

# ECOL 553L

Section 6  
Intro to Perl I

# Turning in your homework

# Turning in your homework

- Make a folder in your home directory named `eco1553_homework`

# Turning in your homework

- Make a folder in your home directory named `eco1553_homework`
- within this folder, make a folder called `homework1`

# Turning in your homework

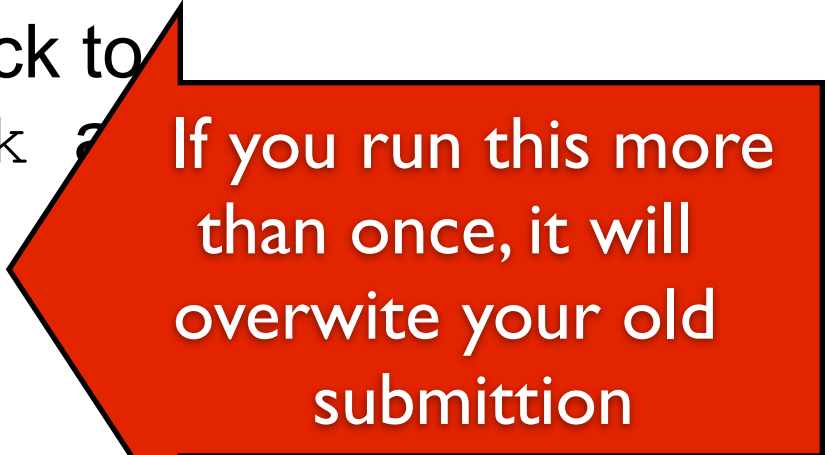
- Make a folder in your home directory named `eco1553_homework`
- within this folder, make a folder called `homework1`
- make a file within this folder called `<last_name>_<first_initial>.txt` and put the homework answers in it

# Turning in your homework

- Make a folder in your home directory named `ecol553_homework`
- within this folder, make a folder called `homework1`
- make a file within this folder called `<last_name>_<first_initial>.txt` and put the homework answers in it
- when finished move back to `~/ecol553_homework` and run `turnin homework1`

# Turning in your homework

- Make a folder in your home directory named `ecol553_homework`
- within this folder, make a folder called `homework1`
- make a file within this folder called `<last_name>_<first_initial>.txt` and put the homework answers in it
- when finished move back to `~/ecol553_homework` and run `turnin homework1`



If you run this more than once, it will overwrite your old submission

# Homework 2



# Homework 2

- The homework for this week has been updated, please re-download it out of the “class5” folder!!

# Homework 2

- The homework for this week has been updated, please re-download it out of the “class5” folder!!
- These slides have also been updated if you downloaded them before class.

# Homework 2

- The homework for this week has been updated, please re-download it out of the “class5” folder!!
- These slides have also been updated if you downloaded them before class.
- Some things need to be changed in the submit `csh` file

# Homework 2

- The homework for this week has been updated, please re-download it out of the “class5” folder!!
- These slides have also been updated if you downloaded them before class.
- Some things need to be changed in the submit `csh` file
  - `-q standard (or windfall)`

# Homework 2

- The homework for this week has been updated, please re-download it out of the “class5” folder!!
- These slides have also been updated if you downloaded them before class.
- Some things need to be changed in the submit `csh` file
  - `-q standard (or windfall)`
  - delete the “`source . . . .`” line

# Homework 2

- The homework for this week has been updated, please re-download it out of the “class5” folder!!
- These slides have also been updated if you downloaded them before class.
- Some things need to be changed in the submit `ssh` file
  - `-q standard (or windfall)`
  - delete the “`source . . . .`” line
  - `cd eeb553_ . . . .` should contain the full path to your homework folder

# Tree Structure Review

# Tree Structure Review

- You're in `/tmp/eco1553`



# Tree Structure Review

- You're in `/tmp/eco1553`
  - What is the full path of `.` ?

# Tree Structure Review

- You're in `/tmp/eco1553`
  - What is the full path of `.` ?
    - `/tmp/eco1553`

# Tree Structure Review

- You're in `/tmp/eco1553`
  - What is the full path of `.` ?
    - `/tmp/eco1553`
  - What is the full path of `..` ?

# Tree Structure Review

- You're in `/tmp/eco1553`
  - What is the full path of `.` ?
    - `/tmp/eco1553`
  - What is the full path of `..` ?
    - `/tmp/`

# Tree Structure Review

- You're in `/tmp/eco1553`
  - What is the full path of `.` ?
    - `/tmp/eco1553`
  - What is the full path of `..` ?
    - `/tmp/`
  - What is the full path of `~` ?

# Tree Structure Review

- You're in `/tmp/eco1553`
  - What is the full path of `.` ?
    - `/tmp/eco1553`
  - What is the full path of `..` ?
    - `/tmp/`
  - What is the full path of `~` ?
    - mine is `/home/u21/deblasio`

# Tree Structure Review

- You're in `/tmp/eco1553`
  - What is the full path of `.` ?
    - `/tmp/eco1553`
  - What is the full path of `..` ?
    - `/tmp/`
  - What is the full path of `~` ?
    - mine is `/home/u21/deblasio`
  - What is the full path of `../.hidden` ?

