Perl for Biologists

Session 2
March 11, 2015

Constants, variables and functions

Jaroslaw Pillardy
#!/usr/local/bin/perl
use warnings;
#this is my first Perl script
print "Hello, CBSU\n";

"shebang" notation – path to the program to interpret the script, must be the first line and start with #!

using external module/library

use warnings;

#this is my first Perl script

print "Hello, CBSU\n";

anything starting with # is a comment, unless it is #! in the first line

function to print out text

statement ends with a semicolon
"shebang" notation – path to the program to interpret the script, must be the first line and start with #!

anything starting with # is a comment, unless it is #! in the first line

statement ends with a semicolon

parentheses can be always omitted, unless it changes the meaning of expression

shebang notation – path to the program to interpret the script, must be the first line and start with #!

anything starting with # is a comment, unless it is #! in the first line

statement ends with a semicolon

parentheses can be always omitted, unless it changes the meaning of expression
Session 1 Exercises Review

1. Write a Perl program that prints your name and e-mail in the following format in one line:
first_name last_name <emailaddr@domain.edu>

/home/jarekp/perl_01/exercise1.pl

    #!/usr/local/bin/perl

    print 'Jarek Pillardy <jp86@cornell.edu>,'
    print "\n"
    print "Jarek Pillardy <jp86\@cornell.edu>\n";

2. Are the following modules installed on your BioHPC Lab machine?

    Net::Ping
    XML::Special
    Net::Telnet
    CBSU::HDF5

    >perl -MNet::Ping -e "print \"OK\n\"; "

Scalar Variables

Variable – a name for a block in computer memory holding something

Scalar variable – a variable containing only one element

Expression representing a constant value is a literal
Scalar Variables

Scalar variables in Perl always start with $

String variable:

```perl
$variable = "Jarek Pillardy";
```

- **variable name**
- **literal with a value to assign**
- **assignment operator**
Scalar Variables

Scalar variables in Perl always start with $.

Numerical variable:

$variable = 55;

variable name

literal with a value to assign

assignment operator
Variable names can contain letters, numbers and underscores

Case sensitive

Cannot start from number (digit)

$JarekPillardy

$jarekpillardy #different than above

$jp86_cornell_edu

$123jarek ← INVALID, starts with a number

$jp86@cornell.edu ← INVALID, contains @ and .
#!/usr/local/bin/perl

$svar = "\"Hello, CBSU\"\n";

$nvar = 55.55;

print $svar;

print $nvar;

print "\n";
#!/usr/local/bin/perl

$svar = "\"Hello, CBSU\"\n";

$nvar = 55.55;

print $svar;

print $nvar;

print "\n";

All scripts for this session can be copied from /home/jarekp/perl_02
in this case /home/jarekp/perl_02/script1.pl
>cp /home/jarekp/perl_02/script1.pl .

copies this script to your current directory
#!/usr/local/bin/perl

$svar = "\"Hello, CBSU\"\n";

$nvar = 55.55;

print $svar;

print $nvar;

print "\n";
#!/usr/local/bin/perl

$svar = "\"Hello, CBSU\"\n";

$nvar = 55.55;

print "$svar nvar=$nvar\n";
#!/usr/local/bin/perl

$svar = "\"Hello, CBSU\"\n";

$nvar = 55.55;

print "$svar nvar=$nvar\n";

[jarekp@cbsum1c2b014 perl_02]$ perl script2.pl
"Hello, CBSU"
nvar=55.55
[jarekp@cbsum1c2b014 perl_02]$
String Variables

Can be assigned both single quoted or double quoted strings

```perl
$variable1 = "Hello, CBSU\n";
$variable2 = 'Hello, CBSU\n';
```
String operators:

. **Concatenation**

$str1 = "Jarek" . " " . "Pillardy";
$str1 = "Jarek Pillardy";  # same as above

x **Repetition**

$str2 = "AAGT" x 3;
$str2 = "AAGTAAGTAAGT";  # same as above
Numerical Variables

Can be a floating point, integer, or non-decimal number

```perl
$variable1 = 1000000;       #integer
$variable1 = 1_000_000;     #integer, _ ignored
$variable1 = 1e+6;          #integer
$variable1 = 2.6182818285e-3;  #floating point
$variable1 = 0xfff34g;      #hexadecimal
$variable1 = 02351;         #octal
$variable1 = 0b101101;      #binary
```

