LING/C SC/PSYC 438/538

Lecture 8

Sandiway Fong

Today's Topics

- Homework 6 Review
- File I/O in Perl and Python

Homework 6 Review

- **Basic Idea**:
  1. this is a test
  2. t h i s i s a t e s t
  3. t h s s t s t
  4. ths s tst
- **Perl:**
  - @ARGV
  - for $word (@ARGV) { @c = split //, $word; … }
  - for $c (@c) { if $c is a vowel, don't print it, else print $c}

Homework 6 Review

```
1 my @vowels = qw(a e i o u A E I O U);
2 my  %vowel;
3 foreach $v (@vowels) {
4   $vowel{$v} = undef
5 }
6 foreach $word (@ARGV) {
7
8
9
10
11
12
13 }
14 print "\n"
```

```
1 my @vowels = qw(a e i o u A E I O U);
2 my  %vowel;
3 foreach $v (@vowels) {
4   $vowel{$v} = undef
5 }
6 foreach $word (@ARGV) {
7   foreach $char (split //, $word) {
8     unless (exists $vowel{$char}) {
9       print $char
10    }
11  }
12  print " "
13 }
14 print "\n"
```

Homework 6 Review

- Example:

```
perl hw6.perl suddenly a White Rabbit with pink eyes
ran close by her.
sddnly  Wht Rbbt wth pnk ys rn cls by hr.
```

Homework 6 Review

```
$ perl -le 'tr/AEIOUaeiou//d for (@ARGV); print "@ARGV"' suddenly a White Rabbit
 with pink eyes ran close by her.
sddnly  Wht Rbbt wth pnk ys rn cls by hr.
$
```

- *transliterate* `for (@ARGV);`
  - for-loop on @ARGV with reversed syntax; usually `for (@ARGV) {…}`
- `tr/AEIOUaeiou//d`
  - modifies the words stored in @ARGV
- `print "@ARGV"`
  - puts spaces between each word
- `perl -l`
  - prints a newline after each print statement

Homework 6

- Part 2:
  - Modify your program for part 1 to not delete leading vowels
  - Example:
    - `If a sentence is unreadable`
    - `f sntnc s nrdbl`
    - `If a sntnc is unrdbl`
- **An idea:**
  - inside the for-loop examining each character at a time
  - use a variable to flag whether the character is the first character of a word

Homework 6 Review

flag

```
1 my @vowels = qw(A E I O U a e i o u);
2 my %vowel;
3 foreach $v (@vowels) {
4   $vowel{$v} = undef
5 }
6 foreach $word (@ARGV) {
7   $not1stchar = 0;
8   foreach $char (split //, $word) {
9     unless ($not1stchar && exists($vowel{$char})) {
10      print $char
11    }
12    $not1stchar = 1
13  }
14  print ' '
15 }
16 print "\n"
```

Homework 6 Review

```
ling538-21$ perl hw6.perl suddenly a White Rabbit with pink eyes ran close by her.
sddnly  Wht Rbbt wth pnk ys rn cls by hr.
ling538-21$ perl hw6c.perl suddenly a White Rabbit with pink eyes ran close by her.
sddnly a Wht Rbbt wth pnk eys rn cls by hr.
ling538-21$
```

- Option: -s (switch) –`del1stvowel` (if set, doesn't preserve the first letter of a word if it's a vowel)

```
ling538-21$ perl -s hw6d.perl -del1stvowel suddenly a White Rabbit with pink eyes ran close by her.
sddnly  Wht Rbbt wth pnk ys rn cls by hr.
ling538-21$ perl -s hw6d.perl suddenly a White Rabbit with pink eyes ran close by her.
sddnly a Wht Rbbt wth pnk eys rn cls by hr.
ling538-21$
```

The -s option lets you create your own custom switches. Custom switches are placed after the script name but before any filename arguments. Any custom switches are removed from the @ARGV array. Then a scalar variable is named after the switch is created and initialized to 1.