# Tree Structure Review

- You're in `/tmp/eco1553`
  - What is the full path of `.` ?
    - `/tmp/eco1553`
  - What is the full path of `..` ?
    - `/tmp/`
  - What is the full path of `~` ?
    - mine is `/home/u21/deblasio`
  - What is the full path of `../.hidden` ?
    - `/tmp/.hidden`



# Tree Structure Review

- You're in `/tmp/eco1553`
  - What is the full path of `.` ?
    - `/tmp/eco1553`
  - What is the full path of `..` ?
    - `/tmp/`
  - What is the full path of `~` ?
    - mine is `/home/u21/deblasio`
  - What is the full path of `../.hidden` ?
    - `/tmp/.hidden`
  - What is the full path of `~/.hidden`?

# Tree Structure Review

- You're in `/tmp/eco1553`
  - What is the full path of `.` ?
    - `/tmp/eco1553`
  - What is the full path of `..` ?
    - `/tmp/`
  - What is the full path of `~` ?
    - mine is `/home/u21/deblasio`
  - What is the full path of `../.hidden` ?
    - `/tmp/.hidden`
  - What is the full path of `~/.hidden` ?
    - mine is `/home/u21/deblasio/.hidden`

# Tree Structure Review

# Tree Structure Review

- You're in ~

# Tree Structure Review

- You're in ~
  - mine is `/home/u21/deblasio`

# Tree Structure Review

- You're in ~
  - mine is `/home/u21/deblasio`
- What is the full path of `.` ?

# Tree Structure Review

- You're in ~
  - mine is `/home/u21/deblasio`
- What is the full path of `.` ?
  - `/home/u21/deblasio`

# Tree Structure Review

- You're in ~
  - mine is `/home/u21/deblasio`
- What is the full path of `.` ?
  - `/home/u21/deblasio`
- What is the full path of `..` ?



# Tree Structure Review

- You're in ~
  - mine is `/home/u21/deblasio`
- What is the full path of `.` ?
  - `/home/u21/deblasio`
- What is the full path of `..` ?
  - `/home/u21/`

# Tree Structure Review

- You're in ~
  - mine is `/home/u21/deblasio`
- What is the full path of `.` ?
  - `/home/u21/deblasio`
- What is the full path of `..` ?
  - `/home/u21/`
- What is the full path of `~` ?

# Tree Structure Review

- You're in ~
  - mine is `/home/u21/deblasio`
- What is the full path of `.` ?
  - `/home/u21/deblasio`
- What is the full path of `..` ?
  - `/home/u21/`
- What is the full path of `~` ?
  - mine is `/home/u21/deblasio`

# Tree Structure Review

- You're in ~
  - mine is `/home/u21/deblasio`
- What is the full path of `.` ?
  - `/home/u21/deblasio`
- What is the full path of `..` ?
  - `/home/u21/`
- What is the full path of `~` ?
  - mine is `/home/u21/deblasio`
- What is the difference between `~/file`, `./file`, and `../deblasio/file` ?

# Tree Structure Review

- You're in ~
  - mine is `/home/u21/deblasio`
- What is the full path of `.` ?
  - `/home/u21/deblasio`
- What is the full path of `..` ?
  - `/home/u21/`
- What is the full path of `~` ?
  - mine is `/home/u21/deblasio`
- What is the difference between `~/file`, `./file`, and `../deblasio/file` ?
- Whats the difference between `~/ .hidden` and `~/hidden` ?

# Using wc

# Using wc

- wc = word count

# Using `wc`

- `wc` = word count
- returns the number of words, lines, and bytes in a file in tab-delimited format



# Using wc

- wc = word count
- returns the number of words, lines, and bytes in a file in tab-delimited format

A **word** is defined as a run of characters between whitespace (space, tab, newline)

# Using wc

- wc = word count
- returns the number of words, lines, and bytes in a file in tab-delimited format

A **line** is defined as the characters before a newline (i.e. the line count is the number of newlines)

# Using `wc`

- `wc` = word count
- returns the number of words, lines, and bytes in a file in tab-delimited format

A **byte** is 8 *bits*, or the amount of space necessary to store one character. So the byte count of a text file is the number of characters (including newlines, tabs, and spaces)

# Using wc

- wc = word count
- returns the number of words, lines, and bytes in a file in tab-delimited format

**tab-delimited**, or **tab-separated** format is a way of representing data where each record is on a new line, and entry related to that record is separated by a tab ('\t'). An alternate is **comma-separated (csv)**.

# Using wc

- wc = word count
- returns the number of words, lines, and bytes in a file in tab-delimited format

```
wc [<options>] <fileList>
```

# Using wc

- wc = word count
- returns the number of words, lines, and bytes in a file in tab-delimited format

```
wc [<options>] <fileList>
```

- a useful option is:

# Using wc

- wc = word count
- returns the number of words, lines, and bytes in a file in tab-delimited format

```
wc [<options>] <fileList>
```

- a useful option is:
  - -l counts only the number lines

Perl



# Introduction to Perl, Part I

# Introduction to Perl, Part I

- Today's Topics:

# Introduction to Perl, Part I

- Today's Topics:
  - What is Perl?

