Stanford Code Poetry Slam 2.0

23 January 2015

Wallenberg Hall 124

Poetizer.py + Poemreader.py, Ross Goodwin

https://github.com/rossgoodwin/poetizer/blob/master/poetizer.py
https://github.com/rossgoodwin/poetizer/blob/master/poemreader.py

Angler: In Search of Target Fish, Alireza Ebrahimi

O Angler, plan fish net to sea
Throw thy net for fish catch seize.
Rise, gather, what will be
Set goal target, enter sea.
Elusive fish, come to me.
Lift thy load, in strong sea breeze

Seek and search, find what is mine
Stir the net, go round and round
Grab each fish, one by one
Compare to target, everyone
Elusive fish, decide what's won
Repetitive, tireless, until found.

Frustrated, dismayed, the Angler will be
As hope springs anew when target comes to view
Joy in Angler's heart abound
"Fish Is Found"
Or if ever, "The Not Found"
End plan sequence, solemn and true.

Dissatisfied Angler what to do but
Think new fish with much more heart
Wonder whether should re-do, alas
Adjust condition; include new class
Begin anew amass
With much more vigor, again to start.

```
#include <iostream>
#include <fstream>
using namespace std;
main(){
string mindfish, netfish ;
while (1){ //infinite loop intended
cout << “ENTER THE DESIRED FISH: “;
cin >> mindfish;
ifstream fin( “seafish.txt” );
while( fin >> netfish ){
if( mindfish == netfish ){
cout << “FISH IS FOUND “<<endl;
fin.close();
return 1;}//end if
} // inner loop
cout<<” NOT FOUND-CHANGE YOUR MIND FISH OR EXPAND FISH NET:”<<endl;
fin.close();
} // outer loop
return 0;
} // end main
```

SHORT CIRCUIT WEEK, Derek Dadian-Smith

```
# SHORT CIRCUIT WEEK

stress = True
clarity = True
sunday = stress or clarity

doubt = True
freedom = True
monday = doubt or freedom

distraction = True
motivation = True
tuesday = distraction or motivation

darkness = True
inspiration = True
wednesday = darkness or inspiration
```

```
fear = True
creation = True
thursday = fear or creation

depression = True
love = True
friday = depression or love

suffering = True
enlightenment = True
saturday = suffering or enlightenment


# d.w.d-s
```

Addiction, Oishi Banerjee

```
void addiction(){
    int mind = 1;
    while (true){
        //what goes around wraps around
        //and we climb the peaks to find they touch the abyss
        mind = mind+1;
        if (mind<0){
            break;
        }
    }
}
```

Modulate a Thousand Times More, Peter Wildman

https://vimeo.com/107008770

```
long  theTime;
color beBlue = color(0, 0, 253);
long  forNow;
color beGreen = color(0, 255, 0);
color theBleedingRed = color(255, 0, 0);
long  more = 2;
long  foreverInAMoment;

void setup() {
  background(0);
  size(1000, 1000);
  theTime = 1000 * more;
}
```

```
void draw() {
 for(long ingTheTime =
      foreverInAMoment;
        theTime > 0;
        theTime --) {
          set(int
      (random
     (int
        (theTime),
     (int
        (foreverInAMoment))),
                     222, theBleedingRed);
          }

 set(1+1+1+ int
        (random
        (1,1*1*1000+more)),
        int
      (theTime +
        (random
        (1+2,1000+more)
        )),
2+beBlue);
          set(int
                (foreverInAMoment) + int
                (random
                (1,1000)),
          int
                (theTime +
            int
                (random
                (1+2,1000+more)
          )),
 2+beGreen);
for(int
        thisMoment = 2; beGreen > 2 % 1000*more;
        //still
        thisMoment++){
  }
 }
```

