# Day 6: Standard Library

Suggested Reading:

http://perldoc.perl.org/index-functions.html

“Perl functions by category” + skim a few

# Turn In Homework

# Homework Review

# Writing Subroutines is Great!

# (Some) Built-In Functions

# Numeric Functions

int abs · sqrt exp log · sin cos atan2 · hex oct · srand rand

```
int(3.14159)    => 3
abs(-3)         => 3

sqrt(256)       => 16

cos(0)          => 1

hex('AF')       => 175
oct('257')      => 175

rand(10)        => 6.46860370253311
rand(10)        => 9.41780438684997
rand(n)         => [0,n)
```

# String Functions

chop chomp · lc lcfirst uc ucfirst · chr ord · tr///
length index rindex substr · sprintf · reverse

```
ord('A') => 65
chr(65)  => 'A'

index('start', 'art') => 2
rindex('start', 't')  => 4
substr('stop', 1, 2)  => 'to'
substr('stop', -3)    => 'top'

sprintf('[%5.2f]', 3.14159) => '[ 3.14]'
```

# Array & Hash Functions

push pop shift unshift · splice · split join · reverse
exists delete · keys values each

```
my @a = ( 1, 2, 3, 4, 5 );
my @b = splice(@a, 1, 3);
  # @a == ( 1, 5 )
  # @b == ( 2, 3, 4 )

splice(@a, 1, 0, 'a', 'b');
  # @a == ( 1, 'a', 'b', 5 )

join('/', @a);  => '1/a/b/5'
```

*We will revisit* `split()` *in Regular Expressions II*

# Input/Output Functions

warn die · open getc <> (readline) read close
print printf · opendir readdir closedir

```
warn "Something bad happened!\n";
die "Something really bad happened!\n";

open(INPUT, $filename) or die "... $!\n";
while (my $character = getc(INPUT)) {
    # ...
}

printf('[%5.2f]', 3.14159);
```

# File and Directory Functions

-X (-r, -w, …) stat lstat · glob · chdir mkdir rmdir
chmod chown · link symlink readlink · unlink rename

```
my ($dev, $ino, $mode, $nlink, $uid, $gid,
    $rdev, $size, $atime, $mtime, $ctime,
    $blksize, $blocks) = stat($filename);

chdir $new_directory;

if (-l $path) {
    my $real_path = readlink($path);
}
```

# 3 Amazing Functions

# Sorting

# sort()

Sort a list in *string* order

```
my @a = ( 1, 2, 3, 5, 7, 11, 13, 17 );
my @b = sort(@a);

=> ( 1, 11, 13, 17, 2, 3, 5, 7 )
```

Sort hash keys

```
foreach my $key (sort keys %hash) {
    print "$key => $hash{$key}\n";
}
```

# Custom `sort()`: The Long Way

```
sub compare_numbers {
    if ($a < $b) { return -1; }
    if ($a > $b) { return 1; }
    return 0;   # i.e., $a == $b
}

my @a = ( 10, 5, 22, 3, 1, 17 );
my @b = sort compare_numbers @a;

=> ( 1, 3, 5, 10, 17, 22 )
```

# Custom sort(): Simpler

```
sub compare_numbers {


    return $a <=> $b;
}

my @a = ( 10, 5, 22, 3, 1, 17 );
my @b = sort compare_numbers @a;

=> ( 1, 3, 5, 10, 17, 22 )
```

# Custom `sort()`: The Perl Way

```
my @a = ( 10, 5, 22, 3, 1, 17 );
my @b = sort { $a <=> $b } @a;

=> ( 1, 3, 5, 10, 17, 22 )
```

# Filtering

# Filtering: As We Know It

```
my @numbers = ( ... );

my @results;
foreach my $item (@numbers) {
    if ($item % 2) {
        push @results, $item;
    }
}
```

# Filtering: Statement Modifier

```
my @numbers = ( ... );

my @results;
foreach my $item (@numbers) {

    push @results, $item if $item % 2;

}
```

# Filtering: Automatic Variable

```
my @numbers = ( ... );

my @results;
foreach              (@numbers) {

    push @results, $_    if    $_ % 2;

}
```

# Filtering: The Perl Way

```
my @numbers = ( ... );



my @results = grep($_ % 2, @numbers);
```

# Filtering: The *Really* Perl Way

```
my @numbers = ( ... );



my @results = grep { $_ % 2 } @numbers;
```

*Any* expression or set of statements can go in {}

# Transforming

# Transforming: As We Know It

```
my @numbers = ( ... );

my @results;
foreach my $item (@numbers) {
    push @results, $item * 2;
}
```

# Transforming: The Perl Way

```
my @numbers = ( ... );



my @results = map { $_ * 2 } @numbers;
```

# Transforming: Example 1

```
my @f = <INPUT_FILE>;

my $i = 0;
my @what_am_i = map { $i++ . ": $_" } @f;

print join('', @what_am_i);
```

# Transforming: Example 2

```
my @words = <INPUT_FILE>;

my %hash = map { $_ => 1 } @words;
my @unique = keys %hash;

print join("\n", sort @unique);
```

# Using Modules

# Modules

Perl *Module* = collection (library) of pre-written code

```
use Benchmark;
use POSIX qw/strftime floor ceil/;
```

**CPAN** : **C**omprehensive **P**erl **A**rchive **N**etwork

**www.cpan.org**

**search.cpan.org**

**perldoc.perl.org** — “Modules”

# File::Basename

## Pick apart file paths

```
use File::Basename;

my $path = '/usr/share/dict/words';
my $dir  = dirname($path);
my $name = basename($path);

$dir   =>  /usr/share/dict
$name  =>  words
```

# File::Find

## Recursive directory reader (like shell find)

```
use File::Find;
find(\&find_handler, $directory);

sub find_handler {          # $_ is filename
    return if $_ eq '.' or $_ eq '..';
    if (-f $_) {
        print "In '$File::Find::dir', ";
        print "got '$File::Find::name'\n";
    }
}
```

# POSIX 'strftime'

## Convert dates to readable strings

```
use POSIX 'strftime';
strftime('%Y-%m-%d %H:%M:%S', localtime);

=> 2010-07-23 11:47:32
```

# Last 2 Slides…

# Other Scripting Languages

- All have standard libraries, functions, etc.
- Just a matter of learning what is available
- Find a good reference (book, website, …)
- ***Don't reinvent the wheel!***

# Homework

- Print interesting info about directories, recursively
- Use built-in functions and modules as appropriate
- Read the hints in the homework carefully!
- Start early…