DO YOUR SCRIPTS EVER FEEL LIKE…

DO YOUR SCRIPTS EVER FEEL LIKE…

http://www.abnoteblog.com/wp-content/uploads/2011/10/reinventing-the-wheel.jpg

TODAY'S TOPICS:

- Perl Modules, Objects, Methods
- CPAN
- List::Util
- Statistics::Descriptive
- BioPerl Preview

WHAT ARE PERL MODULES ??

- A **module** is like a **Plug-in** that can be called from your program to carry out specific tasks.

WHAT ARE PERL MODULES ??

- A **module** is like a **Plug-in** that can be called from your program to carry out specific tasks.

Where to find modules ?

- **On LOGIN**: /uaopt/perl/5.14.2/lib/site_perl/5.14.2/
- **Online**: Comprehensive Perl Archive Network (CPAN)

WHAT ARE PERL MODULES ??

- A **module** is like a **Plug-in** that can be called from your program to carry out specific tasks.

Where to find modules ?

- **On LOGIN**: /uaopt/perl/5.14.2/lib/site_perl/5.14.2/
- **Online**: Comprehensive Perl Archive Network (CPAN)

Where to find module documentation?

- http://search.cpan.org/
- http://perldoc.perl.org/index.html

WHAT ARE PERL MODULES ??

- A **module** is like a **Plug-in** that can be called from your program to carry out specific tasks.

Where to find modules ?

- **On LOGIN**: /uaopt/perl/5.14.2/lib/site_perl/5.14.2/
- **Online**: Comprehensive Perl Archive Network (CPAN)

Where to find module documentation?

- http://search.cpan.org/
- http://perldoc.perl.org/index.html
- Try searching the web!

PERL ON THE WEB

An old coder's wit (not a flash of it!)
Rejected my Perl (the whole stash of it).
    He'd been coding all day
    An associative array,
So I said, "Well, you've sure made a hash of it!"

http://www.perlmonks.org/?node=Perl%20Poetry

OBJECT ORIENTED PROGRAMMING

- **Most Perl modules use OOP.**

OBJECT ORIENTED PROGRAMMING

- **Most Perl modules use OOP.**

- **Objects are representations of physical or conceptual entities such as a Sequence, an Alignment, or a BLAST result.**

