

← *Shameful advertisement*

Foundations in Perl Programming – for experienced programmers

**Week One**

## This weeks topics

---

- **About the Course**
- **What is Perl**
- **Getting Setup**
- **Fundamentals of the Perl language**
- **Assignments**

## About the Course

---

Review description, discuss goals and objectives

What you need to do to complete the class?

Pass out class list

Study groups

Need Help during the week?

## About the Course

---

### Final Project

- Required in order to complete the class
- It is up to you to come up with the idea.
- You must submit your idea by the start of the third class. By then you will have seen enough Perl to get an idea what is reasonable for you to do.

# What is Perl?

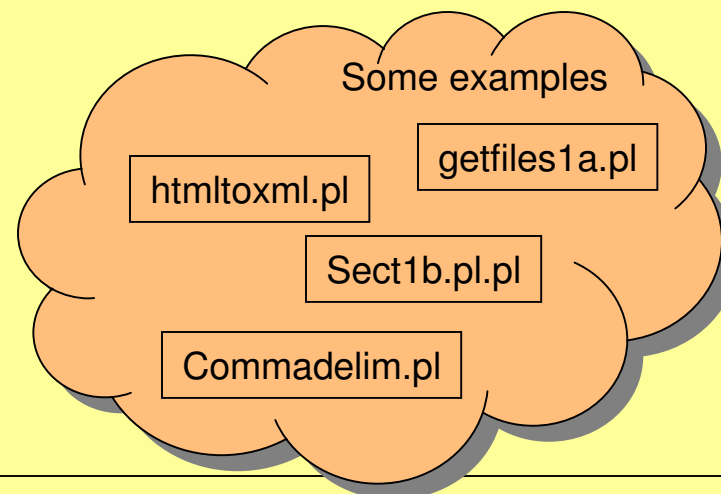
---

## Perl (Practical Extraction Reporting Language)

general purpose language that can be used for a variety of applications including...

---

- Developing programs and tools for maintaining and administering operating systems
- Creating dynamic HTML pages
- Creating server side WEB scripts
- Writing complex parsers that read, interpret, and modify text files
- And many more applications.



Manual: p9-10

## What is Perl?

---

“There’s more than one way to do it”

*“How does Perl put the focus onto the creativity of the programmer? Very simple. Perl is humble. It doesn't try to tell the programmer how to program. It lets the programmer decide what rules today, and what sucks. It doesn't have any theoretical axes to grind. And where it has theoretical axes, it doesn't grind them. Perl doesn't have any agenda at all, other than to be maximally useful to the maximal number of people.”*

*“True greatness is measured by how much freedom you give to others, not by how much you can coerce others to do what you want.”*

Manual: p10-11

## Getting Setup

---

Who has done it so far? Questions? Problems encountered?

Quick tour on how

# Fundamentals of the language

---

## Basics

- Hello World
- Print
- Statements
- Brackets
- STDIN
- Running a Perl program
- Comment codes

# Fundamentals of the language

---

## Scalars

- In Perl you do not need to worry about data types. Scalars can hold numbers or characters.
- Scalars can be scoped local or global
- Scalar names begin with a \$ when declaring or referring to in code
- Scalars can hold large text strings, words, characters, numbers
- Scalars cannot begin with an underscore or dash
- Legal characters are a-z,A-Z,0-9,-,\_,
- Scalar names can be as long as you want (but please do not write a novel)
- Scalar name must contain a character
- The second letter of a scalar must be a character (strange)
- Scalars can be declared 'on the fly' unless you use USE STRICT (which I recommend you do)

Note - Perl considers the following to be variables:

Scalars  
Arrays  
Hashes  
Subroutines  
Typeglobs

Manual: p15-16

# Fundamentals of the language

---

## Scalars

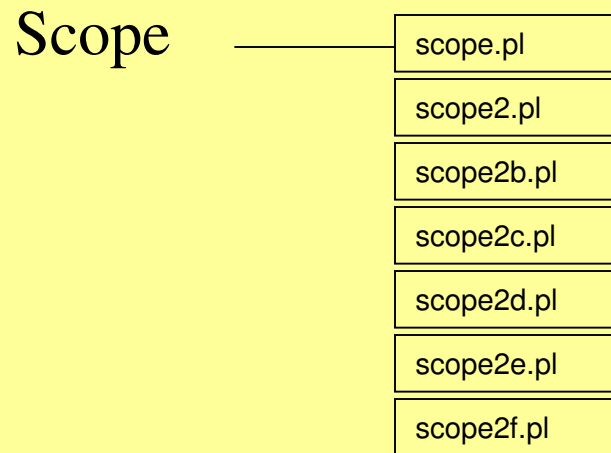
```
scalar.pl  
Scalar_1.pl  
Scalar_2.pl
```

