Abhishek Pyakurel

5/15/2020

tk3223

Analysis of Perl

1. Classification:

- Multi-paradigm
- Imperative, functional as well as object-oriented
- Interpreted, dynamic language
- Object-oriented was very difficult achievable but very complicated
- It favoured procedural style
- It's more useful for its regex-search engine
- For text manipulation

2. readability, write-ability, orthogonality, cost, and reliability of the language:

- Its similar to c/c++/java where you need to declare variable type
- But compiler can recognize and convert between strings and int easily
- Readability is similar to python but little more code.
- Write-ability is where it lacks
- Even though my algorithm was not very complex, it was even crashing a lot
- Well my algorithm that was making random numbers was producing many errors
- So, I think it's not very orthogonal, it affects other functions.
- It is even slower compared to python

- Well it crashed two times

3. syntax and semantics rules of the language:

- Its syntax is similar to c/c++
- It semantics is also similar to c/c++
- Variable type needs to be declared beforehand.
- There is type checking
- It is done at run-time
- It is even like a english language
- My is used to declare variables of any type
- There was difference in operator checking too
- == was different than eq and so on

4.Implementation methods:

- It was dynamic type checking
- It was all done at run-time
- Compiler even gave exact error at exact line
- Bug was frequent but it was easier to debug for sure

5.static and dynamic aspects of the language:

- Static is used while type checking with the use of strict library
- Without strict it uses dynamic type checking
- It is a strongly typed language

```
#!/usr/bin/perl
# Library used
use warnings;
use strict;
use File::Slurp;
```

6.projects this languages may be best suited for:

- I found it very slow, it was even crashing,
- May be due to my algorithm
- It is favoured for text manipulation
- To be honest I really liked it.
- Once i figure out how it works
- Web-programming
- scripting

7.data types and control structures:

- Scalars
- Arrays and hashes
- There were three ways to declare a scalar, but could be declared any other three ways.
- With @,$,%

```
my $x = '';
my $xx = '';
my $xxx = '';
```

8.polymorphism:

- It supported polymorphism but its not recommended to use object oriented in Perl

9.Object-Oriented:

- Well certainly object oriented programming was very difficult in perl
- It feels like it has been just thrown at it
- Without careful thought
- I even read a book, it straight jumped to web programming than teaching objects.

10.pointers/references:

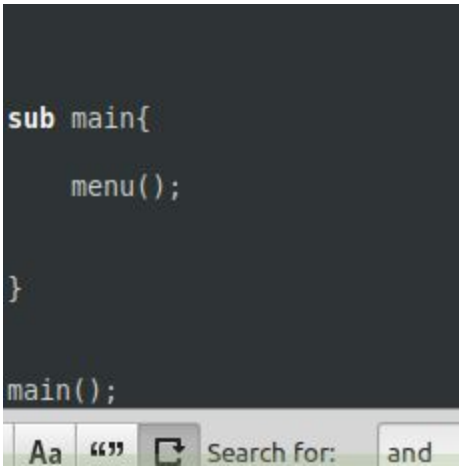
- It uses references more than pointers
- It also doesn't seem to keep track of previous values
- So, you let know compiler what you are doing

11.Memory Management:

- It has garbage collection.
- simple scheme called reference counting

12. functions/sub-programs:

- Functions/Sub-programs were a little weird.
- It was similar to the scheme in many ways.
- Where a function doesn't need to be declared with ()
- If its declared with () it treated as a variable just like scheme
- But while calling the function you need to call it with ()
- Just like other programming language



13.lambda functions:

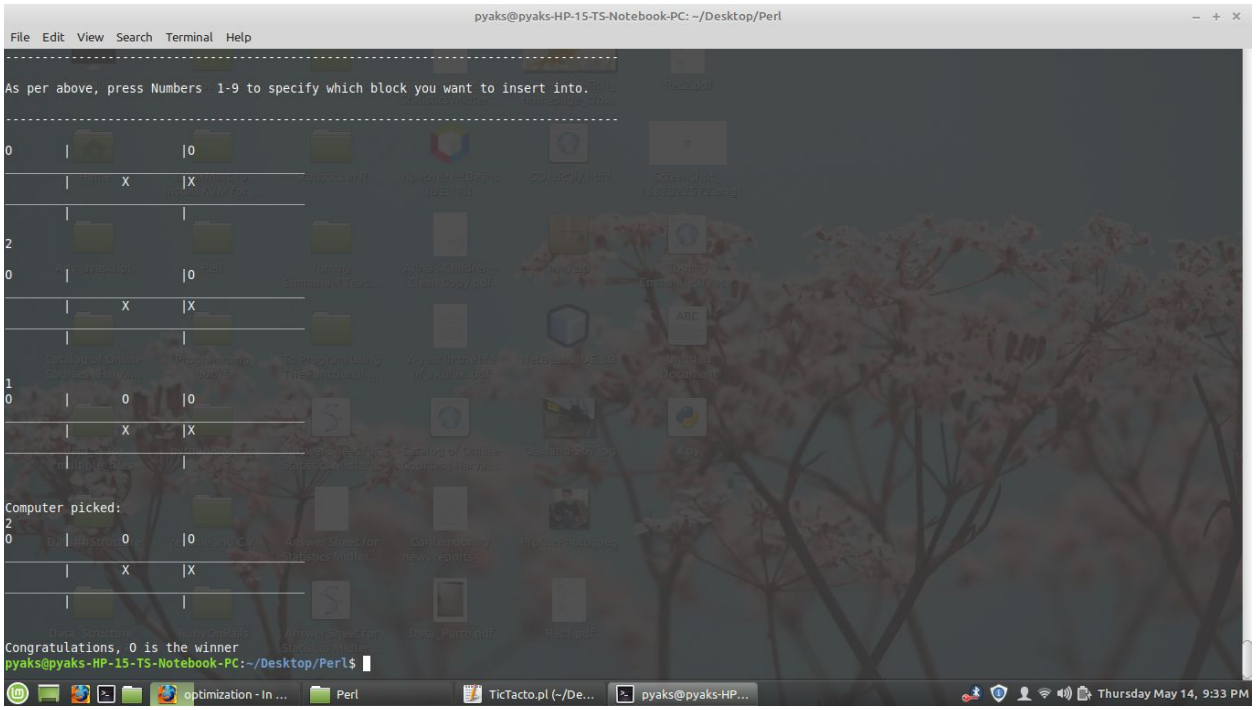
- Perl has an intensive library.
- To use lambda function you need to download one using cpan

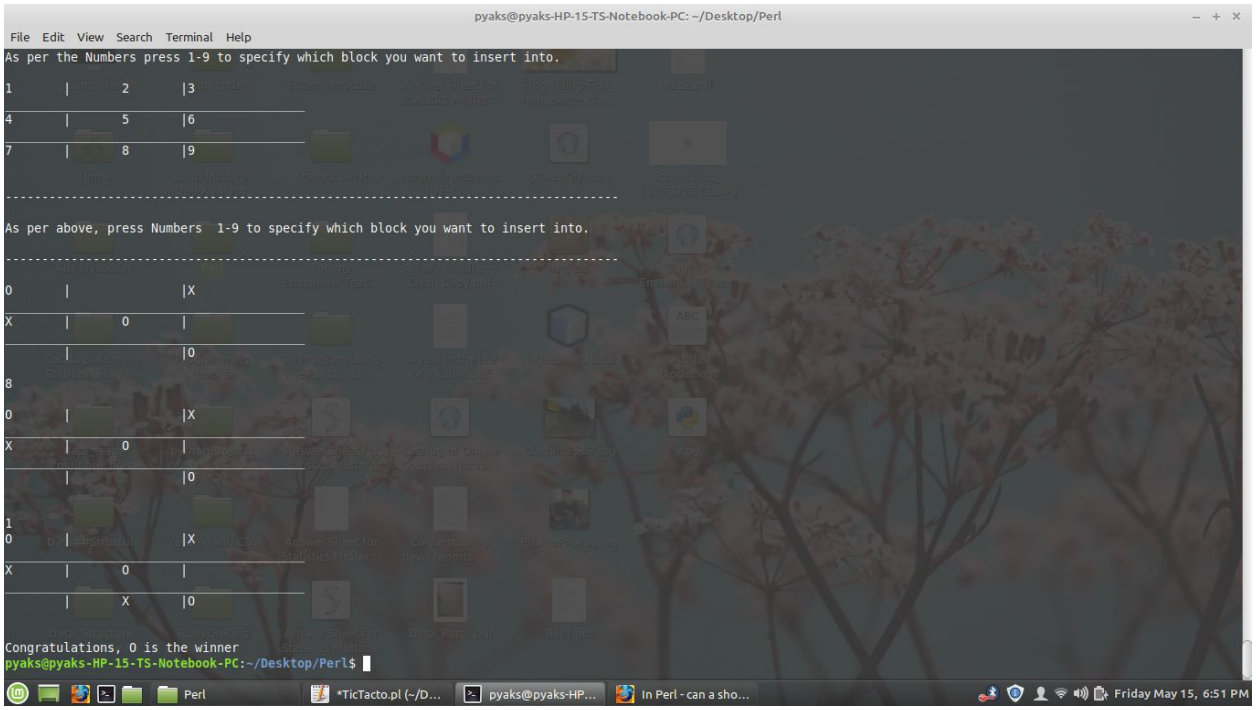
14.Others:

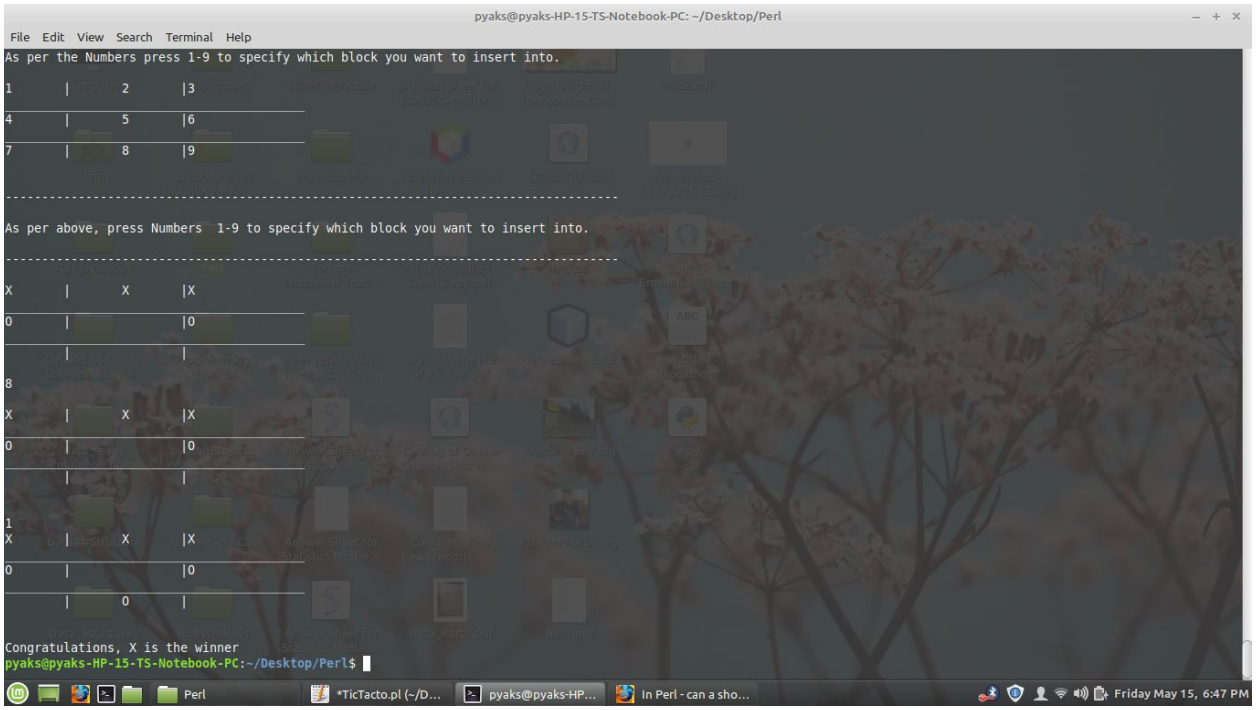
- Well it used a lot of short circuit evaluations. There were ways to get around it.
- But for beginners it's tricky because you think all your operators would be evaluated.
- But no.
- == was not the same and efficient as eq and so on

```
# for second player
sub emptyCheker2 {
if ((length($x) == 1 and $input == 1) or (length($xx) == 1 and $input == 2) or (length($xxx) == 1 and $input ==
    or (length($y) == 1 and $input == 4) or (length($yy) == 1 and $input == 5) or (length($yyy) == 1 and $input
    or (length($z) == 1 and $input == 7) or (length($zz) == 1 and $input == 8) or (length($zzz) == 1 and $input
    )
    {
```

Output:


pyaks@pyaks-HP-15-TS-Notebook-PC: ~/Desktop/Perl
File Edit View Search Terminal Help
As per above, press Numbers 1-9 to specify which block you want to insert into.
Computer picked:
Congratulations, O is the winner
pyaks@pyaks-HP-15-TS-Notebook-PC:~/Desktop/Perl$
optimization - In ...
Perl
TicTacto.pl (~/De...
pyaks@pyaks-HP...
Thursday May 14, 9:33 PM


pyaks@pyaks-HP-15-TS-Notebook-PC: ~/Desktop/Perl
File Edit View Search Terminal Help
As per the Numbers press 1-9 to specify which block you want to insert into.
As per above, press Numbers 1-9 to specify which block you want to insert into.
Congratulations, O is the winner
pyaks@pyaks-HP-15-TS-Notebook-PC:~/Desktop/Perl$
Perl
*TicTacto.pl (~/D...
pyaks@pyaks-HP...
In Perl - can a sho...
Friday May 15, 6:51 PM


pyaks@pyaks-HP-15-TS-Notebook-PC: ~/Desktop/Perl
File Edit View Search Terminal Help
As per the Numbers press 1-9 to specify which block you want to insert into.
1 | 2 |3
4 | 5 |6
7 | 8 |9
As per above, press Numbers 1-9 to specify which block you want to insert into.
X | X |X
0 | |0
| |
8
X | X |X
0 | |0
| |
1
X | X |X
0 | |0
| 0 |
Congratulations, X is the winner
pyaks@pyaks-HP-15-TS-Notebook-PC:~/Desktop/Perl$
Perl
*TicTacto.pl (~/D...
pyaks@pyaks-HP...
In Perl - can a sho...
Friday May 15, 6:47 PM