1. So-called *filehandles* allow you to input/output text to particular locations, and Perl has several filehandles automatically defined:

- **STDIN** — standard input (i.e., terminal/shell/command prompt)
- **STDOUT** — standard output (i.e., terminal/shell/command prompt)
- **STDERR** — standard error
- **DATA** — stores data
- **ARGV** — list of argument variables
- **ARGVOUT**

But we can also define our own filehandles:

```perl
# The '<' indicates input
open $MY_IN, '<', $myinputfile; # 3 argument version
    # same as: open($MY_IN, $myinputfile);
    chomp(@lines = <$MY_IN>);
    close($MY_IN);

# The '>' indicates output
open $MY_OUT, '>temp.txt'; # 2 argument version

# select the output filehandle
select $MY_OUT;
print "this is going to temp.txt";

# return output to screen:
select STDOUT;
print "this is going to the screen\n";
close $MY_OUT;
```
2. Subroutines: allow us to write cleaner code by defining a chunk of code as a particular function which we can call again and again

- sub — how to define a subroutine
- Arguments are defined by @_ (and thus $_[0], $_[1], etc.)

```
sub simple_print {
  print "This is my special global variable \$n: \$n\n";
}

sub arg_print {
  print "I believe your value is: $_[0]\n";
}

# declare a global variable:
$n = 'epstein';

# invoke the function:
simple_print;
# and provide an argument:
arg_print('barbarino');
```

3. Return values—perl automatically returns the last value

- Can also use the return operator, to force exit from the subroutine, or to make your code cleaner

```
sub minimum {
  # @_ is the incoming array of arguments

  # set the minimum to be the first element
  $min = shift @_; 
  foreach (@_) { 
    if ($_ < $min) { 
      $min = $_; 
    }
  } 
  # equivalent to: return $min; 
  $min;
}

$the_min = minimum(7,4,5,6); # $the_min = 4
```