PISELLO ( Italian for pea…the vegetable)

Goal: create a timer which counts a work period and rest period, giving an alert when period is complete

With a Pomodoro type timer, a person could start stressing, knowing that their time is running out or could start procrastinating towards the end of the timer, reducing their productivity. An approximate timer, would emulate another person telling them to stop at an approximate/slightly unexpected time (more human, less machine-like).

---------

Steps

---------

1.) Set number of peas in a pod be relative to seconds

2.) Set number of pods in a bucket

3.) A random number generator will be called from an array of numbers in a range

a.) Number of peas in a pod will be randomized

b.) 1 tier of randomization only

4.)

5.) When there are no more pods in the bucket, return “STOP”

Learning Goals:

1. All of Week 1 learning goals
2. Week 2 learning goals
   1. Testing
      1. Test driven development- structure with autotest
         1. Too soon to know exactly how we will do this
   2. Ruby Ecosystem
      1. Learn as we go!

Measures of Success:

1. Work stages:
   1. Pisello V.1
      1. Workflow
         1. Get basic app running
         2. Static, timer- no user choice
      2. Teamwork v1
   2. Pisello V.2
      1. Workflow
         1. Allow user custom configuration of timer lengths
         2. Require break time of length x, before the program will restart- remove manual restart timer
      2. Teamwork v2
   3. Pisello V.3
      1. Workflow
         1. Implement audio files for return messages
         2. Implement picture/memes for counter
      2. Teamwork v3?
2. Teamwork
   1. Standups
   2. Reviews of how workflow has worked for each version
      1. Version 1:
         1. One driver- 3 Navigators for Version 1, then reassess how it worked
         2. Research together
            1. 3x5 cards our research
            2. No lone wolf
         3. After success completion of v1. Review what worked and if we want to break out in smaller groups for the next part of the project
         4. Review individual strengths and weaknesses
         5. Review what we think we are (Driver/Navigator)
   3. Structure and follow etiquette for 1 driver 3 navigators.
3. Testing
   1. Each team member comes up with own testing procedure to use.
   2. Use Ruby debugger or Pry
4. Git Repository file sharing and editing
   1. Set up, push, pull, etc.