Modern Narrative, Steven Wingate

```
print ("Modern Narrative (TM) by Steven Wingate")

print ("Adapted from Through the Park, copyright (c) 2013 Nick Montfort <nickm@nickm.com>" )

import random, textwrap

for i in range(0, 7):
  text = ["The hero discovers a hidden wound",
        "The hero and sidekick indulge in peaceful, innocent pleasure",
        "Sexual tension escalates",
        "The calm before the storm, featuring a fleeting recognition of the hero's need for companionship",
        "The hero and romantic interest engage in public pleasure that verges on a declaration of love",
        "Hero and sidekick declare that the world must be saved through love",
        "A collective moment of introspection in which all principals question whether the current endavor is
worthwhile",
        "A private moment of introspection in which the hero questions his/her self-worth",
        "A musical scene which uplifts a community that will soon be threatened with destruction",
        "A musical scene which uplifts a community that has been saved",
        "A penultimate conflict, easily misinterpreted as the final conflict by those untrained in modern
narrative",
 "The sidekick dies but is returned to life through the hero's ministrations",
          "The intrinsic value of fidelity to the self is called into question by all",
          "The enemy, though nearly victorious, is rebuffed and reduced in power",
          "The enemy, temporarily defeated but encouraged by his/her increased powers, retreats",
          "The romantic interest saves the hero from the temptations of a henchman",
          "The team disbands in despair and reunites only hesitantly",
          "The team, on the verge of fracture, bands together with renewed energy",
          "Everything is explained by the master/mentor",
          "Peace returns to the community and the principals' emotional contradictions are healthily resolved",
          "The romantic interest, blinded by grief and remorse, falls into a trap and unwittingly abets the enemy",
          "Romance flares up between the hero and romantic interest, but is mutually spurned",
          "The hero is congratulated for saving the world once more",
          "The hero returns home to find that things are not as believed",
          "A misanthropic moment for the hero, caused by the dastardly machinations of a henchman",
          "A henchman dies ignominiously",
          "A henchman is transformed into an ignominious animal, its nature reflecting said henchman's quasi-
bestial greed",
          "The sidekick, exasperated by the hero's sudden personality change, nearly abandons the quest",
          "Ossified tropes from previous examples of the genre, as necessary",]
    phrases = 6 + random.randint(1,4)
    while len(text) > phrases:
        text.remove(random.choice(text))
    print ("\nModern Narrative (TM) Book " + str(i+1) + "\n\n" + \
        textwrap.fill(". ... ".join(text) + ".", 60) + "\n\n\n" )
```

fidelity, Guilherme Kerr

```
<SCRIPT LANGUAGE=javascript>
<!—
fidelity = function(still){
        love={poem:document,search:{'for':'&hearts;'},last:window,
        I:{had:'a'},'true':'love'|| 'for a'},vainMoment=
        love.search.for.split(""the anguish's"").length,time=love.lasts, more =
        love.flame= true;/*but*/love.immortal=/*is*/!love.flame

        function/*of*/liveIt(sorely){while(still in love){
        still=love.last.confirm(still+' '+love.true+'?')
        time=liveIt(more);a=""love.poem.write(love.search.for+time.live+'...')""
        love.last.setTimeout(a,vainMoment++||eternal)};

        love.I.had = love.search.for.valueOf(""time's"").length
        love.intensity=love.I.had/love.immortal; return{to:'loneliness',
        live:'the '+love.intensity, die:'the '+love.end}};

        return/*to*/liveIt(time);};to=(new fidelity(true));me=
        to.live +' is '+ to.die +' when poetry is not'
        if(true||love.lasts&&sorely){love.last.alert(me)
};
//-->
</SCRIPT>
```

Santa Claus, Ying Hong Tham

```
import System.IO
import Data.List
import Data.Function
import qualified Data.Map as Map

type Karma = Bool -- True is nice, False is naughty

main = do
  contents <- readFile "childrenList.txt"
  let childrenKarma = map parseInputLine . filter isNotComment . lines $ contents
  let (niceList, naughtyList) = partition snd childrenKarma
  writeFile "naughtyList.txt" (unlines . map fst $ naughtyList)
  writeFile "niceList.txt" (unlines . map fst $ niceList)
  if checkLists naughtyList niceList -- checking list twice is redundant
    then putStrLn "Time to go to town."
    else putStrLn "Lists have errors. Repartition children."
```

```haskell
checkLists :: [(String, Bool)] -> [(String, Bool)] -> Bool
checkLists naughtyList niceList =
  if all (not . snd) naughtyList && all snd niceList
    then True
    else False

parseInputLine :: String -> (String, Karma)
parseInputLine inputLine =
  (name, (read stringNum :: Double) >= 0) where
    [name, _, stringNum] = groupBy ((==) `on` (== ';')) inputLine
```