Homework 6 Review

- Readability:
  - `sddnly  Wht Rbbt wth pnk ys rn cls by hr.`
  - `sddnly a Wht Rbbt wth pnk eys rn cls by hr.`

- Example:
  - `perl hw6c.perl suddenly a White Rabbit with pink eyes ran close by her.`
  - `sddnly a Wht Rbbt wth pnk eys rn cls by hr.`

Homework 6 Review

- Making it switchable
  - Option: `-s` (switch) `-del1stvowel`
    - (*if set, doesn't preserve the first letter of a word if it's a vowel*)
    - The -s option lets you create your own custom switches. Custom switches are placed after the script name but before any filename arguments. Any custom switches are removed from the @ARGV array. Then a scalar variable is named after the switch is created and initialized to 1.
- Example:

```
$ perl hw6d.perl suddenly a White Rabbit with pink eyes ran
close by her.
sddnly a Wht Rbbt wth pnk eys rn cls by hr.
$ perl -s hw6d.perl -del1stvowel suddenly a White Rabbit
with pink eyes ran close by her.
sddnly  Wht Rbbt wth pnk ys rn cls by hr.
$
```

Homework 6 Review

option

```
1 my @vowels = qw(a e i o u A E I O U);
2 my %vowel;
3 foreach $v (@vowels) {
4   $vowel{$v} = undef
5 }
6 foreach $word (@ARGV) {
7   $notfirst = $del1stvowel;
8   foreach $char (split //, $word) {
9     unless ($notfirst && exists $vowel{$char} ) {
10      print $char
11    }
12    $notfirst = 1
13  }
14  print ' '
15 }
16 print "\n"
```

Perl: file I/O

- Step 1: call open()

Files and I/O

You can open a file for input or output using the `open()` function. It's documented in extravagant detail in perlfunc and perlopentut, but in short:

```
1.  open(my $in,  "<",  "input.txt")  or die "Can't open input.txt: $!";
2.  open(my $out, ">",  "output.txt") or die "Can't open output.txt: $!";
3.  open(my $log, ">>", "my.log")     or die "Can't open my.log: $!";
```

Files: must be opened for reading “<“ or writing “>” (overwrite or append mode “>>”)
Shell syntax: **I/O *redirection*** “<“ “>”
Opening a file creates a **file handle** (Perl variable)
– not to be confused with filename
Supply the **file handle** for read/write

Perl: file I/O

- Step 2: use the <> operator:

You can read from an open filehandle using the `<>` operator. In scalar context it reads a single line from the filehandle, and in list context it reads the whole file in, assigning each line to an element of the list:

```
1.  my $line  = <$in>;
2.  my @lines = <$in>;
```

Reading in the whole file at one time is called slurping. It can be useful but it may be a memory hog. Most text file processing can be done a line at a time with Perl's looping constructs.

$in is the file handle instantiated by the open() call

Perl: file I/O

- Line by line:

The <> operator is most often seen in a while loop:

```
1.  while (<$in>) {     # assigns each line in turn to $_
2.      print "Just read in this line: $_";
3.  }
```

```
open($txtfile, $ARGV[0]) or die "File $ARGV[0] not found!\n";
while ($line = <$txtfile>) {
    print "$line";
}
close($txtfile)
```

Notes:

1. the command **$line = <$txtfile>** inside the condition reads in a line from the file referenced by the *file handle* **$txtfile**
2. and places that line into the variable **$line** (*including the newline at the end of the line*)
3. At the end of the file, **$line** is just an empty string (equivalent to false).
4. the filename is the first argument to the Perl program (arguments go in **@ARGV**).

Perl: file I/O

falconheavylaunch.txt

```
Elon Musk's Falcon Heavy rocket launches successfully
By Jonathan Amos BBC Science Correspondent

US entrepreneur Elon Musk has launched his new rocket, the Falcon Heavy, from the Kennedy
Space Center in Florida.

The mammoth vehicle - the most powerful since the shuttle system - lifted clear of its pad
without incident to soar high over the Atlantic Ocean.

It was billed as a risky test flight in advance of the lift-off.

The SpaceX CEO said the challenges of developing the new rocket meant the chances of a
successful first outing might be only 50-50.

"I had this image of just a giant explosion on the pad, a wheel bouncing down the road.
But fortunately that's not what happened," he told reporters after the event.

With this debut, the Falcon Heavy becomes the most capable launch vehicle available.

It is designed to deliver a maximum payload to low-Earth orbit of 64 tonnes - the
equivalent of putting five London double-decker buses in space.

Such performance is slightly more than double that of the world's next most powerful
rocket, the Delta IV Heavy - but at one third of the cost, says Mr Musk.

For this experimental and uncertain mission, however, he decided on a much smaller and
whimsical payload - his old cherry-red Tesla sports car.

A space-suited mannequin was strapped in the driver's seat, and the radio set to play a
David Bowie soundtrack on a loop.
```

Perl: file I/O

- What does this code do?
  - `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f>) {print((split " ")[0],"\n")}'`
  - `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f>) {print((split " ")[0],"\n")}'  | wc -l`
  - reports 49 lines

Perl: file I/O

- A bit more:
  - `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f>) {@words = split " "; $sum+=@words}; print $sum'`

- Compare with:

```
wc falconheavylaunch.txt
      49     669    3973 falconheavylaunch.txt
```

Perl: file I/O

And another bit more.

