . $i . “\n”;
}
```

Perl Basics – ‘foreach’



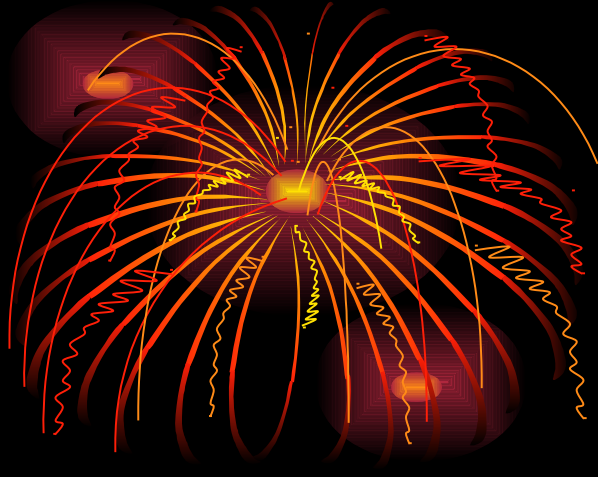
- A handy way of processing a list
  - foreach $grocery (@groceries)
    {
     scan($grocery);
    }
- Can use the default variable ($_)
  - foreach (@groceries)
    {
     scan($_);
    }

Control Structures



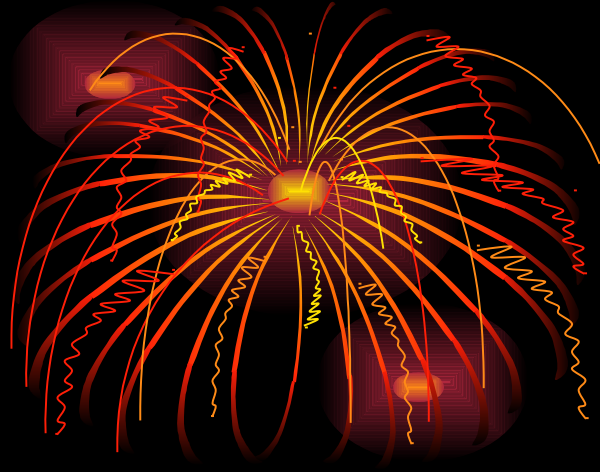
```
#!/usr/local/bin/perl
# Print out 0,1,2,3,4,5,6,7,8,9
# in this case, $x is local only to the loop because my is used
for (my $x = 0; $x < 10; $x++) {
    print “$x”;
    if ($x < 9) {
        print “,”;
    }
}
print “\n”;
```

Control Structures



```
#!/usr/local/bin/perl
# Demonstrate the foreach loop, which goes through elements
# in an array.

my @users = (“bonzo”, “gorgon”, “pluto”, “sting”);

foreach $user (@users) {
    print “$user is alright.\n”;
}
```

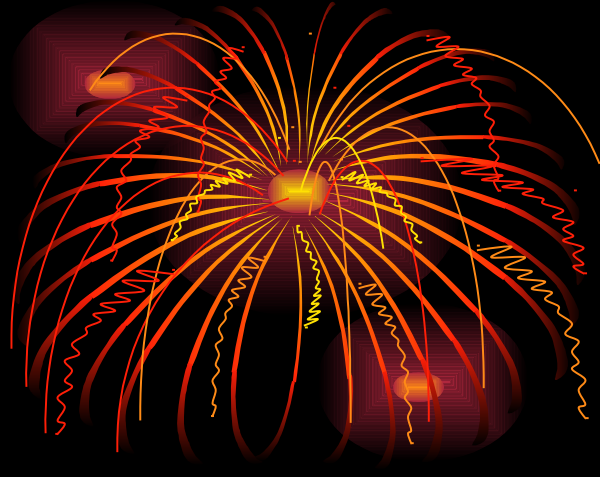
Functions



- Use sub to create a function.
  - No named formal parameters, assign @_ to local subroutine variables.

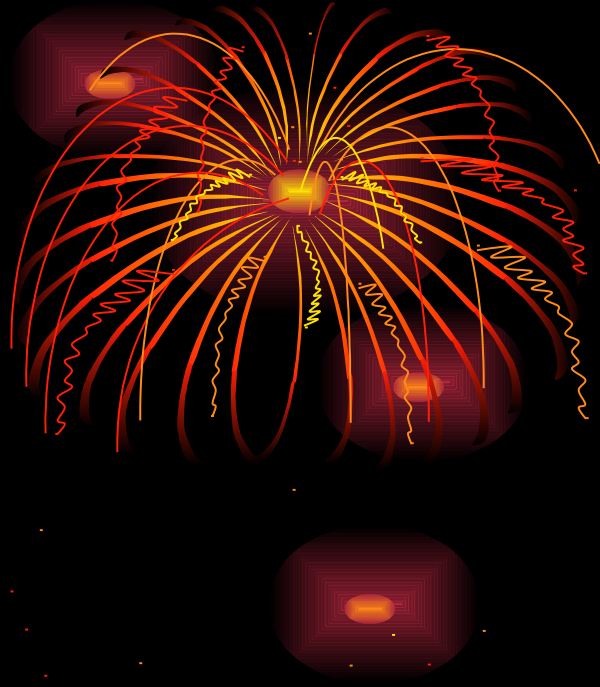
```
#!/usr/local/bin/perl
# Subroutine for calculating the maximum
sub max {
    my $max = shift(@_);    # shift removes the first value from @_
    foreach $val (@_) {
        $max = $val if $max < $val;  # Notice perl allows post ifs
    }
    return $max;
}

