Looking for a pattern in a string

• Checking string for complete and exact match:

```
$bob = "Willis";
if ($bob eq "Willis") { print "Match!\n";
}
```

• Checking string for partial match:

```
$bob = "Wachu talkin bout Willis?";
if ($bob =~ /Willis/) { print "Match!\n";
}
```

• Checking string complex pattern:

```
$bob = "Wachu talkin bout Willis?";
if ($bob =~ /^\[Ww\]achu\s+talkin\(g\)\{0,1\}/ { print "Match!\n";
}
```
The Basics

- Test strings for equality with another with the eq operator.
- Test strings for a match to a pattern with the =~ operator.
  - Put the patterns to be matched in slashes
    - /Pattern/
- eq and =~ evaluate to true or false
- Patterns can contain special 'regular expression' characters.
Regular Expression Characters

- Match one of a group of characters
  - Put things in square brackets
    - /[ACGTacgt]/
  - Use the – for a range
    - /[A-Z]/
  - Start the group of characters with a ^ to mean anything but these characters
    - /[^qrstuv]/
Regular Expression
Characters 2

- \d    A digit 0,1,2,3,4, etc.
  - [0-9]

- \w    A word character
  - [A-Za-z_0-9]

- \s    Whitespace
  - [ \t\n\r\f]

- .    Any one character
Regular Expression Characters 3

• $  End of the string or variable
  - End of the string if it is last
    • As in /Willis$/>
  - Whatever is in a variable if it isn't last
    • As in /$name/"

• ^ The beginning of a string
  - /\^Wachu talkin/"
Try it yourself

- Copy the program called day3.pl and add the part that prints the
  - name with the initials JP
  - name with the number in it
  - name that starts with A,L, or B
  - first name that is four letters long

- day3.pl is available in the TA home directory:
  - cp ~bio01/day3.pl ./
Regular Expression Quantifiers

- **+** One or more of a thing
  - `$name =~ /q+/;`
    - Do you have one or more q's in your name?

- ***** Zero or more of a thing
  - `$word =~ /trans.*tion/;`
    - Matches translation or transcription

- **{}** specify a range of how many
  - `$word =~ /go{2,7}gle/;`
    - Matches everything from google to gooooooooolge
More Quantifier Tricks

- `$word =~ /go{,7}gle/;`     # 7 or fewer
- `$word =~ /go{2,}gle/;`     # 2 or more
- `$word =~ /go{6}gle/;`      # exactly 6
Save time with patterns

- **split** is a function that chops up a string into a list.
  
  ```perl
  $names = "bob, joe, fred";
  @words = split(\/, \/, $names);
  ```

- **join** is a function that can put them back together again
  
  ```perl
  $names = join("\t", @words);
  ```
Try it yourself

• Copy day3.2.pl to your directory and change it to print:
  - genes from human or mouse
  - genes with between 3 and 7 A's in row
  - genes with less than 4 G's between TACGAC and TCGCCCC
  - genes with one or more A's after TCGCCCC
  - genes that don't have any N's in them
tr/ / / and s/ / /

- tr/ / / and s/ / / are handy functions for editing strings
- s/ / / substitutes one pattern for another
  - $name =~ s/Duck/Trump/;
    - “Donald Duck” becomes “Donald Trump”
- tr/ / / translates one thing for another
  - $sequence =~ tr/Tt/Uu/g;
    - DNA becomes RNA
  - $sequence =~ tr/acgtACGT/tgcaTGCA/g;
    - complement, a becomes t, c becomes g, etc.
    - Still need to reverse for reverse complement
Options for Pattern Matching

- You can modify the action of pattern matching functions
  - $\text{species} =~ /\text{human}/i$;
    - The i means ignore case
      - Human, human, and HuMan all match
  - $\text{sequence} =~ \text{tr/acgtACGT/tgcaTGCA/g}$;
    - The g means globally
    - Change all matches in $\text{sequence}$, not just the first
Grabbing parts of a string

- When matching a pattern, parentheses will grab the parts of the string they match and put them in some 'special' variables

```
$person = "First: Joe Last: Smith";
$person =~ /First:(\[A-Za-z\]+) Last: (\[A-Za-z\]+)/;
$first = $1;
$last = $2;
```

- $1, $2, etc. are special read only variables that get the contents of the first, second, etc. parenthesis
Tricks with $1 and $2

- The quantifiers * and + are 'greedy'
- They will match as much as they can unless you tell them not to with the ? character

```
$person =~ /First: (.*)\s/;
#matches “Joe Last: “ or everything until the last space
$person =~ /First: (.*)\s/;
#matches “Joe”
```