Homework1-sols

January 27, 2017

1 Homework 1

- This homework is a "warm up" - it is NOT graded
- Submit the notebook on courseworks2 before noon Thursday Jan 26
- Your notebook file name must be 'YourUNI.ipynb'

2 Tasks

- Install the Anaconda distribution on your machine
- Try running Spyder
  - double click on YourHomeDir/anaconda/bin/spyder
  - startup tends to be a bit slow
  - enter 'x=5' in the left hand window
  - press the green run button (6th from the left) to load your code into ipython
  - click on the lower right window, type 'x' and return, you should see '5'
- Try running the notebook server
  - in a terminal window, 'cd' to the directory containing this file
  - enter 'jupyter notebook'
  - it should open a window in your browser with the directory's files displayed
  - double click on 'homework-1.ipynb' to open this file
  - click on Help/User Interface Tour
  - click on Help/Keyboard Shortcuts
  - learn how to navigate cells, enter python expressions, and evaluate them
- Look at the problems below
  - Try doing some or all of them in the notebook
  - If you can't do them, try to think about how to approach them
  - I will go over the problems in class

```
In [3]: # you MUST evaluate this cell, or the code below will not work

        # the output of this cell should look something like:

        # '3.5.2 |Anaconda custom (x86_64)| (default, Jul  2 2016, 17:52:12)
        # [GCC 4.2.1 Compatible Apple LLVM 4.2 (clang-425.0.28)]'
```

```
         # if you don't see '3.5.2' and 'anaconda', something
         # went wrong with your install

         import math
         import random
         import sys
         import re

         sys.version

Out[3]: '3.5.2 |Anaconda custom (x86_64)| (default, Jul  2 2016, 17:52:12) \n[GCC 4
```

2.0.1 Hints

- function definition is described in the 03-classes file

```
In [7]: # a useful method on string is isdigit

        s = 'a3'

        [s[0].isdigit(), s[1].isdigit()]

Out[7]: [False, True]
```

```
In [47]: # math functions, pi constant

         [math.sqrt(4), math.sin(math.pi/2), math.cos(math.pi/4)]

Out[47]: [2.0, 1.0, 0.7071067811865476]
```

```
In [9]: # abs value function

        [abs(4), abs(-4)]

Out[9]: [4, 4]
```

```
In [64]: # random.choice randomly picks an element from a list

         clist = range(10)
         for j in range(7):
             print(random.choice(clist))

1
6
7
7
2
0
7
```

```
In [67]: # instead of

         pt = [3,4]

         xcord = pt[0]
         ycord = pt[1]

         # can use destructuring...

         xcord, ycord = pt

         [xcord, ycord]

Out[67]: [3, 4]
```

```
In [7]: # find the digits in a string with a regular expression

        re.findall("[0-9]", "abc3def7xy8z")

Out[7]: ['3', '7', '8']
```

3 circlePoints

- represent a 2D point as a list - [x,y]
- origin is [0,0]
- generates n evenly spaced points on a circle centered at the origin
- points can be viewed as the vertexes of a regular n side polygon(n-gon)
- note that due to floating point rounding, zero is often represented by very small numbers, like 1e-16

```
In [60]: def circlePoints(n, radius):
             ans = []
             for j in range(n):
                 ang = j * 2 * math.pi / n
                 ans.append([radius * math.cos(ang), radius * math.sin(ang)])
             return ans

         # pts are really [[1,0],[0,1],[-1,0],[-1,-1]]
         circlePoints(4,1)

Out[60]: [[1.0, 0.0],
          [6.123233995736766e-17, 1.0],
          [-1.0, 1.2246467991473532e-16],
          [-1.8369701987210297e-16, -1.0]]
```

4 Distance computations

- write function 'euclid' that computes the straight line distance between two points

- write function 'manhattan' that computes the 'manhattan' distance(can only move horizontally or vertically) between two points

```
In [54]: def euclid(pt1, pt2):
             x1,y1 = pt1
             x2,y2 = pt2
             return math.sqrt((x1-x2)**2 + (y1-y2)**2)

         def manhattan(pt1, pt2):
             x1,y1 = pt1
             x2,y2 = pt2
             return abs(x1-x2)+abs(y1-y2)
```

```
In [62]: pt1 = [0,0]
         pt2 = [3,4]

         euclid(pt1, pt2)
```

```
Out[62]: 5.0
```

```
In [63]: manhattan(pt1, pt2)
```

```
Out[63]: 7
```

5 sumPoints

- write a function that computes the x sum and the y sum of a list of points

```
In [16]: def sumPoints(points):
             xsum = 0
             ysum = 0
             # destructure x,y in for loop
             for x,y in points:
                 xsum += x
                 ysum += y
             return [xsum, ysum]
```

```
In [17]: sumPoints([[2,3],[3,4],[10,20]])
```

```
Out[17]: [15, 27]
```

```
In [18]: # try running sumPoints on some n-gons from circlePoints
         # the output is really [0,0], but remember floating point roundoff

         sumPoints(circlePoints(4,1))
```

```
Out[18]: [-1.8369701987210297e-16, 2.220446049250313e-16]
```

```
In [19]: sumPoints(circlePoints(8,1))
```

```
Out[19]: [-4.440892098500626e-16, -2.220446049250313e-16]

In [20]: # it seems like for even n, the points always sum to [0,0]
         # a simple symmetry argument proves this...

         sumPoints(circlePoints(64,1))

Out[20]: [3.3306690738754696e-16, -3.552713678800501e-15]

In [21]: # what about odd n?

         sumPoints(circlePoints(5,1))

Out[21]: [-1.1102230246251565e-16, 1.1102230246251565e-16]

In [22]: sumPoints(circlePoints(17,1))

Out[22]: [-9.992007221626409e-16, 1.0547118733938987e-15]

In [23]: # it may seem surprising that odd n-gons also sum to [0,0]
         # i know a complex way to prove this
         # if you know a simple way - please tell me!!

         sumPoints(circlePoints(63,1))

Out[23]: [-5.10702591327572e-15, 1.5543122344752192e-15]
```

