CS 3430: Python & Perl
Assignment 4: Sorting and Filtering Random Toops

Vladimir Kulyukin
Department of Computer Science
Utah State University

1. Learning Objectives

1. Py List Comprehension
2. Py & PL Custom Sorting
3. Py Filtering & PL Grepping
4. Py Tuples & PL List References

2. Random Toops

In this assignment, you will manipulate 4-tuples (4-toops) of random positive and negative integers. You will generate lists of 4-toops and do some custom sorting, filtering, and grepping of these lists.

Step 1: Defining the function/subroutine **random_4_toop** that takes two non-negative integers **a** and **b** and returns a 4-toop where each element is a positive or negative integers whose magnitude is in **[a, b]**. The signs of the individual differences should also be assigned randomly.

```
def random_4_toop(a, b):  // your code
sub random_4_toop {
   my ($a, $b) = @_;
   // your code
   return \@toop; ## return an array reference
}
```

Below are the Py and PL test functions that you should use to test the accuracy of your implementation of Step 1.

```
def test_step_1():
  toop1 = random_4_toop(1, 100)
  toop2 = random_4_toop(1, 100)
  toop3 = random_4_toop(1, 100)
  print 'toop1 = ' + str(toop1)
  print 'toop2 = ' + str(toop2)
  print 'toop3 = ' + str(toop3)
```

```
sub test_step_1 {
   ## generate three random 4-toops whose elements are in [1, 100]
   ## and print them out
   my $toop1 = random_4_toop(1, 100);
   my $toop2 = random_4_toop(1, 100);
```

```
    my $toop3 = random_4_toop(1, 100);
    ## . is the PL operator for string concatenation
    print '@toop1 = ' . "@{$toop1}" . "\n";
    print '@toop2 = ' . "@{$toop2}" . "\n";
    print '@toop2 = ' . "@{$toop3}" . "\n";
}
```

A possible Py shell output of **test_step_1():**

```
toop1 = (1, -25, 57, -23)
toop2 = (-47, -13, -14, 72)
toop3 = (-69, 26, 95, -92)
```

A possible PL command line output of **test_step_1()**:

```
@toop1 = 5 -93 8 94
@toop2 = 95 66 81 50
@toop2 = -20 49 32 -61
```

Of course, repeated multiple runs of **test_step_1** in either language should give you different toops.

Step 2: Use Py list comprehension to define the function/subroutine **gen_2nd_deg_polys(a, b, n)** that returns a list of **n** random 4-toops whose elements are in **[a, b]** where **a** and **b** are defined in Step 1.

```
def gen_random_4_toops(a, b, n): ## your code
```

The PL equivalent gen_random_4_toops should take the same input arguments and return a reference to a list of random 4-toop references.

```
sub gen_random_4_toops {
    my ($a, $b, $n) = @_; my @toops = ();
   ## your code
    return \@toops; ## return an array reference
}
```

Here are the Py & PL tests you can use in testing your implementation.

```
def test_step_2():
  toops = gen_random_4_toops(1, 100, 5)
  for n, toop in zip(xrange(1, 6), toops):
     print 'toop ' + str(n) + ": " + str(toop)

sub test_step_2 {
    ## generate 5 random 4-toops and print them out
```

```
   my $toops = gen_random_4_toops(1, 100, 5);
   my $tn = 1;
   foreach(@{$toops}) {
           print "toop $tn:\t(@{$_});\n";
           $tn++;
   }
}
```

A possible Py shell output of **test_step_2()**:

```
toop 1: (30, 51, 96, -13)
toop 2: (-80, -53, -49, 71)
toop 3: (79, -14, -57, -28)
toop 4: (28, 33, -76, -54)
toop 5: (-88, 69, -87, -93)
```

A possible PL command line window output of **test_step_2():**

```
toop 1: (-97 93 30 -49);
toop 2: (-30 -46 6 49);
toop 3: (3 91 84 -68);
toop 4: (60 -5 -2 -96);
toop 5: (31 24 -78 88);
```

Step 3: Define a Py function and a PL sub **sort_random_4_toops_by_sum** that takes a list of random 4-toops, sorts then non-destructively by the sum of each toop from largest to smallest, and returns the sorted list.

```
def sort_random_4_toops_by_sum(toops): ## your code
```

The PL equivalent should do the same. You may want to use the following use pragmas when working on the PL version of sort_random_4_toops_by_sum:

```
use strict;
use warnings;
use 5.10.0;
use List::Util qw(sum); ## import the sum sub from List::Util package that allows you to compute the sum
                        ## integers in arbitrary lists.

sub sort_random_4_toops_by_sum {
   my @sorted_toops = ();
   ## your code here
   return \@sorted_toops;
}
```

The Py and PL tests for Step 3 are as follows.

```
def test_step_3():
  toops = gen_random_4_toops(1, 100, 5)
  print "---- random 4-toops:"
  for n, toop in zip(xrange(1, 6), toops):
    print 'toop ' + str(n) + ": " + str(toop) +\
          'sum = ' + str(sum(toop))
  print "\n---- random 4-toops sorted by sum:"
  sorted_toops = sort_random_4_toops_by_sum(toops)
  for n, toop in zip(xrange(1, 6), sorted_toops):
    print 'toop ' + str(n) + ": " + str(toop) +\
          'sum = ' + str(sum(toop))
```

```
sub test_step_3 {
    my $toops = gen_random_4_toops(1, 100, 5);
    print "---- random 4-toops:\n";
    my $tn = 1;
    foreach(@{sort_random_4_toops_by_sum(@{$toops})}) {
              print "toop $tn:\t(@{$_}); " . sum(@{$_}) . "\n"; $tn++;
    }
    print "---- random 4-toops sorted by sum:\n";
    my $sorted_toops = sort_random_4_toops_by_sum(@{$toops});
    $tn = 1;
    foreach(@{$sorted_toops}) {
              print "toop $tn:\t(@{$_}); sum = ". sum(@{$_}) . "\n";
              $tn++;
    }
}
```