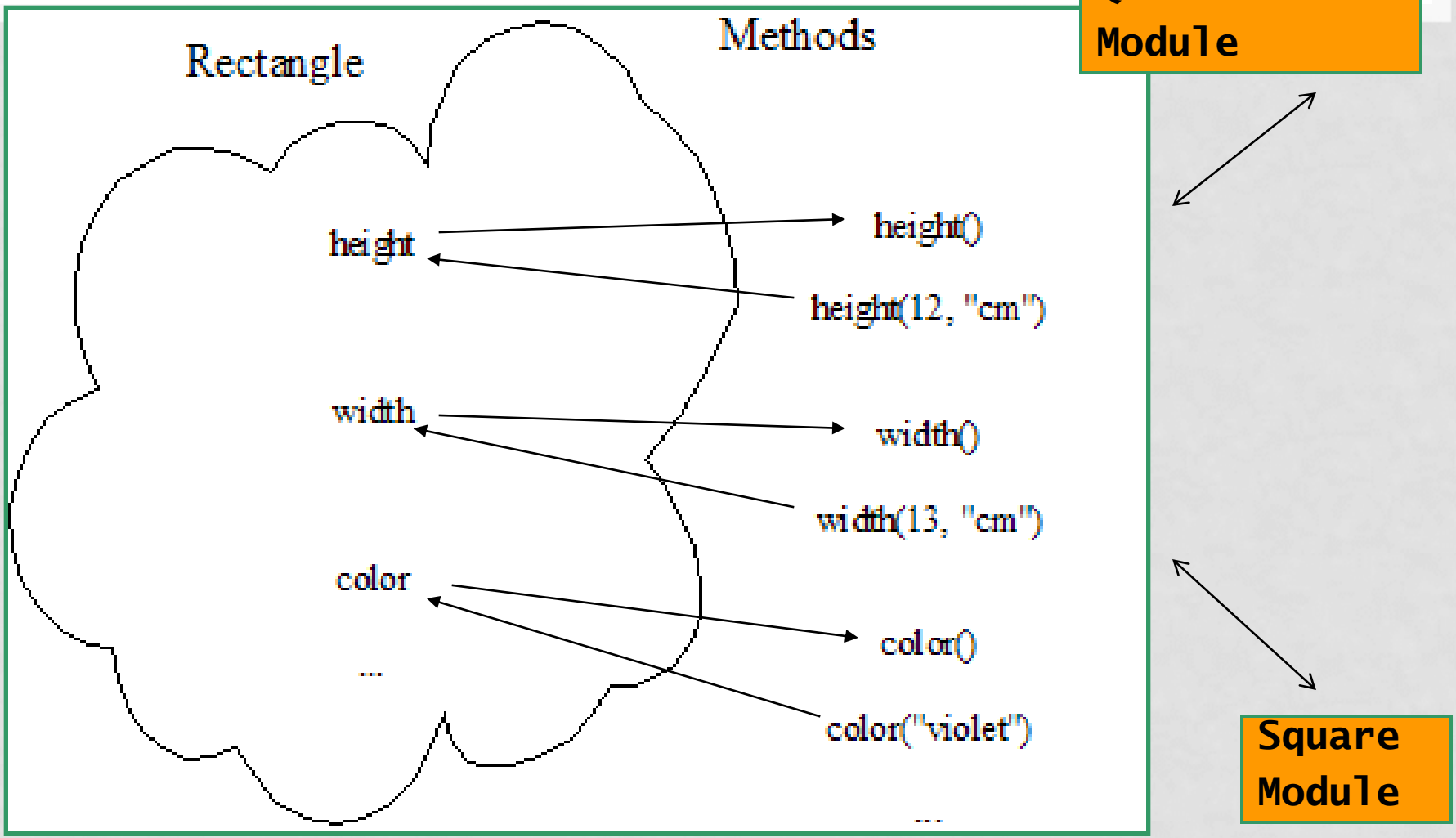
OBJECT ORIENTED PROGRAMMING

- Most Perl modules use OOP.
- Objects are representations of physical or conceptual entities such as a Sequence, an Alignment, or a BLAST result.
- Methods are functions that allow you to access and/or modify the contents of an object. Every object includes its own set of methods.

OBJECT ORIENTED PROGRAMMING

- Most Perl modules use OOP.
- Objects are representations of physical or conceptual entities such as a Sequence, an Alignment, or a BLAST result.
- Methods are functions that allow you to access and/or modify the contents of an object.
- Every object includes its own set of methods. Without methods you cannot access information in an object!

'QUADRILATERAL::RECTANGLE' MODULE



PERL MODULE EXAMPLES

- **Hierarchies:**

  Statistics::ANOVA, Statistics::ChiSquare, Statistics::Descriptive, Bio::SeqIO, Bio::SearchIO, Bio::DB::Genbank

PERL MODULE EXAMPLES

- **Hierarchies:**

  Statistics::ANOVA, Statistics::ChiSquare, Statistics::Descriptive, Bio::SeqIO, Bio::SearchIO, Bio::DB::Genbank

- **Commonly used:**

warnings	generate warnings for possible script errors
strict	force strict variable declarations
DBI	connect to a relational database
Getopt::Long	parse command line options

PERL MODULE EXAMPLES

- **Hierarchies:**

  Statistics::ANOVA, Statistics::ChiSquare, Statistics::Descriptive, Bio::SeqIO, Bio::SearchIO, Bio::DB::Genbank

- **Commonly used:**

warnings	generate warnings for possible script errors
strict	force strict variable declarations
DBI	connect to a relational database
Getopt::Long	parse command line options

Choose only the modules that you need to work with!

HOW TO USE MODULES

Add a line that says:

use <module_name_goes_here>;

HOW TO USE MODULES

Add a line that says:

use <module_name_goes_here>;

Examples:

```
use warnings;
use strict;
use Getopt::Long;
```

HOW TO USE MODULES

Add a line that says:

use <module_name_goes_here>;

Examples:

```
use warnings;
use strict;
use Getopt::Long;
```

Unix command, documentation for (some) modules:

```
perldoc  modulename
```

MORE ABOUT OBJECTS AND METHODS

- **To call a method, you use the single arrow -> following an object:**

```
$some_obj->some_method();
```

MORE ABOUT OBJECTS AND METHODS

- To call a method, you use the single arrow -> following an object:

```
$some_obj->some_method();
```

- Each method returns a particular type of object (Seq, SeqIO, GenBank, even integers and strings!)

