Crochet Simulator

Have you ever spent hours crocheting the perfect sweater for your 15lb chihuahua only to realize your pattern is meant for an adult German Shepherd?

Alright, that is kind of an unlikely mistake, but correctly fitting any crochet creation poses a major challenge for the everyday crochet person.

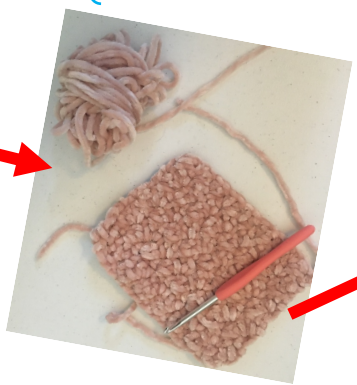
It can be difficult to have a clear idea of what any crochet project will look like when it's done. Crochet projects usually start with a pattern, some yarn and a few hooks. Often the pattern will specify the materials the author used for their project. But what if you want to change the materials or alter the pattern? Without trial and error, how will you know it will fit?

When it comes to fitting your project to a specific subject, the traditional solution is to use a gauge square. A gauge square is a 4x4 inch crocheted square made with your project materials. Essentially, it's a sample of your project. You can count the number of stitches and rows needed to create 4 inches of fabric, and with that information, get a sense of how large your finished product will be.

The problem with the gauge square is that each square only applies to projects where the same stitch, yarn and hook is used. If you want to change any of these three factors, the recommended strategy is to just crochet a new gauge square. The purpose of this project, is to solve the problem of having to create multiple gauge squares by simulating your crochet project.

Click Here for Original Dog Sweater Pattern

Gauge Square to test new velvet yarn & hook size (5mm)



Finished Project!

Ideally, this program would simulate the finished crochet project, by reading entire patterns and factoring in user-defined yarn, stitches and hook sizes.

In the interest of time, my goal for this project is to simulate the gauge square. The user will be able to select stitches, yarn types, and hook sizes to build a sample of their crochet pattern. The program will then return the estimated height, width and weight of the sample gauge square. It will also give feedback to the user if there is an issue with the fit of two consecutive rows. Additionally, it will print out ASCII representations of patterns, rows and stitches.

Preview of Classes

(Please note classes are subject to change)

```
class Pattern:
   '''A Pattern is a  list of  rows'''
   patterns =  []

   def __init__(self, name = None, row):
      self.name = name
      self.list_of_rows = []
      self.list_of_rows.append(row)

   def add_row(self, row, row_num = 1):
      self.list_of_rows.append(row)

   def print_pattern(self):
      pass

   def compare_row_compatibility(self):
      pass

class Row:
   '''A row is a list of stitches'''
   def __init__(self, stitch):
      self.list_of_stitches = []
      self.list_of_stitches.append(stitch)

      self.row_length = self.calc_row_length
      self.row_width = self.calc_row_width
      self.row_weight = self.calc_row_weight
      self.yarn_per_row = self.calc_yarn_per_row

   def add_stitch(self, stitch, repeatitions = 1):
      '''Function adds 1 stitch to row per repeatition'''
      self.list_of_stitches.append(stitch)

   def calc_row_length(self):
      '''returns max stitch_length of list_of_stitches'''
      pass

   def calc_row_width(self):
      '''returns sum stitch_width of list_of_stitches'''
      pass

   def calc_row_weight(self):
      '''returns max stitch_weight of list_of_stitches'''
      pass

   def calc_yarn_per_row(self):
      '''returns sum yarn_per_stitch of list_of_stitches'''
      pass

   def print_row(self):
      for stitch in self.list_of_stitches:
         print(stitch)

   def compare_stitch_compatibility(self):
      pass
```

```
class Stitch:
   '''A stitch is comprised of a stitch type, yarn and hook size'''
   stitches = []

   def __init__(self, name = None, stitch_type = None, yarn_type = None,
hook_size = None):

      Stitch.stitches.append(name)
      self.name = name
      self.stitch_type = stitch_type
      self.yarn_type = yarn_type
      self.hook_size = hook_size
      self.yarn_per_stitch = self.calc_yarn_per_stitch
      self.height = self.calc_height
      self.width =  self.calc_width
      self.weight =  self.calc_weight

   def calc_yarn_per_stitch(self):
      ''Uses stitch_type, yarn_type & hook_size to calc amt of yarn per stitch.'''
      pass

   def calc_height(self):
      '''Uses stitch_type, yarn_type & hook_size to calc stitch height.'''
      pass

   def calc_width(self):
      ''Uses stitch_type, yarn_type and hook_size. to calc width of stitch.'''
      pass

   def calc_weight(self):
      ''Uses stitch_type, yarn_type and hook_size to calc weight of stitch.'''
      pass

   def how_many_stitches_to_reach_x_length(self, length):
      ''Input desired length of piece. Returns # stitches needed achieve length.'''
      pass

   def how_many_rows_to_reach_x_width(self):
      ''Input desired width of piece. Returns # stitches needed achieve width.'''
      pass

   def create_gauge_square(self):
      '''Function will generate a 4x4 gauge square with the stitch.'''
      pass

   def stitch_feedback(self):
      pass

   def __repr__(self):
      txt = self.stitch_type.vis_rep
      return txt

class Stitch_Type:
   def __init__(self, name = None, abrv = None, vis_rep = None)
      self.name = name
      self.abrv = abrv
      self.vis_rep = vis_rep

class Yarn:
   yarns = []
   def __init__(self, weight, yardage, name = "no name", composition =
"unknown")
      self.name = name
      self.weight_of_skien = weight
      self.yardage_of_skien = yardage
      self.composition = composition
      Yarn.yarns.append(self.name)

class Hook:
   def __init__(self, size = 5):
      self.size = size
```