Below is a possible Py shell output of my testing **test_step_3()**:

```
---- random 4-toops:
toop 1: (-34, -85, -89, -6); sum = -214
toop 2: (94, -31, -26, 50); sum = 87
toop 3: (10, 51, 38, 3); sum = 102
toop 4: (53, 29, -7, -5); sum = 70
toop 5: (-93, -50, 82, 75); sum = 14

---- random 4-toops sorted by sum:
```

```
toop 1: (10, 51, 38, 3); sum = 102
toop 2: (94, -31, -26, 50); sum = 87
toop 3: (53, 29, -7, -5); sum = 70
toop 4: (-93, -50, 82, 75); sum = 14
toop 5: (-34, -85, -89, -6); sum = -214
```

Here is a possible command window output of the PL **test_step_3**:

```
---- random 4-toops:
toop 1: (93 97 78 77); 345
toop 2: (-1 82 92 -45); 128
toop 3: (62 25 -31 -4); 52
toop 4: (-77 -86 18 36); -109
toop 5: (-72 -96 -83 -6); -257
---- random 4-toops sorted by sum:
toop 1: (93 97 78 77); sum = 345
toop 2: (-1 82 92 -45); sum = 128
toop 3: (62 25 -31 -4); sum = 52
toop 4: (-77 -86 18 36); sum = -109
toop 5: (-72 -96 -83 -6); sum = -257
```

Step 4: Write a function/subroutine **filter_random_4_toops_by_sum** that takes a list of toops and a numerical threshold and filters the list of toops by the sum whose value strictly exceeds the threshold.

```
def filter_random_4_toops_by_sum(toops, thresh):
  ## your code
```

The PL version of filter_random_4_toops_by_sum should return the reference to a list of the toops also filtered, i.e., grepped, by the sum above the thresh.

```
sub filter_random_4_toops_by_sum {
    my ($toops, $thresh) = @_;
    my @filtered_toops = ();
    ## your code;
    return \@filtered_toops;
}
```

Below are the Py and PL tests of Step 4:

```
def test_step_4():
  toops = gen_random_4_toops(1, 100, 5)
  print "---- random 4-toops:"
  for n, toop in zip(xrange(1, 6), toops):
    print 'toop ' + str(n) + ": " + str(toop) +\
          'sum = ' + str(sum(toop))
  thresh = 0
  print "\n---- random 4-toops filtered by sum:"
  filtered_toops = filter_random_4_toops_by_sum(toops, thresh)
  for n, toop in zip(xrange(1, 6), filtered_toops):
    print 'toop ' + str(n) + ": " + str(toop) +\
          '; sum = ' + str(sum(toop))

sub test_step_4 {
    my $toops = gen_random_4_toops(1, 100, 5);
    print "---- random 4-toops:\n";
    my $tn = 1;
    foreach(@{sort_random_4_toops_by_sum(@{$toops})}) {
              print "toop $tn:\t(@{$_}); sum = " . sum(@{$_}) . "\n"; $tn++;
    }
    my $thresh = 0;
    print "\n---- random 4-toops filtered by sum above $thresh:\n";
    my $filtered_toops = filter_random_4_toops_by_sum($toops, $thresh);
    $tn = 1;
    foreach(@{$filtered_toops}) {
              print "toop $tn:\t(@{$_}); sum = ". sum(@{$_}) . "\n";
              $tn++;
    }
}
```

A possible Py shell output of **test_step_4**:

```
---- random 4-toops:
toop 1: (-24, 13, 57, -41)sum = 5
toop 2: (45, -57, -94, 45)sum = -61
toop 3: (-48, 96, 88, 89)sum = 225
toop 4: (60, 48, -11, 55)sum = 152
toop 5: (-18, -33, 84, -16)sum = 17

---- random 4-toops filtered by sum:
toop 1: (-24, 13, 57, -41); sum = 5
toop 2: (-48, 96, 88, 89); sum = 225
toop 3: (60, 48, -11, 55); sum = 152
toop 4: (-18, -33, 84, -16); sum = 17
```

A possible PL shell output of **test_step_4**:

```
---- random 4-toops:
toop 1: (49 49 4 64); sum = 166
toop 2: (-2 16 57 76); sum = 147
toop 3: (-94 93 -48 85); sum = 36
toop 4: (19 -47 14 38); sum = 24
toop 5: (-57 80 -60 -35); sum = -72

---- random 4-toops filtered by sum above 0:
toop 1: (19 -47 14 38); sum = 24
toop 2: (-94 93 -48 85); sum = 36
toop 3: (49 49 4 64); sum = 166
toop 4: (-2 16 57 76); sum = 147
```

4. What to Submit

Save your Py solution in **random_toops.py** and your PL solution in **random_toops.pl** and submit both files in Canvas.

Happy Hacking!