# Introduction to Perl, Part I

- Today's Topics:
  - What is Perl?
  - Links to Perl Books and Tutorials

# Introduction to Perl, Part I

- Today's Topics:
  - What is Perl?
  - Links to Perl Books and Tutorials
  - Installing Perl on your own computer

# Introduction to Perl, Part I

- Today's Topics:
  - What is Perl?
  - Links to Perl Books and Tutorials
  - Installing Perl on your own computer
  - Perl-aware Text editors:

# Introduction to Perl, Part I

- Today's Topics:
  - What is Perl?
  - Links to Perl Books and Tutorials
  - Installing Perl on your own computer
  - Perl-aware Text editors:
    - vim, syn for Windows, TextWrangler for Mac

# Introduction to Perl, Part I

- Today's Topics:
  - What is Perl?
  - Links to Perl Books and Tutorials
  - Installing Perl on your own computer
  - Perl-aware Text editors:
    - vim, syn for Windows, TextWrangler for Mac
  - Using the Unix which command to find the perl interpreter



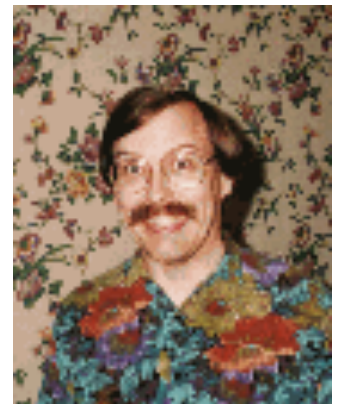
# Introduction to Perl, Part I

- Today's Topics:
  - What is Perl?
  - Links to Perl Books and Tutorials
  - Installing Perl on your own computer
  - Perl-aware Text editors:
    - vim, syn for Windows, TextWrangler for Mac
  - Using the Unix which command to find the perl interpreter
  - First Steps in Perl

# Introduction to Perl, Part I

- Today's Topics:
  - What is Perl?
  - Links to Perl Books and Tutorials
  - Installing Perl on your own computer
  - Perl-aware Text editors:
    - vim, syn for Windows, TextWrangler for Mac
  - Using the Unix which command to find the perl interpreter
  - First Steps in Perl
    - print; doing arithmetic; comments

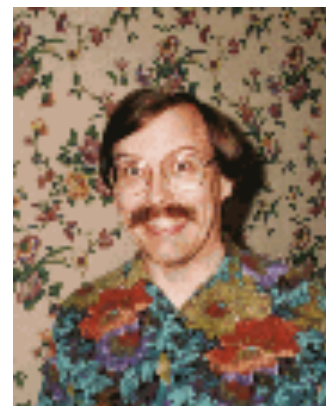
# What is Perl?



Perl Author Larry Wall

# What is Perl?

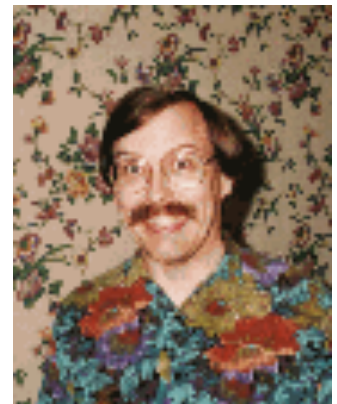
- Practical Extraction and Report Language



Perl Author Larry Wall

# What is Perl?

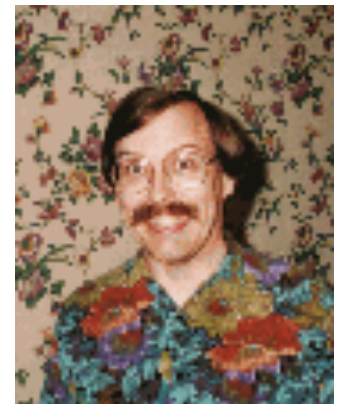
- **PRACTICAL EXTRACTION AND REPORT LANGUAGE**
- Author: Larry Wall <http://www.wall.org/~larry/>



Perl Author Larry Wall

# What is Perl?

- **P**ractical **E**xtraction and **R**eport **L**anguage
- Author: Larry Wall <http://www.wall.org/~larry/>
- Perl is a scripting language: designed to promote rapid solutions to problems in data processing and manipulation



Perl Author Larry Wall

# What is Perl?

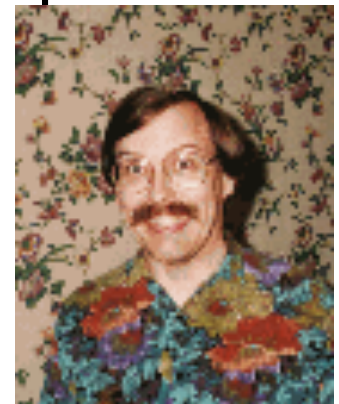
- **P**RACTICAL **E**XTRACTION and **R**EPORT **L**ANGUAGE
- Author: Larry Wall <http://www.wall.org/~larry/>
- Perl is a scripting language: designed to promote rapid solutions to problems in data processing and manipulation
- Makes "easy things easy and hard things possible"