ALL numerical variables are stored the same way in Perl – as double precision floating point numbers
Numerical Operators

$variable1 = 1000000 + 222;

$variable1 = 1000000 * 222;

$variable1 = 1000000 - 222;

$variable1 = 1000000 / 222;

$variable1 = 121**3; #power, =1771561

$variable1 = 1000000 % 222; #modulus, =112
Numerical Built-In Functions

- `abs`  #absolute value
- `sin`, `cos`, `tan`, `atan2`  #trigonometry
- `exp`, `log`, `sqrt`  #exponent, log, square root
- `int`  #convert to int
- `hex`, `oct`  #convert to hex oct
- `srand`, `rand`  #random numbers

```
$variable = 100;
print sqrt($variable);
```
String Built-In Functions (some)

substr($var, $start, $length)  #substring, 0-based
chomp($var)                    #removes trailing \n
index($var, $str)              #position of $str in $var
reverse($var)                  #reverse string
rindex($var, $str)             #reverse index
uc($var)                       #uppercase
lc($var)                       #lowercase
String Built-In Functions (some)

\texttt{ord}\hspace{2pt}($\texttt{var}$) \hspace{5pt} #converts character to its numerical ASCII value

\texttt{$num = ord("a")$} \hspace{5pt} #$num$ is now 97

\texttt{chr}\hspace{2pt}($\texttt{nvar}$) \hspace{5pt} #converts int into corresponding ASCII character

