

Week 4 Lab, SI 601
Jackie Cerretani and Cheng-Lun Li

Question 1 Code:

```
#week4 homework
use warnings;

use LWP::Simple;
use URI::URL;
use HTML::LinkExtor;
use HTML::Strip;
use HTML::Strip::Whitespace qw(html_strip_whitespace);
use HTML::Entities;
    use HTML::Entities::Numbered;
use LWP::UserAgent

# 1. Using URI::URL, prepare the URL so we can deal with relative image links later
$urltoget = "http://projects.si.umich.edu/~efrierso/welcome.htm";
my $url=URI->new($urltoget);

# 2. Using LWP::Simple, get the contents of the page
$content = get($url);

# Convert numeric entities to html entities
# a neat trick, though I haven't been able to get a proper conversion
# from numeric to english to work with the HTML::Strip, which
# calls HTML::Entities by default
# my $entity_contents = decimal2name($content);
# my $entity_contents = decode_entities($entity_contents);

# 3. Use HTML::Strip to clean off the HTML tags.

my $hs = HTML::Strip->new();
my $clean_content = $hs->parse($content);

# 4. Use HTML::Strip::Whitespace to clean off extra space

my $whiteout_content;
html_strip_whitespace (
    'source' => \$clean_content,
    'out' => \$whiteout_content
);

# 5. Print stripped and cleaned text to output file

open(OUT,"> dooce_out.txt");
print OUT "Contents of http://projects.si.umich.edu/~efrierso/welcome.htm:\n-----\n";
print OUT "$whiteout_content\n\n";

# 6. Grab links using HTML::LinkExtor

$ua = LWP::UserAgent->new;

# Set up a callback that collect image links
my @imgs = ();
sub callback {
    my($tag, %attr) = @_;
    return if $tag ne 'img'; # we only look closer at <img ...>
    push(@imgs, values %attr);
}
```


<http://projects.si.umich.edu/~efrierso/images/spacer.gif>

Question 2 Script:

(The annotation explains our goals with the code.)

```
# module to parse HTML
use HTML::TokeParser;
# module for simple web retrieval
use LWP::Simple;

#specify year of car models and the number of results to retrieve
#specify the selection of data
#In this case, it is going to get the data after 1997
#From capacity of 1500cc
#looking for "weight", acceleration time from 0~100km/h (100), and Maxspeed (kmh)
#for each capacity, try to find as many cars as it can, maximum 999 cars each page.
my $startyear = 1997;
my $lcap = 1500;
my @type = ("wh", "100", "kmh");
my @typelabel = ("Weight(kg)", "0~100(km/h)", "MaxSpeed(km/h)");
my $number_of_results = 999;
my $count = (@typelabel-1);

# retrieve a page from carfolio.com that contains cars listed
# from heaviest to lightest

#print the header
print "Class\tType\t1997\t1998\t1999\t2000\t2001\t2002\t2003\t2004\t2005\t2006\t2007\n";

#start of the loop, this nested loop is going to print the results in the format that many eyes
like
#For each class of engine capacity, Perl is going to look for the average value of each type, from
year 1997 to year 2007
while($lcap <6000){
    #set the upper limit of class
    $hcap = ($lcap+499);
    $i = 0;
    do {
        #reset the year
        $year = $startyear;
        #print class and type (type changes each loop)
        print "$lcap\t$typelabel[$i]\t";
        #print the value of certain type, from 1997 to 2007
        while ($year < 2008){
            $value = &getvalue;
            print "$value\t";
            $year++;
        }
        $i++;
        #go the the next line when type changes
        print "\n";
    } until ($i > $count);
    #go on to next class
    $lcap = ($hcap+1);
}

sub getvalue {
    $totalvalue = 0;
    my $page = get
    "http://www.carfolio.com/search/specific/?searchon=$type[$i]&asp=&y1=$year&y2=$year&num=$number_of_
    results&lcap=$lcap&hcap=$hcap";

    # create a new instance of a parser
    my $p = HTML::TokeParser->new(\$page)
        || die "parse error parsing $page\n";

    # the information will be after the 3rd table tag
    $p->get_tag('table') || die "Not enough table tags!" foreach (1..3);
```

```

# the first row ('tr') contains header info, so we'll skip it
$p->get_tag('tr');

# we expect our info to be in the second row
$p->get_tag('tr');

my $singlevalue;
my $model;
my $scarcount = 0;
# output column labels
#print "weight (kg)\tmodel\n";
#print "weight (kg)\n";
#print "-----\n";
do {
    # the first column contains the search result #. we're skipping it
    $p->get_tag('td');

    # the second column contains the weight in kilos
    $p->get_tag('td');

    # we need to explicitly get the text associated with that tag
    $singlevalue = $p->get_trimmed_text, "\n";

    # and also strip some newlines because in the file what we have is
    # <td>
    # 2267
    # </td>
    $singlevalue =~ s/[\^d\.]//g;

    # ok the next column is going to contain the model
    $p->get_tag('td');

    # but, actually, it will be the anchor text of a link within that column
    $p->get_tag('a');

    # tried grabbing the url here, but it didn't work
    # my($tag, $attr, $attrseq, $rawtxt) = @{$p};
    # my $a_href = $attr->{'href'};

    #print "$a_href\n";

    # so we'll get the anchor text
    $model = $p->get_text, "\n";
    #print "$model\n";

    # if we were successful in the above, and the weight is a number
    # we'll output the weight and model
    if ($singlevalue =~ /^[^d\.]+$/) {
        #print "$singlevalue\t$model\n";
        $totalvalue += $singlevalue;
    }
    $scarcount++;
    #print "$scarcount\n";
} until ($singlevalue !~ /^[^d\.]+$/);
#when divide by 0, return 0 instead
if(($scarcount-1) == 0){return 0;}
else{
    #If not divided by 0, get average value and return it
    $averagevalue = ($totalvalue/($scarcount-1));
    return $averagevalue;
}
#print "$averagevalue\n";
}

# we stop when we did not get a numerical weight
# meaning we got to the end of the search results

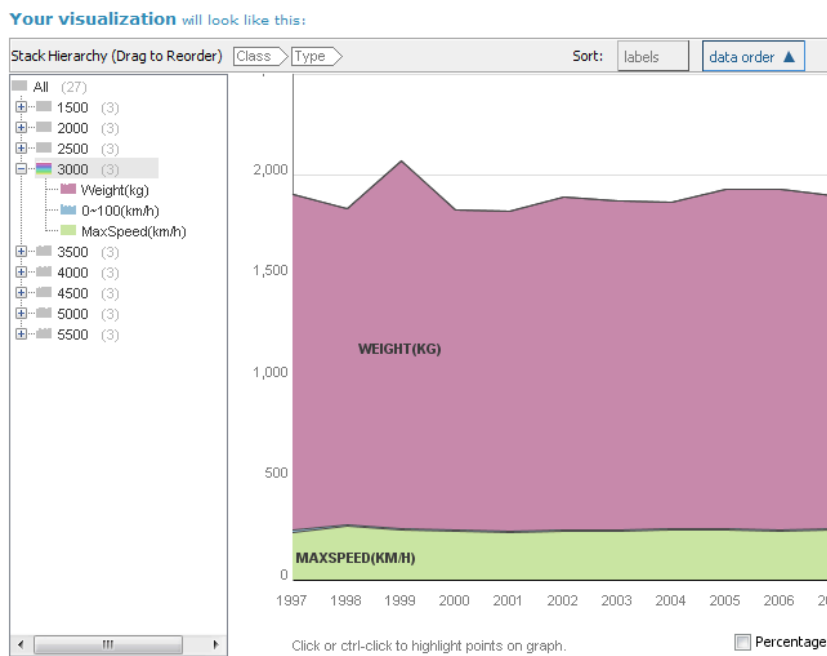
```

Output for question 2: [word wrapped]

The format of the table for many eyes is:

Class	Type	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007
1500	Weight(kg)	1227.33333333333	1274.20720720721	1241.52845528455	1271.03703703704	1278.384	1345.99333333333	1328.98507462687	1348.65322580645	1407.82448979592	1388.48042704626	1400.77777777778
1500	0~100(km/h)	10.9928571428571	10.6354838709677	10.81625	10.6941176470588	10.8311764705882	9.95537190082644	11.1010582010582	10.2452830188679	9.79182692307693	10.2769547325103	10.0015957446808
1500	MaxSpeed(km/h)	195.216666666667	198.458333333333	197.823008849558	203.544642857143	195.898936170213	205	194.917431192661	205.22869955157	208.485981308411	200.736220472441	205.210526315789
2000	Weight(kg)	1418.11111111111	1453.65384615385	1523.27586206897	1469.90909090909	1490.91463414634	1511.35106382979	1525.49514563107	1476.21153846154	1522.96551724138	1503.44047619048	1592.5
2000	0~100(km/h)	9.112	9.26	11.1521739130435	9.57230769230769	9.93333333333333	9.36923076923077	9.81951219512195	9.88235294117647	8.86818181818182	8.88448275862069	9.2375

Visualization for question 2:



We used a stack graph for categories to show our findings for each Capacity Class of cars per year. You can see the visualization on Many Eyes at http://services.alphaworks.ibm.com/manyeyes/view/Sm4H4JsOtha6IW_AC_xHJ2- or under the title, “Technology improvement of car industry [Jack&Jackie]” in the DRAT topic hub.