Perl Author Larry Wall

# What is Perl?

- **PRACTICAL EXTRACTION and REPORT LANGUAGE**
- Author: Larry Wall <http://www.wall.org/~larry/>
- Perl is a scripting language: designed to promote rapid solutions to problems in data processing and manipulation
- Makes "easy things easy and hard things possible"
- Freely available on many platforms

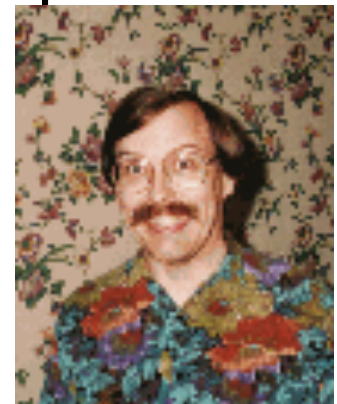


Perl Author Larry Wall



# What is Perl?

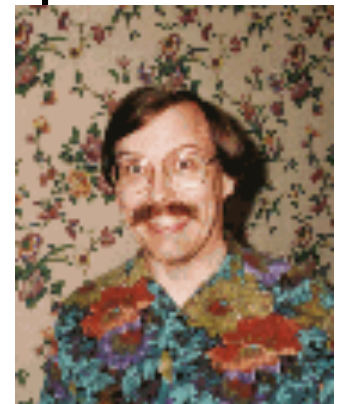
- **PRACTICAL EXTRACTION and REPORT LANGUAGE**
- Author: Larry Wall <http://www.wall.org/~larry/>
- Perl is a scripting language: designed to promote rapid solutions to problems in data processing and manipulation
- Makes "easy things easy and hard things possible"
- Freely available on many platforms
- Well suited to Life Sciences data:



Perl Author Larry Wall

# What is Perl?

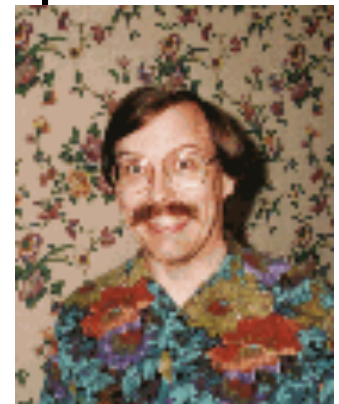
- **P**ractical **E**xtraction and **R**eport **L**anguage
- Author: Larry Wall <http://www.wall.org/~larry/>
- Perl is a scripting language: designed to promote rapid solutions to problems in data processing and manipulation
- Makes "easy things easy and hard things possible"
- Freely available on many platforms
- Well suited to Life Sciences data:
  - pattern matching capabilities



Perl Author Larry Wall

# What is Perl?

- **Practical Extraction and Report Language**
- Author: Larry Wall <http://www.wall.org/~larry/>
- Perl is a scripting language: designed to promote rapid solutions to problems in data processing and manipulation
- Makes "easy things easy and hard things possible"
- Freely available on many platforms
- Well suited to Life Sciences data:
  - pattern matching capabilities
  - BioPerl modules



Perl Author Larry Wall

# Learning Perl

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:



# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs
  - <http://www.perl.org/books/library.html>

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs
  - <http://www.perl.org/books/library.html>
    - Includes 'Impatient Perl', 'Picking Up Perl', & more

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs
  - <http://www.perl.org/books/library.html>
    - Includes 'Impatient Perl', 'Picking Up Perl', & more

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs
  - <http://www.perl.org/books/library.html>
    - Includes 'Impatient Perl', 'Picking Up Perl', & more
- Quicktime tutorials:

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs
  - <http://www.perl.org/books/library.html>
    - Includes 'Impatient Perl', 'Picking Up Perl', & more
- Quicktime tutorials:
  - <http://uacbt.arizona.edu/>

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs
  - <http://www.perl.org/books/library.html>
    - Includes 'Impatient Perl', 'Picking Up Perl', & more
- Quicktime tutorials:
  - <http://uacbt.arizona.edu/>



# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs
  - <http://www.perl.org/books/library.html>
    - Includes 'Impatient Perl', 'Picking Up Perl', & more
- Quicktime tutorials:
  - <http://uacbt.arizona.edu/>
- Don't forget to try Google:

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs
  - <http://www.perl.org/books/library.html>
    - Includes 'Impatient Perl', 'Picking Up Perl', & more
- Quicktime tutorials:
  - <http://uacbt.arizona.edu/>
- Don't forget to try Google:
  - perl arrays

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs
  - <http://www.perl.org/books/library.html>
    - Includes 'Impatient Perl', 'Picking Up Perl', & more
- Quicktime tutorials:
  - <http://uacbt.arizona.edu/>
- Don't forget to try Google:
  - perl arrays
  - perl function split

