

Boolean Expressions

- All kinds of expressions can evaluate to boolean.
- 0, empty strings, undefs evaluate to false.
- Other things are true.

Boolean operators:

- and, &&.
- or, ||.

Different precence!

```
$a = 0 or print 'Yes!'; print "A=$a\n";  
$a = 0 || print 'Yes!'; print "A=$a\n";  
# prints  
Yes!A=0  
Yes!A=1
```

Control Structures

- `if`:
`print "Go biking!\n" if $daysSinceLastExercise>7;`
- Traditional way:
`if($cond){...} elsif($cond2){...} else{...}`
- `unless`:
`washDishes() unless $#itemsInSink == -1;`

Same effect can be achieved differently (TIMTOWTDI):

```
isGeometryFinal()  
  or do { prepareForRestart(); print '.' }  
@bonds=identifyBonds()  
  and stretchBonds @bonds;
```

There Are Many Ways To Do It

Write

```
if($condition) { func(); }
```

in as many ways as you can. For example:

```
unless(not $condition) { func(); }
```

etc.

Loops

- `while($condition) { ... }` or
- `while($condition) { ... } continue{...}`
- `until($condition) { ... }`. Also here:
- `until($condition) { ... } continue {...}`.
- `for` comes in two brands:
 - # Iteration over elements...
 - `for $element (@list) { ... }`
 - # C style...
 - `for(my $i=0; $i<$maxI; $i++) { }`

Loop Terminations and Short Circuits

next, last and redo constructs available.

```
foreach $a (@arr) {  
  last if $a eq '';  
  next unless $a =~ /^\\d+$/;  
  ...  
}
```

Switch Case Constructs

No explicit Switch Case/Select statement. Replacements available:

```
for($dish) {  
  /^fish$/      &&do{print "We do not serve fish today\n";last;}  
  /^potatoes$/ &&do{print "We serve them fresh baked!\n";last;}  
  /^chips$/     &&do{print "Out of oil today.\n"; last;}  
}
```

Iterating Over A Hash

Two ways possible:

```
foreach $k (keys %hash) {  
    ...  
}
```

or more memory efficient for large hashes, and when order is irrelevant.

```
while( ($k, $v) = each %hash) {  
    ...  
}
```

Program Execution

TIMTOWTDI in Action.

- Do it C-style!

```
if(system($cmd) != 0) {  
    warn "I am afraid I couldn't do it, Dave\n"  
}
```

- Be lazy, shell-style: @output = '\$cmd';
- Pipes forever:

```
open PIPE, "g03 $input|" or die;  
@arr = <PIPE>;  
close PIPE or die "g03 does not like *your* input\n";  
...  
open PIPE, "|gzip -9 > matrix.gz" or die;  
my $lasthandle = select PIPE;  
printLargeMatrix();  
select $lasthandle;  
close PIPE or die "gzip failed, apparently.\n";
```

Passing File Handles Around

```
# Reset a default filehandle:
sub a {
    print "Data\n";
}
my $t = select FL;
a();
select $t;
#...
# Pass the file handle as a reference to typeglob:
sub a {
    my ($fileHandle) = @_;
    print $fileHandle "Data\n";
}
a \*FL;
```

Other File Operations

- `link`
- `symlink`
- `unlink`
- `rename`

Examples: H2 Input Generation.

```
for($i=1; $i<3.0; $i+= 0.1) {  
  open FL, '>input' or die;  
  print FL << "EOF"  
  % B3LYP/6-31G  
  
  H 0 0 0  
  H 0 0 $i  
  EOF  
  die unless 0 == system 'g03 < input > output';  
  $energy = (split " ", `grep "SCF Done:" output`)[4];  
  die unless defined $energy;  
  $energies{$i} = $energy;  
}
```

Final Notes

- Reading after: Scan through Chapters 3 and 4.
- Hand in for the next session:
 - Practice conditional statements.
 - Practice string handling and iterations over vector elements: extract vectors of occupied and virtual orbital energies, create a sorted list of all differences. Bonus points for printing the indexes of corresponding virtual and occupied orbitals.