- My
- Use Strict
- Numbers
- Strings

# Fundamentals of the language

---

## Scalars



Manual: p36-40

# Fundamentals of the language

---

## Operators

- Arithmetic `miscoper.pl`
- Assignment `oper-assign.pl`  
`assignmentequals.pl`
- Logical `and-or.pl`
- Comparison
  - String `oper-stringcompare.pl`
  - Numerical `oper-integercompare.pl`
- Matching
- Backslash `case.pl`

### Some things to know and some gothcha's

- = and eq
- assignment and equals
- concatenation
- some terse operators (+=)

Manual: p17-18

# Fundamentals of the language

---

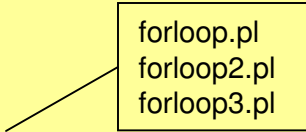

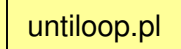
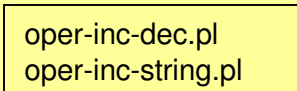
## Logic

- IF `if.pl`
- IF/ELSE `if-else.pl`
- IF/ELSIF `if-elsif.pl`
- Nesting IF `if-nested.pl`

# Fundamentals of the language

---

## Loops

- For 
  - forloop.pl
  - forloop2.pl
  - forloop3.pl
- While 
  - whileloop.pl
  - whileloop2.pl
- Until 
  - untilloop.pl
- Iteration operator 
  - oper-inc-dec.pl
  - oper-inc-string.pl

## In class exercise

---

Write a palindrome checker (a palindrome is a word that is spelled the same both forwards and backwards, like 'toot')

- Use “use strict”
- Test your code so that it works for both palindromes and non-palindromes

```
# to reverse a scalar
my $str = "food";
$str = reverse($str);
print $str; #yields doof
```

Approx  
20 mins

# Fundamentals of the language

## File IO

- Reading
- Error handling `filio-die.pl`
- Copying `filio-io1a.pl`
- Appending `filio-io1b.pl`
- Other `filio-read.pl`

`filio-open.pl`  
`filio-open2.pl`  
`filio-open2a.pl`  
`filio-open2b.pl`  
`filio-open2c1.pl`  
`filio-open2c2.pl`

There is more than one way to do it.

# Fundamentals of the language

## Arrays

```
@x;          #an empty array  
@y = ();    #also an empty array  
@z = (a,b,c,1,2,"3",1+3,"1"+"3");
```

array.pl

- How can I determine the length of an array?
- Can I assign an array to an array?
- Can I dump the contents of a file into an array?
- How do I access an element in an array?
- When I print an array, how can I put a space or other separator between each element?
- How can I sort an array?

array1.pl

arraysort.pl

Manual: p26-28

## In class exercise

---

```
# Week 1
# Exercise 1a
# File: exe1_1a.pl

# finish this script by looping through the array and
# returning California to the terminal

use strict;

my @arr=qw(Texas Idaho Washington Arizona Illinois
           Kansas California Louisiana Delaware);

# Note qw saves us from having to use quotes around array
# elements, and commas to delimit elements.
```

Approx  
20 mins

## Pattern Matching

- Matching operators
- Character patterns matching `patt1.pl`
- Character classes `brackets.pl`
- Matching special characters
- Split `split.pl`

## Fundamentals of the language

---

### Pattern Matching – Matching Operators

```
$str =~ /a/;
```

```
$str !~ /a/;
```

## Fundamentals of the language

---

### Pattern Matching ...+ and \*

patt1.pl

- + matches one or more preceding characters.
- \* matches zero or more preceding characters.

```
my @arr = ("me", "meet", "meat", "meaning", "melvin");  
my $i;  
print ("Match - mea: with the asterisk\n");
```

```
for ($i=0;$i<5;$i++) {  
    if (@arr[$i] =~ /mea*/) {  
        print ("@arr[$i]\n");  
    }  
}
```

#or

```
for ($i=0;$i<5;$i++) {  
    if (@arr[$i] =~ /mea+/) {  
        print ("@arr[$i]\n");  
    }  
}
```

## Fundamentals of the language

---

### Pattern Matching ...+ and \*

```
@arr = (b, bb, bbb, bba, bab);  
  
for ($i=0;$i<5;$i++) {  
    if (@arr[$i] =~ /bb*/ ) {  
        print ("@arr[$i]\n");  
    }  
}
```

Will Match? \_\_\_\_\_

```
@arr = (b, bb, bbb, bba, bab);  
  
for ($i=0;$i<5;$i++) {  
    if (@arr[$i] =~ /bb+/) {  
        print ("@arr[$i]\n");  
    }  
}
```