\texttt{$char = chr(99)$} \hspace{5pt} #$char$ is now "c"
# ASCII Table

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Char</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>000</td>
<td>NUL (null)</td>
<td>32</td>
<td>20</td>
<td>040</td>
<td>≤#32; Space</td>
<td>64</td>
<td>40</td>
<td>100</td>
<td>≤#64; @</td>
<td>96</td>
<td>60</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>001</td>
<td>SOH (start of heading)</td>
<td>33</td>
<td>21</td>
<td>041</td>
<td>≤#33; !</td>
<td>65</td>
<td>41</td>
<td>101</td>
<td>≤#65; A</td>
<td>97</td>
<td>61</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>002</td>
<td>STX (start of text)</td>
<td>34</td>
<td>22</td>
<td>042</td>
<td>≤#34; &quot;</td>
<td>66</td>
<td>42</td>
<td>102</td>
<td>≤#66; B</td>
<td>98</td>
<td>62</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>003</td>
<td>ETX (end of text)</td>
<td>35</td>
<td>23</td>
<td>043</td>
<td>≤#35; #</td>
<td>67</td>
<td>43</td>
<td>103</td>
<td>≤#67; C</td>
<td>99</td>
<td>63</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>004</td>
<td>EOT (end of transmission)</td>
<td>36</td>
<td>24</td>
<td>044</td>
<td>≤#36; $</td>
<td>68</td>
<td>44</td>
<td>104</td>
<td>≤#68; D</td>
<td>100</td>
<td>64</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>005</td>
<td>ENQ (enquiry)</td>
<td>37</td>
<td>25</td>
<td>045</td>
<td>≤#37; %</td>
<td>69</td>
<td>45</td>
<td>105</td>
<td>≤#69; E</td>
<td>101</td>
<td>65</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>006</td>
<td>ACK (acknowledge)</td>
<td>38</td>
<td>26</td>
<td>046</td>
<td>≤#38; &amp;</td>
<td>70</td>
<td>46</td>
<td>106</td>
<td>≤#70; F</td>
<td>102</td>
<td>66</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>007</td>
<td>BEL (bell)</td>
<td>39</td>
<td>27</td>
<td>047</td>
<td>≤#39; `</td>
<td>71</td>
<td>47</td>
<td>107</td>
<td>≤#71; G</td>
<td>103</td>
<td>67</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>010</td>
<td>BS (backspace)</td>
<td>40</td>
<td>28</td>
<td>050</td>
<td>≤#40; )</td>
<td>72</td>
<td>48</td>
<td>110</td>
<td>≤#72; H</td>
<td>104</td>
<td>68</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>011</td>
<td>TAB (horizontal tab)</td>
<td>41</td>
<td>29</td>
<td>051</td>
<td>≤#41; (</td>
<td>73</td>
<td>49</td>
<td>111</td>
<td>≤#73; I</td>
<td>105</td>
<td>69</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>012</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
<td>2A</td>
<td>052</td>
<td>≤#42; *</td>
<td>74</td>
<td>4A</td>
<td>112</td>
<td>≤#74; J</td>
<td>106</td>
<td>6A</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>013</td>
<td>VT (vertical tab)</td>
<td>43</td>
<td>2B</td>
<td>053</td>
<td>≤#43; +</td>
<td>75</td>
<td>4B</td>
<td>113</td>
<td>≤#75; K</td>
<td>107</td>
<td>6B</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>014</td>
<td>FF (NP form feed, new page)</td>
<td>44</td>
<td>2C</td>
<td>054</td>
<td>≤#44; ,</td>
<td>76</td>
<td>4C</td>
<td>114</td>
<td>≤#76; L</td>
<td>108</td>
<td>6C</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>015</td>
<td>CR (carriage return)</td>
<td>45</td>
<td>2D</td>
<td>055</td>
<td>≤#45; -</td>
<td>77</td>
<td>4D</td>
<td>115</td>
<td>≤#77; M</td>
<td>109</td>
<td>6D</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>016</td>
<td>SO (shift out)</td>
<td>46</td>
<td>2E</td>
<td>056</td>
<td>≤#46; .</td>
<td>78</td>
<td>4E</td>
<td>116</td>
<td>≤#78; N</td>
<td>110</td>
<td>6E</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>017</td>
<td>SI (shift in)</td>
<td>47</td>
<td>2F</td>
<td>057</td>
<td>≤#47; /</td>
<td>79</td>
<td>4F</td>
<td>117</td>
<td>≤#79; O</td>
<td>111</td>
<td>6F</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>020</td>
<td>DLE (data link escape)</td>
<td>48</td>
<td>30</td>
<td>060</td>
<td>≤#48; 0</td>
<td>80</td>
<td>50</td>
<td>120</td>
<td>≤#80; P</td>
<td>112</td>
<td>70</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>021</td>
<td>DC1 (device control 1)</td>
<td>49</td>
<td>31</td>
<td>061</td>
<td>≤#49; 1</td>
<td>81</td>
<td>51</td>
<td>121</td>
<td>≤#81; Q</td>
<td>113</td>
<td>71</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>022</td>
<td>DC2 (device control 2)</td>
<td>50</td>
<td>32</td>
<td>062</td>
<td>≤#50; 2</td>
<td>82</td>
<td>52</td>
<td>122</td>
<td>≤#82; R</td>
<td>114</td>
<td>72</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>023</td>
<td>DC3 (device control 3)</td>
<td>51</td>
<td>33</td>
<td>063</td>
<td>≤#51; 3</td>
<td>83</td>
<td>53</td>
<td>123</td>
<td>≤#83; S</td>
<td>115</td>
<td>73</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>024</td>
<td>DC4 (device control 4)</td>
<td>52</td>
<td>34</td>
<td>064</td>
<td>≤#52; 4</td>
<td>84</td>
<td>54</td>
<td>124</td>
<td>≤#84; T</td>
<td>116</td>
<td>74</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
<td>35</td>
<td>065</td>
<td>≤#53; 5</td>
<td>85</td>
<td>55</td>
<td>125</td>
<td>≤#85; U</td>
<td>117</td>
<td>75</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
<td>36</td>
<td>066</td>
<td>≤#54; 6</td>
<td>86</td>
<td>56</td>
<td>126</td>
<td>≤#86; V</td>
<td>118</td>
<td>76</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>027</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
<td>37</td>
<td>067</td>
<td>≤#55; 7</td>
<td>87</td>
<td>57</td>
<td>127</td>
<td>≤#87; W</td>
<td>119</td>
<td>77</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>030</td>
<td>CAN (cancel)</td>
<td>56</td>
<td>38</td>
<td>070</td>
<td>≤#56; 8</td>
<td>88</td>
<td>58</td>
<td>130</td>
<td>≤#88; X</td>
<td>120</td>
<td>78</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>031</td>
<td>EM (end of medium)</td>
<td>57</td>
<td>39</td>
<td>071</td>
<td>≤#57; 9</td>
<td>89</td>
<td>59</td>
<td>131</td>
<td>≤#89; Y</td>
<td>121</td>
<td>79</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>032</td>
<td>SUB (substitute)</td>
<td>58</td>
<td>3A</td>
<td>072</td>
<td>≤#58; :</td>
<td>90</td>
<td>5A</td>
<td>132</td>
<td>≤#90; Z</td>
<td>122</td>
<td>7A</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>033</td>
<td>ESC (escape)</td>
<td>59</td>
<td>3B</td>
<td>073</td>
<td>≤#59; ;</td>
<td>91</td>
<td>5B</td>
<td>133</td>
<td>≤#91; [</td>
<td>123</td>
<td>7B</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>034</td>
<td>FS (file separator)</td>
<td>60</td>
<td>3C</td>
<td>074</td>
<td>≤#60; &lt;</td>
<td>92</td>
<td>5C</td>
<td>134</td>
<td>≤#92; \</td>
<td>124</td>
<td>7C</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>035</td>
<td>GS (group separator)</td>
<td>61</td>
<td>3D</td>
<td>075</td>
<td>≤#61; =</td>
<td>93</td>
<td>5D</td>
<td>135</td>
<td>≤#93; ]</td>
<td>125</td>
<td>7D</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>036</td>
<td>RS (record separator)</td>
<td>62</td>
<td>3E</td>
<td>076</td>
<td>≤#62; &gt;</td>
<td>94</td>
<td>5E</td>
<td>136</td>
<td>≤#94; ^</td>
<td>126</td>
<td>7E</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>037</td>
<td>US (unit separator)</td>
<td>63</td>
<td>3F</td>
<td>077</td>
<td>≤#63; ?</td>
<td>95</td>
<td>5F</td>
<td>137</td>
<td>≤#95; _</td>
<td>127</td>
<td>7F</td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com
Finding more about functions and modules