# Learning Perl

- We'll be using the online book "Beginning Perl" by Simon Cozens
  - <http://www.perl.org/books/beginning-perl/>
    - In particular, chapters 1-6,8,10, 13
- There are of course many other sources for learning Perl:
  - [http://www.perlmeme.org/start\\_here/index.html](http://www.perlmeme.org/start_here/index.html)
    - Tutorials, HowTos, FAQs
  - <http://www.perl.org/books/library.html>
    - Includes 'Impatient Perl', 'Picking Up Perl', & more
- Quicktime tutorials:
  - <http://uacbt.arizona.edu/>
- Don't forget to try Google:
  - perl arrays
  - perl function split
  - perl regular expressions

# Installing Perl and a Perl-aware text editor

# Installing Perl and a Perl-aware text editor

- Perl is included in Linux and MacOSX systems

# Installing Perl and a Perl-aware text editor

- Perl is included in Linux and MacOSX systems
  - Some configuration of the environment may be needed

# Installing Perl and a Perl-aware text editor

- Perl is included in Linux and MacOSX systems
  - Some configuration of the environment may be needed



# Installing Perl and a Perl-aware text editor

- Perl is included in Linux and MacOSX systems
  - Some configuration of the environment may be needed
- For Macs, the Textwrangler editor can be downloaded from <http://www.barebones.com/products/textwrangler/>

# Installing Perl and a Perl-aware text editor

- Perl is included in Linux and MacOSX systems
  - Some configuration of the environment may be needed
- For Macs, the Textwrangler editor can be downloaded from <http://www.barebones.com/products/textwrangler/>

# Installing Perl and a Perl-aware text editor

- Perl is included in Linux and MacOSX systems
  - Some configuration of the environment may be needed
- For Macs, the Textwrangler editor can be downloaded from <http://www.barebones.com/products/textwrangler/>
- For Windows, Perl can be freely downloaded from <http://www.activestate.com/activeperl/downloads>

# Installing Perl and a Perl-aware text editor

- Perl is included in Linux and MacOSX systems
  - Some configuration of the environment may be needed
- For Macs, the Textwrangler editor can be downloaded from <http://www.barebones.com/products/textwrangler/>
- For Windows, Perl can be freely downloaded from <http://www.activestate.com/activeperl/downloads>

# Installing Perl and a Perl-aware text editor

- Perl is included in Linux and MacOSX systems
  - Some configuration of the environment may be needed
- For Macs, the Textwrangler editor can be downloaded from <http://www.barebones.com/products/textwrangler/>
- For Windows, Perl can be freely downloaded from <http://www.activestate.com/activeperl/downloads>
- For Windows, the syn editor can be downloaded from <http://sourceforge.net/projects/syn>

# Installing Perl and a Perl-aware text editor

- Perl is included in Linux and MacOSX systems
  - Some configuration of the environment may be needed
- For Macs, the Textwrangler editor can be downloaded from <http://www.barebones.com/products/textwrangler/>
- For Windows, Perl can be freely downloaded from <http://www.activestate.com/activeperl/downloads>
- For Windows, the syn editor can be downloaded from <http://sourceforge.net/projects/syn>
  - Perl keywords and syntax constructs are highlighted in different colors

# Installing Perl and a Perl-aware text editor

- Perl is included in Linux and MacOSX systems
  - Some configuration of the environment may be needed
- For Macs, the Textwrangler editor can be downloaded from <http://www.barebones.com/products/textwrangler/>
- For Windows, Perl can be freely downloaded from <http://www.activestate.com/activeperl/downloads>
- For Windows, the syn editor can be downloaded from <http://sourceforge.net/projects/syn>
  - Perl keywords and syntax constructs are highlighted in different colors
  - Highlights help with coding and debugging

# The big picture...how scripting works



# The big picture...how scripting works

- Perl scripts are text files...

# The big picture...how scripting works

- Perl scripts are text files...
  - Commonly named with a .pl extension

# The big picture...how scripting works

- Perl scripts are text files...
  - Commonly named with a .pl extension
  - The scripts are stored on disk. When you run them, their code and variables occupy space in memory.

# The big picture...how scripting works

- Perl scripts are text files...
  - Commonly named with a .pl extension
  - The scripts are stored on disk. When you run them, their code and variables occupy space in memory.
  - Scripts can read/write disk files, but these too are buffered through memory.