Will Match? \_\_\_\_\_

## Fundamentals of the language

### Pattern Matching ... [ ]

brackets.pl

```
$str = "abcdefghij";  
  
if ($str =~ /[fik]/) {  
    print ("$str\n");}
```

**[fik]** Reads? \_\_\_\_\_

## Fundamentals of the language

---

### Pattern Matching – Misc. Stuff

#### Match any character

```
/[a-zA-Z]/
```

or

```
/[a-z]/i
```

#### Match any number

```
/[0-9]/
```

#### Match a special character

```
\<special_character>
```

```
/\?/
```

```
/\.\?abc/
```

## Fundamentals of the language

---

### Pattern Matching - Split

split.pl

```
$str = "this is the way to Ed and Fred and Teds  
place";  
@arr = split (/ /, $str);  
$cnt = @arr;  
print "$cnt\n";
```

```
$str1 = "this is the one? This is two! This is  
three.";  
@arr1 = split ([\.\!\?]/, $str1);  
$cnt1 = @arr1;  
print "@arr1\n";  
print "$cnt1\n";
```

# Fundamentals of the language

---

Subroutines `sub1.pl`

`Manual: p31-32`

# Fundamentals of the language

---

Chop/Chomp

chop-chomp.pl  
pattern2b.pl

Manual: p33-34

## Assignments

---

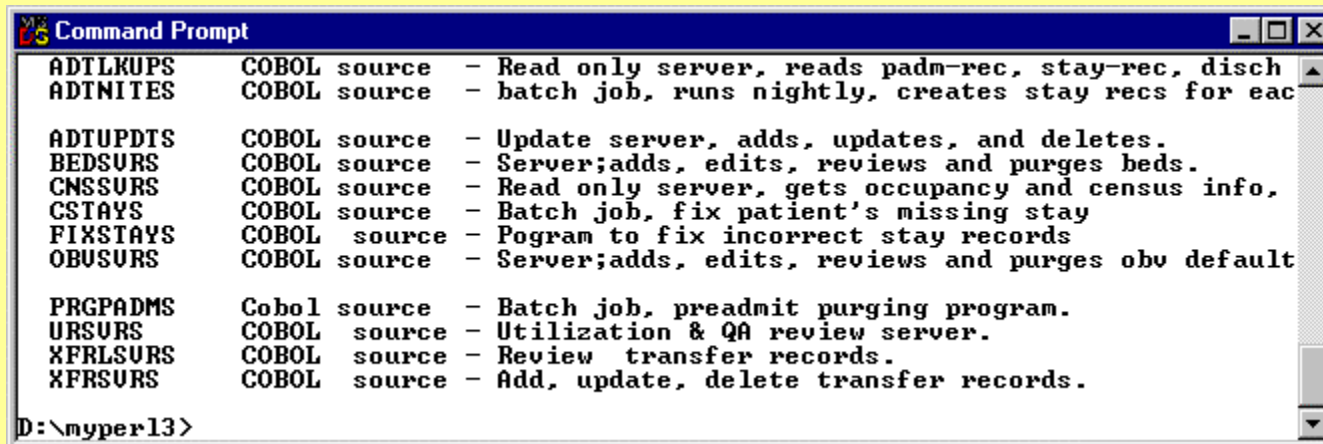
Read and review code on pages 9-34 of  
**Foundations of Perl Programming**

## Assignments

```
# Week 1
# Exercise 2a
# File: exe1_2a.pl

use strict;

# write a program that opens the file "index.txt" and returns a
# list of all the Cobol not scobol) programs to the console. Make
# sure you return all of them!
```



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The window contains a list of COBOL programs and their descriptions, displayed in a monospaced font. The programs listed are:

Program Name	Language	Description
ADTLKUPS	COBOL source	- Read only server, reads padm-rec, stay-rec, disch
ADTNITES	COBOL source	- batch job, runs nightly, creates stay recs for eac
ADTUPDTS	COBOL source	- Update server, adds, updates, and deletes.
BEDSURS	COBOL source	- Server;adds, edits, reviews and purges beds.
CNSSURS	COBOL source	- Read only server, gets occupancy and census info,
CSTAYS	COBOL source	- Batch job, fix patient's missing stay
FIXSTAYS	COBOL source	- Pogram to fix incorrect stay records
OBUSURS	COBOL source	- Server;adds, edits, reviews and purges obv default
PRGPADMS	Cobol source	- Batch job, preadmit purging program.
URSURS	COBOL source	- Utilization & QA review server.
XFRLSURS	COBOL source	- Review transfer records.
XFRSURS	COBOL source	- Add, update, delete transfer records.