MORE ABOUT OBJECTS AND METHODS

- To call a method, you use the single arrow -> following an object:

```
$some_obj->some_method();
```

- Each method returns a particular type of object (Seq, SeqIO, GenBank, even integers and strings!)

- If the object returned is a number or string, you can print its value, operate on it, etc.

```
my $integer = $some_obj->return_integer();
```

CREATING OBJECTS

- You can create an object via its **new()** method *or* by calling a method that returns the type of object you need:

```
use Bio::SeqIO;

$obj1 = Bio::SeqIO->new();
$obj2 = $obj1->next_seq();
```

CREATING OBJECTS

- You can create an object via its **new()** method *or* by calling a method that returns the type of object you need:

```
use Bio::SeqIO;

$obj1 = Bio::SeqIO->new();
$obj2 = $obj1->next_seq();
```

- The **new()** method allows you to associate object attribute names and values with the double arrow =>

```
Bio::SeqIO->new( -file => "<$file", -format => $format );
```

LIST::UTIL

- Provides functions that can be applied to a list, such as max, min, sum, and shuffle.

LIST::UTIL

- Provides functions that can be applied to a list, such as max, min, sum, and shuffle.

- Check documentation for List::Util at CPAN. SYNOPSIS contains module use statement:

```
use List::Util qw(max min sum);
```

LIST::UTIL

- Provides functions that can be applied to a list, such as max, min, sum, and shuffle.

- Check documentation for List::Util at CPAN. SYNOPSIS contains module use statement:

```
use List::Util qw(max min sum);
```

- DESCRIPTION lists methods with examples:

```
@list = (2, 17, 5, 94);
$largest = max(@list);
```

STATISTICS::DESCRIPTIVE

- Computes various statistical functions on lists of numbers. Contains two objects: "Full" objects have a larger set of methods than "Sparse". From CPAN:

```
use Statistics::Descriptive;
# Create a Full object
$stat = Statistics::Descriptive::Full->new();
```

STATISTICS::DESCRIPTIVE

- **Computes various statistical functions on lists of numbers. Contains two objects: "Full" objects have a larger set of methods than "Sparse". From CPAN:**

```
use Statistics::Descriptive;
# Create a Full object
$stat = Statistics::Descriptive::Full->new();
```

- **Method examples from CPAN:**

```
@list = (2, 17, 5, 94);
$stat->add_data(@list);
$var = $stat->variance();
```

BIOPERL MODULES

- **Used on sequence data, annotations, alignments, BLAST outputs, etc.**

BIOPERL MODULES

- Used on sequence data, annotations, alignments, BLAST outputs, etc.

- Commonly used BioPerl modules:

Bio::SeqIO	read/write Sequence data (in many formats)
Bio::SeqFeature	extract Features e.g. gene, repeat, from Seq object
Bio::SearchIO	parse BLAST/FASTA/HMMER output
Bio::AlignIO	read multiple sequence alignment
Bio::DB::GenBank	set up a connection to GenBank to run queries

BIOPERL MODULES

- Used on sequence data, annotations, alignments, BLAST outputs, etc.

- Commonly used BioPerl modules:

Bio::SeqIO	read/write Sequence data (in many formats)
Bio::SeqFeature	extract Features e.g. gene, repeat, from Seq object
Bio::SearchIO	parse BLAST/FASTA/HMMER output
Bio::AlignIO	read multiple sequence alignment
Bio::DB::GenBank	set up a connection to GenBank to run queries

- To use Bioperl on LOGIN, you need to first run:

  module load perl/5.14.2

FOR NEXT CLASS

- Start reading the BioPerl HOWTO for Beginners.: http://www.bioperl.org/wiki/HOWTO:Beginners

You may SKIM or SKIP the sections:

- Installing Bioperl
- Writing a script in Unix
- BLAST

IN CLASS EXERCIZES

cp –r /gsfs1/xdisk/ssolonen/ecol553_oct30 .

1) get_opt.pl
   - Run and read the script
   - How is the Getopt::Long module used? Check CPAN.
   - Try specifying arguments of incorrect type.
   - Modify one of your own scripts to use Getopt::Long
2) list_util.pl
   - Pass an integer argument. This will generate a list from 1 to n.
   - Read the script for examples using the List::Util module
   - Check CPAN on how to use the reduce() method and calculate n! (factorial of n)
3) stat_descript.pl
   - Read the script for examples using Statistics::Descriptive
   - Find another method on CPAN and use it in the script