$high = max(1,5,6,7,8,2,4,9,3,4);
print “High value is $high\n”;
```

Perl – File I/O Open



- Opening a file
  - Open(HANDLE,”filename.txt”);
    - ‘HANDLE’ is name of an undeclared filehandle.
    - “filename.txt” is the file.
  - Open modes – Specified in the filename
    - “filename.txt” – Read.
    - “>filename.txt” – Truncate and write to file.
    - “>>filename.txt” – Append to file.

Perl – File I/O - Read



- Reading
  - $string = <FILEHANDLE>;
  - @list = <FILEHANDLE>;

```
while ($string = <FILEHANDLE>)
{
    print “LINE : “ . $string;
}
```

Perl – File I/O - Write

- Writing to a file.

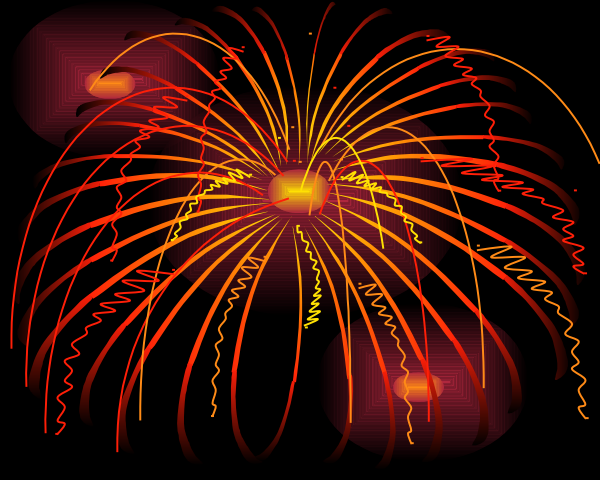
```
print FILEHANDLE “Hello”;

foreach $item (@list)
{
    print FILEHANDLE $item;
}

foreach (@list)
{
    print FILEHANDLE;
}
```

Files

- File handles are used to access files
  - open and close functions

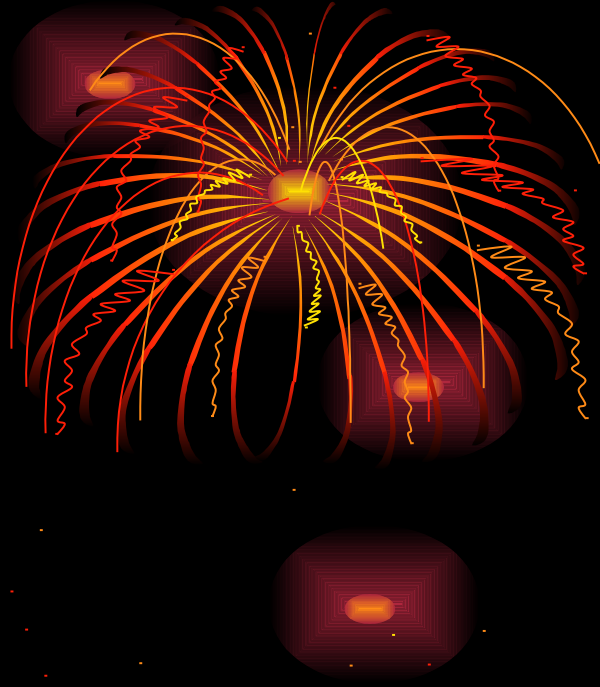
```
#!/usr/local/bin/perl
# Open a file and print its contents to copy.txt
my $filename = $ARGV[0];
open(MYFILE, “<$filename”);  # < indicates read, > indicates write
open(OUTPUT, “>copy.txt”);
while ($line = <MYFILE>) {   # The <> operator reads a line
    print OUTPUT $line;      # no newline is needed, read from file
}
close MYFILE;                # Parenthesis are optional
```

Platforms



- HP-UX / Itanium ('ia64') 11.0
- HP-UX / PA-RISC 11.0
- MacOS
- MacOS X
- CygWin NT_5 v. 1.3.10 on Windows 2000 5.00
- Win32, WinNT i386
- IRIX64 6.5 SGI
- Solaris 2.8 UltraSparc
- OpenBSD 2.8 i386
- RedHat Linux 7.2 i686
- Linux i386

Perl CGI – Example



```
#!/usr/bin/perl –w
# My Third Perl script.
# I call this script like :
# http:///www.redbrick.dcu.ie/~username/script.pl?name=something

use strict;
use CGI;

my $query = new CGI;
my $name = $query->param('name');

print $query->header();
If ($name eq "")
{
     print "No Name Specified!\n";
}
else
{
     print "Hello, " . $name . "!\n";
}
```