- What does this code do?
  - `perl –e 'open $f, "falconheavylaunch.txt"; while (<$f>) {for (split " ") {$freq{$_}++}};'`

Perl: file I/O

- Let's print out the table sorted in descending order
  - `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f>) {for (split " ") {$freq{$_}++}}; for (sort {$freq{$b} <=> $freq{$a}} keys %freq) {printf "%-20s %s\n", $_, $freq{$_}}' | head -10`

```
1. the                  49
2. of                   19
3. to                   18
4. a                    14
5. and                  14
6. -                    11
7. on                   9
8. as                   8
9. was                  8
10.is                   7
```

formatted printing

```
printf "%-20s %s\n", $_, $freq{$_}
```

Perl's `sprintf` permits the following universally-known conversions:

```
%%    a percent sign
%c    a character with the given number
%s    a string
%d    a signed integer, in decimal
%u    an unsigned integer, in decimal
%o    an unsigned integer, in octal
%x    an unsigned integer, in hexadecimal
%e    a floating-point number, in scientific notation
%f    a floating-point number, in fixed decimal notation
%g    a floating-point number, in %e or %f notation
```

flags

one or more of:

```
space   prefix non-negative number with a space
+       prefix non-negative number with a plus sign
-       left-justify within the field
0       use zeros, not spaces, to right-justify
#       ensure the leading "0" for any octal,
        prefix non-zero hexadecimal with "0x" or "0X",
        prefix non-zero binary with "0b" or "0B"
```

https://perldoc.perl.org/functions/sprintf

Perl: file I/O

- Strictly speaking, we'd like to remove prefix/suffix punctuation, i.e. avoid accumulating "words" like:
  - `"It'll 2`
  - `…`
  - `sea. 1`
  - `literally." 1`
- We haven't covered regex yet:
  - `for $w (split " ") {$w=~s/(^\W+)|(\W+$)//g; $freq{$w}++}`

Python: Files

- Like all other programming languages, uses a file handle, called **file variable**: open()
- `infile = open("file.txt","r")` `outfile = open("results.txt","w")`

```
>>> with open('workfile') as f:
...     read_data = f.read()
```

`<filevar>.read()` Returns the entire remaining contents of the file as a single (potentially large, multi-line) string.

`<filevar>.readline()` Returns the next line of the file. That is all text up to *and including* the next newline character.

`<filevar>.readlines()` Returns a list of the remaining lines in the file. Each list item is a single line including the newline character at the end.

```
infile = open(someFile, 'r')
for line in infile.readlines():
    # process the line here
infile.close()
```

same as

```
infile = open(someFile, 'r')
for line in infile:
    # process the line here
infile.close()
```

Python: Files

- https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files

For reading lines from a file, you can loop over the file object. This is memory efficient, fast, and leads to simple code:

```
>>> for line in f:
...     print(line, end='')
...
This is the first line of the file.
Second line of the file
```

If you want to read all the lines of a file in a list you can also use `list(f)` or `f.readlines()`.

Python: Files

```
>>> i = open("falconheavylaunch.txt",'r')
>>> type(i)
<class '_io.TextIOWrapper'>
>>> text = i.read()
>>> len(text)
3962
>>> text[:100]
"Elon Musk's Falcon Heavy rocket launches successfully\nBy Jonathan Amos BBC Sci
ence Correspondent\n\nUS"
>>> sentences = i.readlines()
>>> len(sentences)
0
>>> i.close()
```

Python: Files

```
>>> i = open("falconheavylaunch.txt",'r')
>>> sentences = i.readlines()
>>> len(sentences)
49
>>> sentences[0]
"Elon Musk's Falcon Heavy rocket launches successfully\n"
>>> sentences[1]
'By Jonathan Amos BBC Science Correspondent\n'
>>> sentences[2]
'\n'
>>> sentences[3]
'US entrepreneur Elon Musk has launched his new rocket, the Falcon Heavy, from t
he Kennedy Space Center in Florida.\n'
>>>
```