The prompt shows the current directory as `D:\myper13>`.

Manual: p17-18

# Assignments

```
# Week 1
# Exercise 2b
# File: exel_2b.pl

use strict;

# enhance exel_2a.pl to offer the user a menu that lets them
# select the following:
#   - list of Cobol programs
#   - list of Scobol programs
#   - list entire contents of index.txt
# ...the program needs to execute once (you do not need to
#   write the program until the user wants to exit
```

```
Select Command Prompt
-----MENU-----
cob  - to display list of all Cobol Programs
scob - to display list of all Scobol Programs
all  - to display whole file

your choice>> cob
ADILKUPS  COBOL source - Read only server, reads padm-rec, stay-rec, disch
ADNITES   COBOL source - batch job, runs nightly, creates stay recs for eac

ADTUPDTS  COBOL source - Update server, adds, updates, and deletes.
BEDSURS   COBOL source - Server;adds, edits, reviews and purges beds.
CNSSURS   COBOL source - Read only server, gets occupancy and census info,
CSTAYS    COBOL source - Batch job, fix patient's missing stay
FIXSTAYS  COBOL source - Pogram to fix incorrect stay records
OBUSURS   COBOL source - Server;adds, edits, reviews and purges obv default

PRGPADMS  Cobol source - Batch job, preadmit purging program.
URSURS    COBOL source - Utilization & QA review server.
XFRLSURS  COBOL source - Review transfer records.
XFRSURS   COBOL source - Add, update, delete transfer records.

D:\myper13>
```

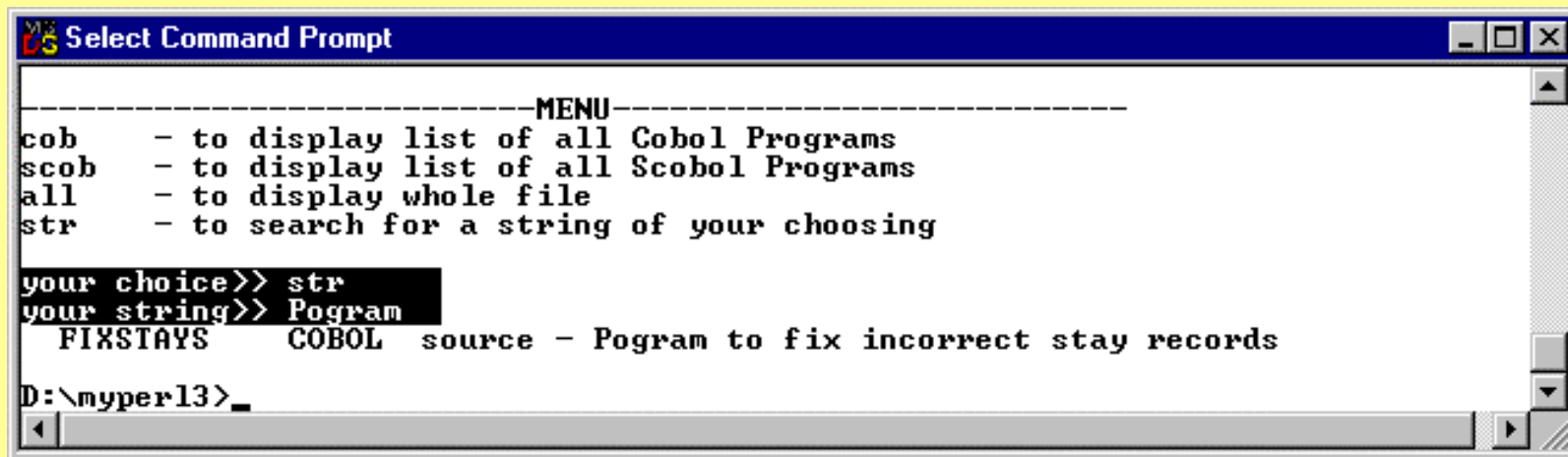
Manual: p17-18

## Assignments

```
# Week 1
# Exercise 2c
# File: exe1_2c.pl

use strict;

# enhance exe1_1b.pl to also allow the user to search for any
# string within the file:
```



```
Select Command Prompt

-----MENU-----
cob   - to display list of all Cobol Programs
scob  - to display list of all Scobol Programs
all   - to display whole file
str   - to search for a string of your choosing

your choice>> str
your string>> Pogram
FIXSTAYS COBOL source - Pogram to fix incorrect stay records

D:\myperl3>
```

Manual: p17-18

## Assignments

---

### Count words

- Provide a count of total number of words in a file
- Test program against file “cob1s.txt”

Countwords\_ANSWER.pl