6 Approximate the area and circumference of a circle with n-gons

- write function perimeter, which gives the perimeter of an n-gon
- write function area, which gives the area of n-gon
  - can compute area by dividing n-gon into triangles
  - can find the area of each triangle by getting the base and height
- use euclid in both functions
- as n increases, the n-gon becomes more like a circle
- show that as n increases, area and perimeter values approach circle values

```
In [49]: def perimeter(n, radius):
             cp = circlePoints(n, radius)
             # add the first point at the end
             # so for loop will work
             cp.append(cp[0])
             sum = 0
             for j in range(0, n):
                 sum += euclid(cp[j], cp[j+1])
             return sum

         def area(n, radius):
             # break up n-gon into triangles around the center
             # find base and height of triganle, area = base*height/2
             cp = circlePoints(n, radius)
             cp.append(cp[0])
             area = 0
             for j in range(n):
                 x1,y1 = cp[j]
                 x2,y2 = cp[j+1]
                 xc = (x1+x2)/2
                 yc = (y1+y2)/2
                 height = euclid([xc,yc], [0,0])
                 base = euclid(cp[j], cp[j+1])
                 area += height * base/2
             return area
```

```
In [70]: circlePoints(4,1)

Out[70]: [[1.0, 0.0],
          [6.123233995736766e-17, 1.0],
          [-1.0, 1.2246467991473532e-16],
          [-1.8369701987210297e-16, -1.0]]
```

```
In [68]: area(4,1)

Out[68]: 2.0
```

```
In [69]: perimeter(4,1)

Out[69]: 5.65685424949238
```

```
In [71]: 4*math.sqrt(2)

Out[71]: 5.656854249492381
```

```
In [50]: # area of radius 2 circle

         math.pi * 2**2

Out[50]: 12.566370614359172
```

```
In [72]: def testArea(radius):
             # see how we do for various n-gons
             for n in [3,4,5,8,14,20,30,50,100,1000,1000000]:
                 print(area(n, radius))

         testArea(2)

5.196152422706632
8.0
9.510565162951536
```

```
11.313708498984763
12.148744695291626
12.360679774997898
12.47470144906556
12.533323356430415
12.55810390586267
12.566287931117719
12.566370614345734
```

```
In [56]: # perimeter of radius 1 circle

         2*math.pi

Out[56]: 6.283185307179586
```

```
In [73]: def testPerimeter(radius):
             for n in [3,4,5,8,14,20,30,50,100,1000,1000000]:
                 print(perimeter(n, radius))

         testPerimeter(1)
```

```
5.196152422706632
5.65685424949238
5.877852522924732
6.122934917841436
6.2305861507768014
6.257378601609234
6.27170779605921
6.279051952931337
6.282151815625652
6.2831749717590775
6.283185307177944
```

7 random string generator

- string module - has useful constants
- string doc page

```
In [41]: import string

         def randomString(n):
             # don't want digits in the string
             chars = string.ascii_letters + string.punctuation
             return ''.join([random.choice(chars) for j in range(n)])

         for j in range(4):
             print(randomString(30))
```

```
eagqbjtYPsXTB]]$bTesug{=-,&"Ue
pZZHe"JqwGOYH}e_F&&z$QK;\S{M{G
i$[]lr%_)zCqa'-fXZAIfm+cTC_#jG
GSzrQ;+~s[;`HCRsl?ydfw'h}Vgh~&
```

8 define encrypt and decrypt functions

- encrypt - not so great encryption technique.
  - takes a list of words and encrypts them
  - each word is prefixed by a single digit character count
  - the digits/words are surrounded by random strings
- decrypt by searching for single digits
- warning: these are a tad tricky. don't spend too much time on them

```
In [44]: def encrypt(words):
             rlen = range(5,15)
             # put random string in front of each word
             lenwords = [randomString(random.choice(rlen)) + str(len(w))+w for w i
             # end with random string
             lenwords.append(randomString(random.choice(rlen)))
             return ''.join(lenwords)

         def decrypt(s):
             digits = re.findall("[0-9]", s)
             lastindex = 0
             words = []
             for digit in digits:
                 newindex = s.find(digit, lastindex)
                 newindex += 1
                 wlen = int(digit)
                 lastindex = newindex + wlen
                 words.append(s[newindex:newindex+wlen])
             return words
```

```
In [45]: e = encrypt(['Python', 'is', 'really', 'great!'])
         e
```

```
Out[45]: '-)mxUd/I6Pythone*&P(n:}M|`2is/"VQjea?\'Q%gx6reallyNZAIa6great!KTeDcBc\\Dw
```

```
In [46]: decrypt(e)
```

```
Out[46]: ['Python', 'is', 'really', 'great!']
```

9 is there something odd about the above functions?