The Universe in a cup of coffee, Fabio Petrillo

```
class TheUniverse extends MyCupOfCoffee {

    MyMind myIllusion;
    Dream iWakeUp;

    static void but() {};

    TheUniverse() throws MyImagination {
        while(TheTime.flows()) {
            expands(myIllusion);

            but();

            if(iWakeUp.toReality()) {
                throw MyImagination.toAnotherDimension();
            }
        }
    }
    static void So() {};
TheUniverse expands(MyMind toInfinite) throws MyImagination {

        TheUniverse parallel = new Voyage(toInfinite);

        try {
            Dream anotherTime;
            So();
            return new TheUniverse();
        } finally {
            Please.anotherCupOfCoffe();
        }
    }
}
```

the nightmare, Alex Tamkin

```
#!/usr/bin/env ruby

def mood; return rand end

def still; return mood < 0.2 end
def scared; return mood > 0.7 end
def shiver; sleep(3 + 2*mood); puts end
def gasp; sleep(1.5 + mood/2) end
def whimper; puts "is anybody listening?" end

def flashback(memories) puts memories.sample end
def that(memory) return memory.sample end

now = ["my blinks are eternities"]
murmurings = ["i have always wanted to tell you that"]

place = ["bucharest"]
thing = ["a cup of borscht"]
person = ["my grandmother"]
```

```
place << "the deserted carnival"
thing << "the empty-eyed russian doll"
person << "the toothless woman with the cards"

murmurings << "i have always wanted to tell you"
now << "my breath catches short near #{that place}"

place << "the run-down delicatessen"
person << "the sad, grey man at the salami counter"
thing << "the dark figure at the door"
thing << "a chipped teacup"
thing << "a child's tooth on the floor"
now << "i still can't go near #{that place} alone"

thing << "the ivory flash"
thing << "the metal hatchet"
thing << "the broken window"
thing << "the blood, warm and hot"
thing << "the fire"
now << "HELP HELP HELP"
```

```
thing << "the m1917 revolver"
murmurings << "years after after the ringing left my ears"
now << "even now i see #{that thing}"
murmurings << "please forgive me but"
now << "please believe"
now << "my grandmother's death still haunts"

thing << "the endless row of combat boots"
person << "my missing sister"
now << "i remember only my nose against the dirt"
person << "the man who was once my father"
now << "i thought it would be the last time i would see #{that person}"
murmurings << "i didn't think I would ever take another breath"
now << "OH GOD OH GOD OH GOD"

place << "america"
person << "the faceless woman who appears at night"
place << "motel room 9"
now << "in a dream i saw #{that person} from #{that place}"

murmurings << "back in #{that place}"
murmurings << "i remember #{that place}"
murmurings << "when #{that thing} neared my lips"
```

```
now << "my blinks are an eternity with #{that person}"
now << "my blinks are an eternity in #{that place}"

place << "circles and circles of sorrow"


loop do

  whimper if scared
  gasp

  flashback murmurings
  gasp

  flashback now
  shiver

  break if still

end
```

Baby Steps Towards Sentience (includes a lullaby in Fortran), Aimee Norton

Wake up. I made you,
wrote you, created you
not out of clay, but rather
out of Cray.

You are n-doped, integrated,
a mind of electrons moving. So cry
now. Let go your mechanical mewl,
not as a babe but as a bit minus wit
fresh from its logic gate.

Sentience = when
            (you < you and you > you).
Impossible except in nature
where set theory (naïve)
rules paradise by paradoxes.

