13B. Loops and Lists

Topics:
- Functions that return more than 1 thing
- Nested Loops
- Map
Computing the Diameter of a Cloud of Points

500 Points. Which two are furthest apart and what is their separation?
**Same Problem:** What's the Biggest Number in This Table?

Which two cities are furthest apart and what is their separation?

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Amsterdam</td>
<td>Berlin</td>
<td>Bordeaux</td>
<td>Brussels</td>
<td>Copenhagen</td>
<td>Dublin</td>
<td>Lisbon</td>
<td>London</td>
<td>Madrid</td>
<td>Milan</td>
<td>Munich</td>
<td>Paris</td>
<td>Rome</td>
<td>Zurich</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.00000</td>
<td>550.594</td>
<td>1808.367</td>
<td>290.74</td>
<td>765.456</td>
<td>946.404</td>
<td>1254.319</td>
<td>475.011</td>
<td>979.31</td>
<td>1077.74</td>
<td>820.188</td>
<td>503.985</td>
<td>163.55</td>
<td>813.784</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Berlin</td>
<td>0.00000</td>
<td>1634.132</td>
<td>784.376</td>
<td>379.96</td>
<td>1506.491</td>
<td>2604.284</td>
<td>1036.101</td>
<td>2333.429</td>
<td>1033.596</td>
<td>592.956</td>
<td>1053.617</td>
<td>1513.741</td>
<td>644.044</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Bordeaux</td>
<td>1808.367</td>
<td>0.00000</td>
<td>890.136</td>
<td>1785.177</td>
<td>1444.687</td>
<td>1174.092</td>
<td>975.717</td>
<td>733.237</td>
<td>1018.437</td>
<td>1294.774</td>
<td>562.998</td>
<td>1508.036</td>
<td>1021.863</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Brussels</td>
<td>290.74</td>
<td>765.456</td>
<td>0.00000</td>
<td>930.03</td>
<td>775.414</td>
<td>2061.177</td>
<td>306.244</td>
<td>1580.322</td>
<td>881.465</td>
<td>794.539</td>
<td>310.51</td>
<td>1487.05</td>
<td>624.274</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Copenhagen</td>
<td>946.404</td>
<td>1254.319</td>
<td>1506.491</td>
<td>930.03</td>
<td>0.00000</td>
<td>1546.681</td>
<td>2556.016</td>
<td>1177.511</td>
<td>2465.161</td>
<td>1414.722</td>
<td>1080.951</td>
<td>1205.349</td>
<td>2011.726</td>
<td>1185.588</td>
</tr>
<tr>
<td>7</td>
<td>Dublin</td>
<td>1254.319</td>
<td>1785.177</td>
<td>1444.687</td>
<td>775.414</td>
<td>2061.177</td>
<td>0.00000</td>
<td>2609.627</td>
<td>453.606</td>
<td>2138.772</td>
<td>1641.326</td>
<td>1554.939</td>
<td>863.552</td>
<td>2227.14</td>
<td>1383.364</td>
</tr>
<tr>
<td>8</td>
<td>Lisbon</td>
<td>975.717</td>
<td>1174.092</td>
<td>1444.687</td>
<td>2061.177</td>
<td>306.244</td>
<td>1546.681</td>
<td>0.00000</td>
<td>2142.281</td>
<td>626.064</td>
<td>2150.158</td>
<td>2448.688</td>
<td>1749.502</td>
<td>2536.253</td>
<td>2185.763</td>
</tr>
<tr>
<td>11</td>
<td>Milan</td>
<td>2138.772</td>
<td>1641.326</td>
<td>1554.939</td>
<td>863.552</td>
<td>2227.14</td>
<td>1383.364</td>
<td>0.00000</td>
<td>2379.263</td>
<td>0.00000</td>
<td>492.736</td>
<td>847.819</td>
<td>584.634</td>
<td>279.263</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Munich</td>
<td>2138.772</td>
<td>1641.326</td>
<td>1554.939</td>
<td>863.552</td>
<td>2227.14</td>
<td>1383.364</td>
<td>0.00000</td>
<td>2379.263</td>
<td>0.00000</td>
<td>492.736</td>
<td>847.819</td>
<td>584.634</td>
<td>279.263</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Paris</td>
<td>1753.377</td>
<td>400.452</td>
<td>1230.522</td>
<td>848.489</td>
<td>630.414</td>
<td>1418.908</td>
<td>653.606</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>14</td>
<td>Rome</td>
<td>1976.029</td>
<td>1768.102</td>
<td>1950.207</td>
<td>569.94</td>
<td>303.606</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>15</td>
<td>Zurich</td>
<td>2138.521</td>
<td>2189.521</td>
<td>1499.309</td>
<td>653.299</td>
<td>865.466</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
It Will Have Three Functions

MakeCloud\((n, \sigma)\)
This generates two lists \(x\) and \(y\) that define the coordinates of the points in the cloud.

Diameter\((x, y)\)
This will compute the diameter of the cloud using the \((x, y)\) coordinates of its points.