# The big picture...how scripting works

- Perl scripts are text files...
  - Commonly named with a .pl extension
  - The scripts are stored on disk. When you run them, their code and variables occupy space in memory.
  - Scripts can read/write disk files, but these too are buffered through memory.
  - To run scripts on Windows, you use Start→Run and enter 'cmd'. On a Mac, open the hard drive and search for 'terminal'. Inside the command/terminal window, you can check the perl version by typing:

# The big picture...how scripting works

- Perl scripts are text files...
  - Commonly named with a .pl extension
  - The scripts are stored on disk. When you run them, their code and variables occupy space in memory.
  - Scripts can read/write disk files, but these too are buffered through memory.
  - To run scripts on Windows, you use Start→Run and enter 'cmd'. On a Mac, open the hard drive and search for 'terminal'. Inside the command/terminal window, you can check the perl version by typing:
    - perl -v

# The big picture...how scripting works

- Perl scripts are text files...
  - Commonly named with a .pl extension
  - The scripts are stored on disk. When you run them, their code and variables occupy space in memory.
  - Scripts can read/write disk files, but these too are buffered through memory.
  - To run scripts on Windows, you use Start→Run and enter 'cmd'. On a Mac, open the hard drive and search for 'terminal'. Inside the command/terminal window, you can check the perl version by typing:
    - `perl -v`
- How can you change a script?

# The big picture...how scripting works

- Perl scripts are text files...
  - Commonly named with a .pl extension
  - The scripts are stored on disk. When you run them, their code and variables occupy space in memory.
  - Scripts can read/write disk files, but these too are buffered through memory.
  - To run scripts on Windows, you use Start→Run and enter 'cmd'. On a Mac, open the hard drive and search for 'terminal'. Inside the command/terminal window, you can check the perl version by typing:
    - `perl -v`
- How can you change a script?
  - Use a text editor, such as vim, syn, textwrangler (or any other Perl-aware editor)



# Running a Perl Script

# Running a Perl Script

- `perl <script.pl>`

# Running a Perl Script

- `perl <script.pl>`
- `./<script.pl>`

# Running a Perl Script

- `perl <script.pl>`
- `./<script.pl>`
  - requires you to enable execute permissions

# Running a Perl Script

- `perl <script.pl>`
- `./<script.pl>`
  - requires you to enable execute permissions
  - `chmod` (change file modes)

# Running a Perl Script

- `perl <script.pl>`
- `./<script.pl>`
  - requires you to enable execute permissions
  - `chmod (change file modes)`
    - `chmod +x <script.pl>`

# Running a Perl Script

- `perl <script.pl>`
- `./<script.pl>`
  - requires you to enable execute permissions
  - `chmod` (change file modes)
    - `chmod +x <script.pl>`
      - option can be `[+-][rwx]` or

# Running a Perl Script

- `perl <script.pl>`
- `./<script.pl>`
  - requires you to enable execute permissions
  - `chmod` (change file modes)
    - `chmod +x <script.pl>`
      - option can be `[+-][rwx]` or
    - `chmod 700 <script.pl>`



# Running a Perl Script

- `perl <script.pl>`
- `./<script.pl>`
  - requires you to enable execute permissions
  - `chmod` (change file modes)
    - `chmod +x <script.pl>`
      - option can be `[+-][rwx]` or
    - `chmod 700 <script.pl>`
      - 3 digits between 0 (000b) and 7 (111b) representing binary indications of rwx for user, group everyone

# Running a Perl Script

- `perl <script.pl>`
- `./<script.pl>`
  - requires you to enable execute permissions
  - `chmod` (change file modes)
    - `chmod +x <script.pl>`
      - option can be `[+-][rwx]` or
    - `chmod 700 <script.pl>`
      - 3 digits between 0 (000b) and 7 (111b) representing binary indications of rwx for user, group everyone
        - 700 => read, write and execute for only you, nothing else for anyone else

# Perl Scripting Tidbits

# Perl Scripting Tidbits

- Perl's motto: TIMTOWTDI "tim-too-dee"  
(there is more than one way to do it!)

# Perl Scripting Tidbits

- Perl's motto: TIMTOWTDI "tim-too-dee"  
(there is more than one way to do it!)
- Scripts are generally interpreted line by line

# Perl Scripting Tidbits

- Perl's motto: TIMTOWTDI "tim-too-dee"  
(there is more than one way to do it!)
- Scripts are generally interpreted line by line
- Correct syntax is crucial – as is correct logic

# Perl Scripting Tidbits

- Perl's motto: TIMTOWTDI "tim-too-dee"  
(there is more than one way to do it!)
- Scripts are generally interpreted line by line
- Correct syntax is crucial – as is correct logic
- Error messages are terse (not much information given) but they can still be helpful if you read them carefully

# Perl Scripting Tidbits

- Perl's motto: TIMTOWTDI "tim-too-dee"  
(there is more than one way to do it!)
- Scripts are generally interpreted line by line
- Correct syntax is crucial – as is correct logic
- Error messages are terse (not much information given) but they can still be helpful if you read them carefully
- Sometimes the line number in an error message reflects a line past the actual error – so you will want to check lines above the reported line number!!



# How to think like a Programmer

# How to think like a Programmer

- Q: Is the glass half empty or half full?

# How to think like a Programmer

- Q: Is the glass half empty or half full?
- A: Yes!

# How to think like a Programmer



- Q: Is the glass half empty or half full?
- A: Yes!

