regular expressions

```
 1 print "Ancient element\n" if ($substance =~ m/earth|air|fire|water/i);
 2 print "Stop codon\n" if ($seq =~ m/TAA|TAG|TGA/i);
 3 print "Author's name\n" if ($text =~ m/Ian|Keith/i);
 4
 5 !perl
 6 if( $string =~ /[A-Z]/) {
 7     #matches a capital letter
 8 }
 9
10 if( $string =~ /([A-Z]+)/) {
11     print "capital letters matched are $1\n";
12 }
```

Patterns

- `$str =~ /match/`
- `$str =~ m!match!` # use m to specify the separator
- `$str =~ s/from/to/` # replace
- `$str =~ tr/[A-Z]/[a-z]/` #translate

Reverse complement DNA

Use `tr` and `reverse` to convert DNA into reverse complement.

```
my $str = 'AGCATA';
$str =~ tr/ACGT/TGCA/;
my $rev = reverse($str);
print "$str\n";
print "$rev\n";
```

produces:

```
 AGCATA
 TATGCT
```

Pattern types

- \d – number
- \s – whitespeace
- \D – NOT a number
- \S – NOT whitespace
- . – match anything
- `+` – modifier to match 1 to many times
- `*` – modifier to match 0 to many times
- ? – modifier to match 0 or 1 times
- {N,M} – specify an exact number of matches
- ^ – starts with this
- $ – end of match
- | – to specify 'or' so that multiple things could match

try this

Matching 'GENE' followed by a number, will skip GENEX321

```
my @strs = qw(GENE124 GENE112 GENE180 GENEX321 AGENE12);
for my $str ( @strs ) {
 if( $str =~ /GENE\d+/ ) {
     print $str, "\n";
 }
}
```

try this

Matching 'GENE' followed by a number, will skip GENEX321 and also AGENE12

```
my @strs = qw(GENE124 GENE112 GENE180 GENEX321 AGENE12);
for my $str ( @strs ) {
 if( $str =~ /^GENE\d+/ ) {
    print $str, "\n";
 }
}
```

Parens

Can provide some flexibility in the match with parens and the groups

```
 if ($word =~ m/fire (alarm|engine)/) {…} # ‘fire alarm’ or ‘fire engine’
if ($sport =~ m/(basket|foot)ball/) {…} # basketball or football
if ($name =~ m/Ste(v|ph)en/) {…} # Steven or Stephen

 # method 1
 print “Chili pepper\n” if ($text =~ m/chili|chilli|chillie|chile/i);
 # method 2
 print “Chili pepper\n” if ($text =~ m/chil(i|li|lie|e)/i);
```

Parens to define units

```
my $str = "AAGATCTCTCGATCTGAA";
if( $str =~ /((TC)+)/ ) {
    print "match is $1 of unit $2\n";
}
```

produces: `match is TCTCTC of unit TC`

If you wanted to match a unit multiple times, first set of parens defines the capture the second (TC) would be matched multiple times.

This is different from

```
my $str = "AAGATCTCTCCTGATCTGAA";
if( $str =~ /([TC]+)/ ) {
    print "match is $1\n";
}
```

produces: `match is TCTCTCCT`