ShowCloud\((x, y)\)
This will use SimpleGraphics to display the cloud and highlight the “diameter points”.
from random import normalvariate as randn

def MakeCloud(n,sigma):
    x=[]
    y=[]
    for k in range(n):
        r = randn(0,sigma)
        x.append(r)
        r = randn(0,sigma)
        y.append(r)
    return (x,y)
from random import normalvariate as randn

def MakeCloud(n,sigma):
    x=[]
    y=[]
    for k in range(n):
        r = randn(0,sigma)
        x.append(r)
        r = randn(0,sigma)
        y.append(r)
    return (x,y)
MakeCloud Returns Two Lists

```python
>>> (x,y) = MakeCloud(3,1)
>>> print x
>>> print y

[-2.328, -0.044, -0.241]
[ 2.737,  2.078, -1.272]

Note the parentheses
from random import normalvariate as randn

def MakeCloud(n, sigma):
    x = []
    y = []
    for k in range(n):
        r = randn(0, sigma)
        x.append(r)
        r = randn(0, sigma)
        y.append(r)
    return x, y

Old Stuff
x and y start out as empty lists.
Repeatedly generate a random number and append to x
Ditto for y
The Diameter Function: What It Computes

The "diameter points" and the distance between them

Input: lists x and y that define the yellow dots
def Diameter(x,y):
    """ Returns (d,imax,jmax) where d is a float that is the diameter of a cloud of points defined by lists x and y. imax and jmax are ints that are the indices of the diameter points.

    The diameter of a cloud of points is the maximum distance between any two points in the cloud. The two points for which this occurs are called diameter points.

    PreC: x and y are lists of floats with the same length."""
Diameter: The Implementation

```python
def Diameter(x,y):
    d = 0
    n = len(x)
    for i in range(n):
        for j in range(n):
            dx = x[i] - x[j]
            dy = y[i] - y[j]
            dij = sqrt(dx**2 + dy**2)
            if dij > d:
                d = dij
                imax = i
                jmax = j
    return (d, imax, jmax)
```
Nested Loops

In this situation we have a loop whose body contains a loop

for blahblahblah

and contains a loop.
Nested Loops: A Simple Example

```python
for i in range(2):
    for j in range(3):
        print i, j
        print 'Inner'
print 'Outer'
```
Nested Loops: A Simple Example

for i in range(2):
    for j in range(3):
        print i, j
    print ‘Inner’

print ‘Outer’

Execute the loop body with i=0
Nested Loops: A Simple Example

```python
for i in range(2):
    for j in range(3):
        print i, j
    print 'Inner'
print 'Outer'
```

Execute the loop body with i=0

```
0 0
0 1
0 2
Inner
```
Nested Loops: A Simple Example

for i in range(2):
    for j in range(3):
        print i,j
    print ‘Inner’
print ‘Outer’

Execute the loop body with i=1
Nested Loops: A Simple Example

for i in range(2):
    for j in range(3):
        print i, j
    print ‘Inner’
print ‘Outer’

Execute the loop body with i=1
Nested Loops: A Simple Example

```python
for i in range(2):
    for j in range(3):
        print i, j
    print 'Inner'
print 'Outer'
```

Go to the next statement after the loop body.
Nested Loops: A Simple Example