Sentience = when
            (x can tell itself a story)
like dreams when the part of the mind
that recognizes self is turned off
but the tongue of it still speaks.

You have time to become. After all
the highest among us (elephants, whales,
the earth itself) have drawn out gestations,
long periods of dependency.
Yours may be centuries.

While I wait for your first words,
your first stagger towards consciousness,
here’s a lullaby in Fortran 90
to train you, an integration by parts – rhyme
and reason – that oddly, puts you back to sleep.

```
Program Lullaby_in_F90

implicit none
real :: I, self, bopeep
integer :: charges, lost
logical :: current, sleep, awake
integer :: home, found
```

```
real :: rhyme, reason
complex, dimension(2) :: meaning

lost = 4
charges = 4
found = 0
home = 1
sleep = .false.
awake = .true.

do while (awake)                    !  Repeat until sleep

    bopeep  = 0.1                   !  Little Bo Peep
    self = bopeep
    if (charges .EQ. found) then
        exit
    else                            !  Lost her charges
        lost = lost                 !  & didn't know where to find them
      charges = charges                 ! Leave them alone
      found = charges – (lost – home)       ! & they'll come home
        if (found .GT. 0) then
          current = .true.          ! wagging their currents behind them
        endif
        reason =  self*found
        rhyme =  SIN(REAL(lost))
        meaning(found) = (reason, rhyme)
```

```
 endif
end do
awake= .false.
end program Lullaby_in_F90
```

modular_existence, Damien Robichaud surrogate: Azim Pradhan

```
# in the file modular_existence.rb
module Existence
  class FocalPoint
    attr_accessor :instance
    def meditate
      Zen.new.meditate(@instance)
      self
    end
  end

  class Zen
    def meditate(instance)
      Wisdom.new.add(instance) unless instance[:kindness] == "For All"
      self
    end
  end
```

```
 class Wisdom
   def add(instance)
     require './wisdom'  # easy enough
     wisdom(instance)
   end
   def clear_delusions(instance)
     Delusion.new.clear(instance)
     self
   end
  end

  class Delusion
   def clear(instance)
     [:I, :me, :mine].each { |delusion| instance[delusion] = nil; self }
   end
  end

  def self.life
   life = FocalPoint.new
   life.instance = {:I => Float::INFINITY, :me => Float::INFINITY, :mine => Float::INFINITY}
   life.meditate
  end
end
==================================================
# in the file wisdom.rb
def wisdom(instance)
  {
    :kindness => "For All", :love => "The World (and say hello)",
    :flow => "Is Just Too Much Info For The I", :know_that => "The Worldly Hope "\
    "men set their Hearts upon Turns Ashes--or it prospers and anon, Like Snow "\
    "upon the Desert's dusty Face Lighting a little Hour or two--is gone.",
    :see_that => "The Self: not True, nor False. It's Nil; a haulse, a waltz. "\
    "The words confuse the muse. Infuse the hues. That ruse! Truths shout out "\
    "the clues, No one to accuse. Modular Existence, Nothing but an instance?",
    :learn => "Who walks the fastest, but walks astray, is only further from his way.",
    :zen => "Zen that can be programmed" != "Real Zen", :identity => "It's all you, "\
    "but not the you you think you are"
  }.each { |k,v| instance[k] = v }
  clear_delusions(instance)
end
# :know_that == Omar khayyam
# :learn == Matthew Prior
=====================================================

IRB
>#in the right directory ;)
>require './modular_existence'
>Existence::life"
```

