

# A Whirlwind Tour

The Whole Course in 20 Minutes

# Where Are We Going?

- Let's write a real program
- Use only techniques we'll learn in this class
- Show you where we're headed



# Eliza

```
dtreder% perl eliza.pl
```

Hello, I'm Elly. What would you like to discuss today?

my coffee

Who else is involved with that.

The barista burned it, I think. Perhaps she doesn't like my face.

Why do you think your face ?

Well, she can't see my legs.

When did you start noticing that.

Oh, back in 1988.

Tell me more about oh back in 1988 ?

That was high school for me.

Who else is involved with you ?

I'm innocent, I swear!

Why do you think that.

I'm a good person at heart.

Tell me more about you're a good person at heart ?

Well I haven't killed anybody. yet.

Why do you think you havent killed anybody yet

They're all still breathing.

Why don't we talk about your mother?

Alright, I confess! Professor Plum with the candlestick in the studio!

Tell me more about you confess...

# Saying Hi

```
#!/usr/bin/perl
```

```
# Saying Hi
```

```
print "Hello, I'm Elly.  What would you like  
to discuss today?\n";
```

```
# \n - inside double quotes - means newline
```

# Saying Hi Back

```
# Get input from user
my $input = <>;

# introduce a new variable with my

# scalar variable $input holds a string
print "Why do you think $input ?\n";

# You can print variables too
```

# Never Stop Trying

```
# loop as long as 1 is true (it always is!)
while (1) {
    my $input = <>;

    print "Why do you think $input ?\n";
}
```

# Never Stop Trying

```
while (1) {  
  my $input = <>;  
  # remove newline that the user typed  
  chomp $input;  
  
  print "Why do you think $input ?\n";  
}
```

# I can say lots of things

```
# An array of some canned responses
my @canned = ('Tell me more about',
              'Why do you think',
              'What else is going on with',
              'Who else is involved with',
              'When did you start noticing',
              'How did you feel about');
```

```
# you can also put strings in single quotes
```

# Polly Wanna Cracker

```
while (1) {  
    my $input = <>;  
  
    chomp $input;  
  
    # a number from 0 to 5  
    my $i = rand(6);  
    # access one bucket of an array  
    print $canned[$i];  
}
```

# Fake Like I'm Awake

```
while (1) {  
    my $input = <>;  
  
    chomp $input;  
  
    my $i = rand(6);  
    # separate things to print with commas  
    print $canned[$i], " $input ?\n";  
    # prints "Tell me more about $input?"  
}
```

# We Must Discuss

```
my $count = 0;

while (1) {
    my $input = <>;
    chomp $input;

    $count++;
    if ($count > 10) {
        print "Why don't we talk about your mother?\n";
        next;
    }
}
```

# But Only Once

```
my $count = 0;
my $parents = 0;

while (1) {
    my $input = <>;
    chomp $input;

    $count++;
    if ($count > 10 && ! $parents) {
        print "Why don't we talk about your mother?\n";
        $parents = 1;
        next;
    }
}
```

# I Already Told You!

```
my $count = 0;
my $parents = 0;

while (1) {
    my $input = <>;
    chomp $input;

    # =~ m! ! means matches this regular expression
    if ($input =~ m!\b(mom|mamma|mother)\b!) {
        $parents = 1;
    }

    $count++;
    if ($count > 10 && ! $parents) {
        print "Why don't we talk about your mother?\n";
    }
}
```

# Clean Up Your Act

```
while (1) {  
  my $input = <>;  
  chomp $input;  
  
  # put it in lowercase  
  $input = lc($input);  
  # get rid of all punctuation  
  $input =~ s! [^\w\s] !!gx;  
  
  # \w is "word characters" - 0-9, A-Z, _  
  # \s is "whitespace" - space, tab, newline  
  # [^ ] means everything that isn't  
  # g means G0: over and over  
  # x means whitespace isn't taken literally
```

# Clean Up Your Act

```
while (1) {  
  my $input = <>;  
  chomp $input;  
  
  $input =~ s![^\w\s]!!g;  
  $input = lc($input);  
  
  # user didn't type anything!  
  if (! $input) {  
    print "I'd like to hear what you think.\n";  
    next;  
  }  
}
```

# It's Not You, It's Me

```
# Change you -> me and vice versa,  
# leaving a marker (#) to show those changed  
$input =~ s! \b you \b !#me!gx;  
$input =~ s! \b your !#my!gx;  
$input =~ s! \b you're !#I am!gx;  
$input =~ s! \b my !#your!gx;  
$input =~ s! \b mine \b !#yours!gx;  
$input =~ s! \b me \b !#you!gx;  
$input =~ s! \b i \b !#you!gx;  
$input =~ s! \b im \b !#you're!gx;  
  
# remove all the markers  
$input =~ s!#!!g;
```

# Congratulations!

- We covered:
  - Scalars and Arrays
  - Input and Output
  - Control structures such as if and while
  - Regular expressions and substitutions



# Other Stuff

- Things we didn't cover here, but we'll still do:
  - Hashes
  - Files on disk
  - Dates and times
  - Subroutines