# The Unix which command

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - `/bin`

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - `/bin`
  - `/usr/bin`



# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - `/bin`
  - `/usr/bin`
  - `/usr/local/bin`

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - `/bin`
  - `/usr/bin`
  - `/usr/local/bin`
  - `/genome/ICEbin` ...

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - `/bin`
  - `/usr/bin`
  - `/usr/local/bin`
  - `/genome/ICEbin` ...
- When writing a Perl script, it's useful to include the location of the Perl interpreter in the script

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - `/bin`
  - `/usr/bin`
  - `/usr/local/bin`
  - `/genome/ICEbin` ...
- When writing a Perl script, it's useful to include the location of the Perl interpreter in the script
- We can find the Perl interpreter with the command:

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - `/bin`
  - `/usr/bin`
  - `/usr/local/bin`
  - `/genome/ICEbin` ...
- When writing a Perl script, it's useful to include the location of the Perl interpreter in the script
- We can find the Perl interpreter with the command:
  - `which perl`

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - /bin
  - /usr/bin
  - /usr/local/bin
  - /genome/ICEbin ...
- When writing a Perl script, it's useful to include the location of the Perl interpreter in the script
- We can find the Perl interpreter with the command:
  - which perl
  - [service2][~]> which perl

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - /bin
  - /usr/bin
  - /usr/local/bin
  - /genome/ICEbin ...
- When writing a Perl script, it's useful to include the location of the Perl interpreter in the script
- We can find the Perl interpreter with the command:
  - which perl
  - [service2][~]> which perl
  - /usr/bin/perl

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - /bin
  - /usr/bin
  - /usr/local/bin
  - /genome/ICEbin ...
- When writing a Perl script, it's useful to include the location of the Perl interpreter in the script
- We can find the Perl interpreter with the command:
  - which perl
  - [service2][~]> which perl
  - /usr/bin/perl
- The which command is useful in determining whether a program has been installed, for example:



# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - /bin
  - /usr/bin
  - /usr/local/bin
  - /genome/ICEbin ...
- When writing a Perl script, it's useful to include the location of the Perl interpreter in the script
- We can find the Perl interpreter with the command:
  - which perl
  - [service2][~]> which perl
  - /usr/bin/perl
- The which command is useful in determining whether a program has been installed, for example:
  - [service2][~]> which repeatmasker

# The Unix which command

- Programs executed on a Unix/Linux system can reside in a variety of locations:
  - /bin
  - /usr/bin
  - /usr/local/bin
  - /genome/ICEbin ...
- When writing a Perl script, it's useful to include the location of the Perl interpreter in the script
- We can find the Perl interpreter with the command:
  - which perl
  - [service2][~]> which perl
  - /usr/bin/perl
- The which command is useful in determining whether a program has been installed, for example:
  - [service2][~]> which repeatmasker
  - repeatmasker: Command not found.

# Anatomy of a Perl script, part I

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32"

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)



# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module
  - `use warnings;`

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module
  - `use warnings;`
  - `#` The warnings module is helpful in pointing out errors – use it!!

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module
  - `use warnings;`
  - `#` The warnings module is helpful in pointing out errors – use it!!
  - `#` Notice the semi-colon at the end of the line

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module
  - `use warnings;`
  - `#` The warnings module is helpful in pointing out errors – use it!!
  - `#` Notice the semi-colon at the end of the line
  - `#` nearly all lines of code need `;` as a terminator – don't forget this!!

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module
  - `use warnings;`
  - `#` The warnings module is helpful in pointing out errors – use it!!
  - `#` Notice the semi-colon at the end of the line
  - `#` nearly all lines of code need `;` as a terminator – don't forget this!!
- An alternative to the warnings module: `strict`

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module
  - `use warnings;`
  - `#` The warnings module is helpful in pointing out errors – use it!!
  - `#` Notice the semi-colon at the end of the line
  - `#` nearly all lines of code need `;` as a terminator – don't forget this!!
- An alternative to the warnings module: strict
  - `use strict;`

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module
  - `use warnings;`
  - `#` The warnings module is helpful in pointing out errors – use it!!
  - `#` Notice the semi-colon at the end of the line
  - `#` nearly all lines of code need `;` as a terminator – don't forget this!!
- An alternative to the warnings module: strict
  - `use strict;`
  - `#` The strict module or pragma requires all variables to be declared



# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module
  - `use warnings;`
  - `#` The warnings module is helpful in pointing out errors – use it!!
  - `#` Notice the semi-colon at the end of the line
  - `#` nearly all lines of code need `;` as a terminator – don't forget this!!
- An alternative to the warnings module: strict
  - `use strict;`
  - `#` The strict module or pragma requires all variables to be declared
  - `#` before they are used. We'll learn more about variables next time,

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module
  - `use warnings;`
  - `#` The warnings module is helpful in pointing out errors – use it!!
  - `#` Notice the semi-colon at the end of the line
  - `#` nearly all lines of code need `;` as a terminator – don't forget this!!
- An alternative to the warnings module: strict
  - `use strict;`
  - `#` The strict module or pragma requires all variables to be declared
  - `#` before they are used. We'll learn more about variables next time,
  - `#` but they are simply named storage spaces in memory.