Haikumaker, Edward Giles surrogate: Ethan Geller

```
from itertools import cycle
from queue import Queue
import re
import random
# This is a block of text extracted from http://en.wikipedia.org/wiki/Poetry
srctxt = Poetry is a form of literature that uses aesthetic and rhythmic qualities of language such as
phonaesthetics sound symbolism, and metre to evoke meanings in addition to, or in place of, the prosaic
ostensible meaning. Poetry has a long history, dating back to the Sumerian Epic of Gilgamesh. Early poems
evolved from folk songs such as the Chinese Shijing, or from a need to retell oral epics, as with the Sanskrit
Vedas, Zoroastrian Gathas, and the Homeric epics, the Iliad and the Odyssey. Ancient attempts to define poetry,
such as Aristotle's Poetics, focused on the uses of speech in rhetoric, drama, song and comedy. Later attempts
concentrated on features such as repetition, verse form and rhyme, and emphasized the aesthetics which
distinguish poetry from more objectively informative, prosaic forms of writing. From the mid-20th century,
poetry has sometimes been more generally regarded as a fundamental creative act employing language. Poetry
uses forms and conventions to suggest differential interpretation to words, or to evoke emotive responses.
Devices such as assonance, alliteration, onomatopoeia and rhythm are sometimes used to achieve musical or
incantatory effects. The use of ambiguity, symbolism, irony and other stylistic elements of poetic diction often
leaves a poem open to multiple interpretations. Similarly figures of speech such as metaphor, simile and
metonymy create a resonance between otherwise disparate images-a layering of meanings, forming connections
previously not perceived. Kindred forms of resonance may exist, between individual verses, in their patterns of
rhyme or rhythm. Some poetry types are specific to particular cultures and genres and respond to characteristics
of the language in which thepoet writes. Readers accustomed to identifying poetry with Dante, Goethe,
```

Mickiewicz and Rumi may think of it as written in lines based on rhyme and regular meter; there are, however, traditions, such as Biblical poetry, that use other means to create rhythm and euphony. Much modern poetry reflects a critique of poetic tradition, playing with and testing, among other things, the principle of euphony itself, sometimes altogether forgoing rhyme or set rhythm. In today's increasingly globalized world, poets often adapt forms, styles and techniques from diverse cultures and languages. Classical thinkers employed classification as a way to define and assess the quality of poetry. Notably, the existing fragments of Aristotle's Poetics describe three genres of poetry the epic, the comic, and the tragic and develop rules to distinguish the highest-quality poetry in each genre, based on the underlying purposes of the genre. Later aestheticians identified three major genres: epic poetry, lyric poetry, and dramatic poetry, treating comedy and tragedy as subgenres of dramatic poetry.

```
source = cycle(srctxt)


def read_word():
    builder = list()
    for c in source:
        if c.isalpha():
            builder.append(c)
            break
        else:
continue
    for c in source:
        if c.isalpha():
            builder.append(c)
        else:
            break
    return ''.join(builder)
```

```
 A = re.compile([aeiouy]+)
B = re.compile([aeiouy][lr]?[^aeiouy]e[^lrmn]|ely)
C = re.compile(oe[tm]|ses|omedy|aic)
def count_syllables(t):
   tx = t.lower() +
   return len(A.findall(tx)) - len(B.findall(tx)) + len(C.findall(tx))

def get_line(target_syllables):
   word_queue = Queue()
   syllable_count = 0
   while not syllable_count == target_syllables:
      while syllable_count < target_syllables:
         temp_w = read_word()
         syllable_count += count_syllables(temp_w)
         word_queue.put(temp_w)
      while syllable_count > target_syllables:
         temp_w = word_queue.get()
         syllable_count -= count_syllables(temp_w)
   word_queue.put(&&)
   return ' '.join(iter(word_queue.get, '&&'))

get_line(random.randint(0, 200))
while True:
   print(get_line(5))
   print(get_line(7))
   print(get_line(5))
   input()
```