Use perldoc command

>perldoc -f ord

>perldoc Net::Telnet

Search perldoc on the web

http://perldoc.perl.org/perlfunc.html
Binary assignment

```perl
$variable = 1;
$variable = $variable + 3;  # variable is now 4
$variable += 3;             # variable is now 7,
                            # same effect

$svar = "Jarek";
$svar .= " Pillardy";        # variable is now
                            # "Jarek Pillardy"

$svar = "My name is " . $svar;
    #"My name is Jarek Pillardy"

or

$svar .= "My name is ";   # NOT SAME AS ABOVE
    #"Jarek PillardyMy name is "
```
#!/usr/local/bin/perl

$svar = "Hello, CBSU\n";
print "svar = $svar";
$nvar = 55.55;
print "nvar = $nvar\n";

$nvar += 10;
print "nvar is now $nvar\n";

$svar .= "Hello again\n";
print $svar;
$svar = "Hello first\n" . $svar;
print $svar;
#!/usr/local/bin/perl

$svar = "Hello, CBSU\n";
print "svar = $svar"
$nvar = 55.55;
print "nvar = $nvar\n"

$nvar += 10;
print "nvar is now $nvar\n"

$svar .= "Hello again\n";
print $svar;
$svar = "Hello first\n" . $svar;
print $svar;
#!/usr/local/bin/perl

$svar = "Hello, CBSU\n";
print "svar = $svar\n";

$svar1 = $svar;
chomp($svar1);
print "svar1 = $svar1\n";

$svar1 = substr($svar1, 0, 5);
print "svar1 = $svar1\n";

print index($svar, ",," ) . "\n";
print index($svar1, ",," ) . "\n";

print uc($svar1) . "\n";
#!/usr/local/bin/perl

$svar = "Hello, CBSU\n";
print "svar = $svar\n";

$svar1 = $svar;
chomp($svar1);
print "svar1 = $svar1\n";

$svar1 = substr($svar1, 0, 5);
print "svar1 = $svar1\n";

print index($svar, "," ) . "\n";
print index($svar1, "," ) . "\n";

print uc($svar1) . "\n";
Automatic Variable Conversion
or Variable Interpolation

Perl is a context-based language, variables will be converted (or interpolated) as needed

```
$nvar = 55.5;
$svar = "The number $nvar is " . $nvar;
```

Perl expects string since string operation is being used

$nvar is converted to string and concatenated with preceding string
Automatic Variable Conversion
or *Variable Interpolation*

Perl is a context-based language, variables will be converted (or *interpolated*) as needed

```perl
$nvar = 55.5;
$svar = "2variable6str ";

$nnn = $nvar * $svar;
#$nnn is now 55.5*2 = 111
```

Perl expects number since numeric operation is being used

$svar is converted to number, all trailing letters and non-numbers are discarded, if there are no starting numbers the result is 0
#!/usr/local/bin/perl

$nvar = 55.5;
$svar = "The number nvar is ". $nvar;
$nvar;
print "$svar\n";