```python
for i in range(2):
    for j in range(3):
        print i, j
        print 'Inner'
print 'Outer'
```

Go to the next statement after the loop body.
Back to Diameter

When developing nested-loop solutions, it is essential to apply the methodology of step-wise refinement, perhaps preceded by a small example.

Aspects of our problem

- Must check all possible pairs of points.
- Look at their separation distance.
- What’s the largest among these distances?
Suppose There Are 3 points

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x[0], [y[0])</td>
<td>(x[0], [y[0])</td>
<td>0</td>
</tr>
<tr>
<td>(x[0], [y[0])</td>
<td>(x[1], [y[1])</td>
<td>7</td>
</tr>
<tr>
<td>(x[0], [y[0])</td>
<td>(x[2], [y[2])</td>
<td>9</td>
</tr>
<tr>
<td>(x[1], [y[1])</td>
<td>(x[0], [y[0])</td>
<td>7</td>
</tr>
<tr>
<td>(x[1], [y[1])</td>
<td>(x[1], [y[1])</td>
<td>0</td>
</tr>
<tr>
<td>(x[1], [y[1])</td>
<td>(x[2], [y[2])</td>
<td>10</td>
</tr>
<tr>
<td>(x[2], [y[2])</td>
<td>(x[0], [y[0])</td>
<td>9</td>
</tr>
<tr>
<td>(x[2], [y[2])</td>
<td>(x[1], [y[1])</td>
<td>10</td>
</tr>
<tr>
<td>(x[2], [y[2])</td>
<td>(x[2], [y[2])</td>
<td>0</td>
</tr>
</tbody>
</table>

Number of possibilities.: 9 = 3x3
Suppose There Are 3 points

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x[0], y[0])</td>
<td>(x[0], y[0])</td>
<td>0</td>
</tr>
<tr>
<td>(x[0], y[0])</td>
<td>(x[1], y[1])</td>
<td>7</td>
</tr>
<tr>
<td>(x[0], y[0])</td>
<td>(x[2], y[2])</td>
<td>9</td>
</tr>
<tr>
<td>(x[1], y[1])</td>
<td>(x[0], y[0])</td>
<td>7</td>
</tr>
<tr>
<td>(x[1], y[1])</td>
<td>(x[1], y[1])</td>
<td>0</td>
</tr>
<tr>
<td>(x[1], y[1])</td>
<td>(x[2], y[2])</td>
<td>10</td>
</tr>
<tr>
<td>(x[2], y[2])</td>
<td>(x[0], y[0])</td>
<td>9</td>
</tr>
<tr>
<td>(x[2], y[2])</td>
<td>(x[1], y[1])</td>
<td>10</td>
</tr>
<tr>
<td>(x[2], y[2])</td>
<td>(x[2], y[2])</td>
<td>0</td>
</tr>
</tbody>
</table>

Number of possibilities.: 9 = 3 x 3
And now, stepwise refinement in action....
First Solution

d = 0
n = len(x)
for i in range(n):
    # Examine the distance from
    # (x[i], y[i]) to every other point
d = 0
n = len(x)
for i in range(n):
    for j in range(n):
        # Examine the distance from
        # (x[i],y[i]) to (x[j],y[j])
Third Solution

d = 0
n = len(x)
for i in range(n):
    for j in range(n):
        dx = x[i] - x[j]
        dy = y[i] - y[j]
        dij = sqrt(dx**2 + dy**2)
        # Compare dij to d revising
        # the latter if necessary
d = 0
n = len(x)
for i in range(n):
    for j in range(n):
        dx = x[i] - x[j]
        dy = y[i] - y[j]
        dij = sqrt(dx**2 + dy**2)
        if dij > d:
            d = dij
            imax = i
            jmax = j
return (d, imax, jmax)
Fourth Solution

d = 0
n = len(x)
for i in range(n):
    for j in range(n):
        dx = x[i] - x[j]
        dy = y[i] - y[j]
        dij = sqrt(dx**2 + dy**2)
        if dij > d:
            d = dij
            imax = i
            jmax = j
return (d, imax, jmax)

We have to “remember” where the max separation occurs.
Next Up: ShowCloud
def ShowCloud(x, y):
    """ Displays a point cloud defined by x and y and highlights the two points that define its diameter. """

PreC: x and y are lists of floats with the same length. """
First: How Big a Window?

Idea: look at the x and y coordinates of the points and see how big they can be.

```python
xMax = max(map(abs, x))
yMax = max(map(abs, y))
M = max(xMax, yMax)
MakeWindow(1.1*M, bgcolor=BLACK)
```
Map: Apply a Function to Each Element in a List

Example. Apply the absolute value function to every list element

```python
>>> x = [10, -20, -40]
>>> x = map(abs, x)
>>> print x
[10, 20, 40]
```
Map: Apply a Function to Each Element in a List

Example. Apply the floor function to every list element:

```python
>>> x = [11.3, 12.4, 15.0]
>>> x = map(math.floor, x)
>>> print x
[11.0, 12.0, 15.0]
```
Map: Apply a Function to Each Element in a List

This:

\[
\begin{align*}
y &= [] \\
\text{for } k \text{ in } \text{range}(\text{len}(x)) : \\
&\quad y.\text{append}(\text{math.sqrt}(x([k])))
\end{align*}
\]

Is equivalent to this:

\[
y = \text{map}(\text{math.sqrt}, x)
\]

Assuming that \(x\) is an initialized list of nonnegative numbers.
Map: Formal Syntax

map ( [function], [list] )

The name of a function that returns a value. Every element in the list must satisfy its precondition.

The name of a list.
Now, Back to ShowCloud
First: How Big a Window?

\[
\begin{align*}
x_{\text{Max}} &= \max(\text{map}(\text{abs}, x)) \\
y_{\text{Max}} &= \max(\text{map}(\text{abs}, y)) \\
M &= \max(x_{\text{Max}}, y_{\text{Max}}) \\
\text{MakeWindow}(1.1 \times M, \text{bgcolor}=\text{BLACK})
\end{align*}
\]

\[
x = [-19, 12, -4]
\]

\[
\max(\text{map}(\text{abs}, x))
\]

\[
>>> 19
\]
Next, Use DrawDisk For Each Point

\[
r = M/50;
(d,i,j) = \text{Diameter}(x,y)
\]

for k in range(len(x)):
    if k==i or k==j:
        DrawDisk(x[k],y[k],2*r,FillColor=CYAN)
        DrawDisk(x[k],y[k],r,FillColor=YELLOW)

i and j are the indices of the diameter points.

Before they are displayed, we paint a larger cyan dot.