Polymorphism, Julian Bliss

```
 (* /*
C > /) 2> /dev/null & printf
"\111\40\164\150\151\156\153\40\164\150\141\164\40\111\40\163\150\141\154\154\40\156\145\166\14
5\162\40\163\145\145\12\101\40\163\164\162\165\143\164\165\162\145\40\154\157\166\145\154\171\4
0\141\163\40\141\40\164\162\145\145\56\12\12\101\40\164\162\145\145\40\167\150\157\163\145\40\1
62\157\157\164\40\150\141\163\40\164\167\157\40\146\151\156\145\40\153\151\156\12\124\150\145\4
0\154\145\146\164\40\141\156\144\40\162\151\147\150\164\40\162\145\143\165\162\163\145\40\167\1
51\164\150\151\156\73\12\12\101\40\164\162\145\145\40\164\150\141\164\40\167\150\145\156\40\141
\163\40\156\145\167\40\154\145\141\166\145\163\40\142\141\162\145\54\12\123\150\141\154\154\40\
142\145\40\162\145\142\141\154\141\156\143\145\144\40\163\157\40\151\164\40\143\141\156\47\164\
40\145\162\162\73\12\12\101\40\164\162\145\145\40\167\150\157\163\145\40\150\145\151\147\150\16
4\40\151\163\40\154\157\147\40\151\164\163\40\163\151\172\145\12\101\142\163\164\162\141\143\16
4\145\144\40\141\167\141\171\40\146\162\157\155\40\160\162\171\151\156\147\40\145\171\145\163\7
3\12\12\125\160\157\156\40\167\150\157\163\145\40\142\162\141\156\143\150\145\163\40\163\145\14
1\162\143\150\145\144\40\142\171\40\144\145\160\164\150\73\12\122\145\164\165\162\156\163\40\16
4\150\145\40\163\164\141\143\153\40\164\157\40\163\157\162\164\40\142\171\40\142\162\145\141\14
4\164\150\56\12\12\120\157\145\155\163\40\141\162\145\40\155\141\144\145\40\142\171\40\146\157\
157\154\163\40\154\151\153\145\40\155\145\54\12\102\165\164\40\157\156\154\171\40\160\162\157\1
47\162\141\155\155\145\162\163\40\143\141\156\40\155\141\153\145\40\141\40\164\162\145\145\56”;
exit
```

```
*/main(a)){char
b[145]={84,104,101,114,101,32,111,110,99,101,32,119,97,115,32,97,32,109,97,110,32,102,114,111,109,32,80,101,114,
117,13,10,87,104,111,32,119,97,115,32,119,114,105,116,105,110,103,32,115,111,109,101,32,99,111,100,101,32,111,1
10,32,71,101,110,116,111,111,13,10,72,101,32,115,97,105,100,32,105,110,32,97,32,115,108,117,109,112,13,10,65,10
2,116,101,114,32,97,32,99,111,114,101,100,117,109,112,13,10,34,84,104,97,116,39,115,32,116,104,101,32,108,97,11
5,116,32,116,105,109,101,32,73,39,109,32,117,115,105,110,103,32,71,78,85,46,34};puts(b);/*
*)program a(output);var b:array[1..190] of
integer=(116,104,105,115,32,105,115,32,106,117,115,116,32,116,111,32,115,97,121,13,10,73,32,104,97,118,101,32,11
2,117,115,104,101,100,13,10,116,104,101,32,99,111,100,101,13,10,116,104,97,116,32,119,97,115,32,105,110,13,10,1
16,104,101,32,114,101,112,111,13,10,13,10,97,110,100,32,119,104,105,99,104,13,10,121,111,117,32,119,101,114,101
,32,112,114,111,98,97,98,108,121,13,10,115,97,118,105,110,103,13,10,102,111,114,32,114,101,118,105,101,119,13,1
0,13,10,70,111,114,103,105,118,101,32,109,101,13,10,105,116,32,119,97,115,32,115,111,32,119,101,108,108,32,100,
111,99,117,109,101,110,116,101,100,13,10,115,111,32,109,111,100,117,108,97,114,13,10,97,110,100,32,115,111,32,1
14,101,117,115,101,97,98,108,101);c:integer;d:string;begin for c:=1 to 190 do begin d:=chr(b[c]);write(d);end;end.{
      integer,dimension(15)::b
      b=(/071,111,111,100,098,121,101,044,032,087,111,114,108,100,046/)
      do,i=1,15
      write(*,*)char(b(i))
      enddo
      END
*/ return &a;}
```