$nvar = 55.5;
$svar = "2variable1str 3a";
$nnn = $nvar * $svar;
print "$nnn\n";

print "55.5" . 2 * 7;
print "\n";
#!/usr/local/bin/perl

$nvar = 55.5;
$svar = "The number nvar is " . $nvar;
print "$svar
"

$nvar = 55.5;
$svar = "2variable1str 3a";
$nnn = $nvar * $svar;
print "$nnn
"

print "55.5" . 2 * 7;
print "\n";
#!/usr/local/bin/perl

use warnings;

$nvar = 55.5;
$svar = "The number nvar is ". $nvar;
print "$svar\n";

$nvar = 55.5;
$svar = "2variable1str 3a";
$nnn = $nvar * $svar;
print "$nnn\n";

print "55.5 . 2 * 7;"
print "\n";
#!/usr/local/bin/perl

use warnings;

$nvar = 55.5;
$svar = "The number nvar is " . $nvar;
print "$svar\n";

$nvar = 55.5;
$svar = "2variable1str 3a";
$nnn = $nvar * $svar;
print "$nnn\n";

print "55.5" . 2 * 7;
print "\n";
#!/usr/local/bin/perl

use warnings;

$nvar = 2;
print "\$nvar\n";

$nvar1 = $nvar * 10;
print "\$nvar1\n";

$nvar1 = $nvar * 010;
print "\$nvar1\n";

$nvar1 = $nvar * "010";
print "\$nvar1\n";
#!/usr/local/bin/perl

use warnings;

$nvar = 2;
print "\n$nvar\n";

$nvar1 = $nvar * 10;
print "\n$nvar1\n";

$nvar1 = $nvar * 010;
print "\n$nvar1\n";

$nvar1 = $nvar * "010";
print "\n$nvar1\n";
What if we use a variable that has not been declared?

```perl
print "=>\$newvar<=\n";
```
What if we use a variable that has not been declared?

```perl
print "=>\$newvar<=\n";
```

No problem, any new variable is assigned a special value: `undef`

It will interpolate to

- an empty string in string context
- 0 in numerical context

**BEWARE: USING UNINITIALIZED VARIABLE IS A VERY COMMON SOURCE OF ERRORS. USE WARNINGS, IT HELPS.**
#!/usr/local/bin/perl

use warnings;

print "=>$newvar<=$n";
#!/usr/local/bin/perl

use warnings;

print "=>\$
newvar
<=\n";
Numbers

ALL numbers are represented in Perl as *double-precision floating point* numbers

On 64 bit machines each takes 8 bytes = 64 bits
#!/usr/local/bin/perl

$nvar = 2**1023;
print "$nvar\n";
$nvar = 2**1024;
print "$nvar\n";

$nvar = 2**-1074;
print "$nvar\n";
$nvar = 2**-1075;
print "$nvar\n";
#!/usr/local/bin/perl

$nvar = 2**1023;
print "$nvar
";
$nvar = 2**1024;
print "$nvar
";

$nvar = 2**-1074;
print "$nvar
";
$nvar = 2**-1075;
print "$nvar
";

[jarekp@cbsum1c2b014 perl_02]$ perl script8.pl
8.98846567431158e+307
inf
4.94065645841247e-324
0
[jarekp@cbsum1c2b014 perl_02]$
#!/usr/local/bin/perl

$nvar = log(10);
print "$nvar\n";
$nvar = $nvar + 1e-14;
print "$nvar\n";
$nvar = $nvar + 1e-15;
print "$nvar\n";
#!/usr/local/bin/perl

$nvar = log(10);
print "$nvar\n";
$nvar = $nvar + 1e-14;
print "$nvar\n";
$nvar = $nvar + 1e-15;
print "$nvar\n";

15 digit accuracy:
1e-14 + 1 different than 1
1e-15 + 1 same as 1
Exercises

1. In a Perl program create a string representing a 54 bp DNA strand consisting of 6 repeats, store it in a variable. Create another variable containing the above DNA reversed. Create the third variable storing a subsequence of the original sequence from position 31 to position 47. Print all three.
   Hint: Use string functions and operators to create strings from a repeat.

2. Use perldoc to find out how rand() and srand() functions work. Write a Perl program that produces a 17 character string composed of random lower case letters, store it in a variable and print it out. Run the program several times and compare the results.
   Hint: use chr(), int() functions and ASCII table.