# Anatomy of a Perl script, part I

- Reading: "Beginning Perl, Ch 1, pp 24-32
- First line of a script: path to the Perl interpreter
  - `#!/usr/bin/perl`
  - `#` This is called the she-bang line, to remind you of sharp `#`, bang !
  - `#` Other lines beginning with `#` are comments (script documentation)
- Second line of a script: the warnings module
  - `use warnings;`
  - `#` The warnings module is helpful in pointing out errors – use it!!
  - `#` Notice the semi-colon at the end of the line
  - `#` nearly all lines of code need `;` as a terminator – don't forget this!!
- An alternative to the warnings module: strict
  - `use strict;`
  - `#` The strict module or pragma requires all variables to be declared
  - `#` before they are used. We'll learn more about variables next time,
  - `#` but they are simply named storage spaces in memory.
  - `#` Include the strict pragma if you want to write the cleanest code

# Anatomy of a Perl script, part 2

# Anatomy of a Perl script, part 2

- The Perl print function and the \n newline character:

# Anatomy of a Perl script, part 2

- The Perl print function and the \n newline character:

```
print "Hello, world.\n";
```

# Anatomy of a Perl script, part 2

- The Perl print function and the \n newline character:

```
print "Hello, world.\n";
```

# To print a newline character, it must be enclosed in double quotes

# Anatomy of a Perl script, part 2

- The Perl print function and the \n newline character:

```
print "Hello, world.\n";
```

# To print a newline character, it must be enclosed in double quotes

- The Perl print function and arithmetic:



# Anatomy of a Perl script, part 2

- The Perl print function and the \n newline character:

```
print "Hello, world.\n";
```

# To print a newline character, it must be enclosed in double quotes

- The Perl print function and arithmetic:

```
print "Six times nine is ", 6 * 9, "\n";
```

# Anatomy of a Perl script, part 2

- The Perl print function and the \n newline character:

```
print "Hello, world.\n";
```

# To print a newline character, it must be enclosed in double quotes

- The Perl print function and arithmetic:

```
print "Six times nine is ", 6 * 9, "\n";
```

# Notice that the calculation is NOT in double quotes

# Anatomy of a Perl script, part 2

- The Perl print function and the `\n` newline character:

```
print "Hello, world.\n";
```

# To print a newline character, it must be enclosed in double quotes

- The Perl print function and arithmetic:

```
print "Six times nine is ", 6 * 9, "\n";
```

# Notice that the calculation is NOT in double quotes

- Example:

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

```
use warnings;
```

```
print "Hello there!! \n";
```

```
print "Three plus nine is ", 3 + 9, "\n";
```

```
print " and 3 + 9 * 4 is ", 3 + 9 * 4, "\n";
```

```
print " Good-bye... \n";
```

# Keywords and statement blocks in Perl

# Keywords and statement blocks in Perl

- Perl keywords are reserved for functions and flow control

# Keywords and statement blocks in Perl

- Perl keywords are reserved for functions and flow control
- Examples of functions:
  - `print "Hello, world.\n";`
  - `print "Absolute value of -3 ** 3 is: ", abs(-3 ** 3), "\n";`

# Keywords and statement blocks in Perl

- Perl keywords are reserved for functions and flow control
- Examples of functions:
  - `print "Hello, world.\n";`
  - `print "Absolute value of -3 ** 3 is: ", abs(-3 ** 3), "\n";`
- Examples of flow control keywords:
  - `if (condition is true) ...`
  - `while (condition is true) ...`

# Keywords and statement blocks in Perl

- Perl keywords are reserved for functions and flow control
- Examples of functions:
  - `print "Hello, world.\n";`
  - `print "Absolute value of -3 ** 3 is: ", abs(-3 ** 3), "\n";`
- Examples of flow control keywords:
  - `if (condition is true) ...`
  - `while (condition is true) ...`
- We've seen the use of semi-colon as a statement terminator. One case where it is NOT needed is at the end of a statement block. Statement blocks are enclosed in curly braces { }

- Example:

```
if (6 * 9 > 7 ** 2) {  
    print "Six times nine is more than seven squared! \n";  
    print "In other words: ", 6 * 9, " > ", 7 ** 2, "\n"  
}
```



# Homework

# Homework

- Read "Beginning Perl", Chapter 2

# Homework

- Read "Beginning Perl", Chapter 2
- You may SKIP or SKIM the sections:
  - Binary, Hexadecimal, and Octal Numbers
  - Alternative Delimiters
  - Here-Documents
  - Converting Between Numbers and Strings
  - Bitwise Operators
  - Operators to be Seen Later

# Homework

- Read "Beginning Perl", Chapter 2
- You may SKIP or SKIM the sections:
  - Binary, Hexadecimal, and Octal Numbers
  - Alternative Delimiters
  - Here-Documents
  - Converting Between Numbers and Strings
  - Bitwise Operators
  - Operators to be Seen Later
- Homework 2 includes a small perl assignment