

Perl Lesson 1

L615

Spring 2009

1. Obtain a copy of Perl
 - Mac/Unix/Linux: should already be installed
 - Windows: see, e.g., <http://www.perl.org/get.html>
 - Perl is already available on **jones**
2. Open up a terminal. You'll do something like:
 - Mac/Unix/Linux: Applications → Utilities → Terminal
 - Windows: Start → (All) Programs → Accessories → Command Prompt
 - Two things to know:
 - (a) You'll need to be in the same directory as your perl program to run it
 - (b) You'll need to make sure Perl is "globally" available (this can be an issue on windows machines; you may have to set an environment variable—ask me for help).
3. Save the following program to a file (preferably with `.pl` extension) and run it, to verify that everything works:

```
#!/usr/bin/perl
print "Hello world!";
```

- You run the file by typing `perl hello.pl` in the directory where `hello.pl` is
4. Some basics:
 - First line of your perl programs: `#!/usr/bin/perl ...` or something like that: tells system where perl can be found
 - `print` — not surprisingly, allows you to print
 - `print "Hello world!";`
 - Every command ends with a semi-colon

- Comments start with a # sign
- The full program so far:

```
#!/usr/bin/perl

# Command to print "Hello world!"
print "Hello world!\n";
```

5. Scalar variables: numbers & strings

- Scalar variables start with \$
- Numbers:

```
$a = 1 + 2; # Add 1 and 2 and store in $a
$a = 3 - 4; # Subtract 4 from 3 and store in $a
$a = 5 * 6; # Multiply 5 and 6
$a = 7 / 8; # Divide 7 by 8 to give 0.875
$a = 9 ** 10; # Nine to the power of 10
$a = 5 % 2; # Remainder of 5 divided by 2

++$a; # Increment $a and then return it
$a++; # Return $a and then increment it
--$a; # Decrement $a and then return it
$a--; # Return $a and then decrement it
```

- Strings:

```
$b = 'this'; $c = 'string';
$a = $b . $c; # Concatenate $b and $c
$c = 3;
$a = $b x $c; # $b repeated $c times
# Compare:
print 'this string\n';
# with:
print "this string\n";
```

- String interpolation: if you use double quotes, you can include variables within them

```
# These are equivalent:
print "$a and $b";
print $a . 'and' . $b;
```