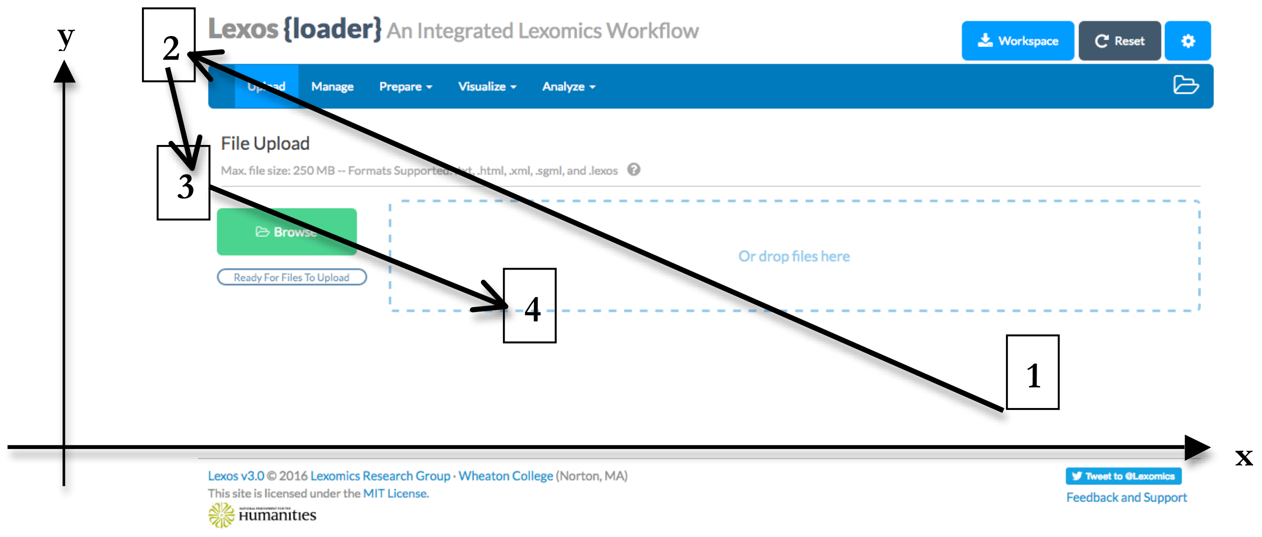
Tracking your Gaze
Programming with eye-tracking data

Due Friday, September 9

Summary

Wheaton recently purchased a set of Tobii Pro eye-tracking glasses[1]. These glasses combine cool with (potential) creep as they (i) take video of wherever you are looking and (ii) save the (x,y) coordinates of *exactly* where you are looking. A number of research areas are applying these glasses, for example: (a) user experience and interaction (tracking where you are looking as you play a video game or use a new app); (b) marketing and consumer research (tracking where a consumer is looking on a shelf while shopping or where they are looking at an advertisement); (c) infant and child research (tracking where early readers are focusing on the page); and (d) psychological and neuroscience (tracking how and why eye movements are made).

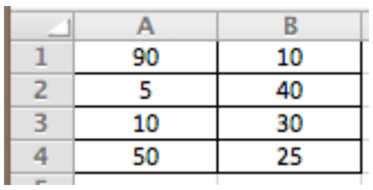
This programming assignment will provide an entry into this exciting space by asking you to work with the first few (x, y) coordinates obtained from your professor as he viewed a new web app. The figure below shows the opening screen to our Lexomics Research Group's recent *v*3.0 release of our *Lexos* software. The numbers and (x, y) coordinates indicate the first four locations on the path where I gazed as recorded by the eye-tracking glasses while viewing the browser window at: http://lexos.wheatoncollege.edu.



Max. file size: 250 MB -- Formats Supported: .txt, .html, .xml, .sgml, and .lexos

Lexos v3.0 © 2016 Lexomics Research Group · Wheaton College (Norton, MA)
This site is licensed under the MIT License.

The eye-tracking glasses export the (x, y) gaze coordinates to a tab-separated-value (.tsv) file that can be opened in Excel. The first four (x, y) coordinates are shown here. Using these four (x,y) coordinates, your task is to write a Python to compute and output metrics about these four gaze points and their associated path.



	A	B
1	90	10
2	5	40
3	10	30
4	50	25

INPUT

The "input" to this program are pairs of (x,y) coordinates, two Real (sometimes called "floating point") values per line.

Notes:

(1) Normally, we'd read these (and many, many more) values from a file, but for now, we'll skip that. We'll of course learn how to read values from files soon.
(2) Rather than force you to enter these values each time at the keyboard (sometimes referred to as the "standard input" or stdin), you may "fake" the input and just assign the (x,y) coordinates to variables, e.g.

[1] Later in the semester, I'd like each of you to try out the eye-tracking glasses and collect your own data. Then your Python programs can process *your own data!* Now that's "programming getting personal" ☺.

```
x1 = 90.0
y1 = 10.0
```
etc.

(3) Of course, you should test your program with other values to make sure your program works no matter the (x, y) coordinates. What possibly could go wrong? If you discover a potential problem, you should caution the user in your input documentation section at the top of your program. Soon, we will be able to handle the situation in our software, but for now, you only have to document any potential input problems.

(4) **When you submit your program, you must use the four (x,y) pairs given on page 1.**

ANALYSIS
Your program must compute the **distance** and **slope** between pairs of the first four gaze points (e.g. 1-to-2, 2-to-3, and 3-to-4). You should also compute the **total** and **average** distance between the fixations.

OUTPUT
The output from this program must be neat and include the following: a title, a neat summary of the four (x,y) points in (x, y) fashion, and good messages that indicate (i) the units of your values, (ii) distance between pairs of the first four gaze points, (iii) the slope between pairs of the first four gaze points, (iv) the total gaze distance, and (v) the average gaze distance. A partial sample output is shown below:

```
==============================================
   GAZE point metrics for four (x,y) points
        (all values in units of pixels)
==============================================

Gaze Fixation (1): ( 90.0, 10.0)
Gaze Fixation (2): (  5.0, 40.0)
Gaze Fixation (3): ( 10.0, 30.0)
Gaze Fixation (4): ( 50.0, 25.0)

Distance from (1) to (2) is: 90.14
  with a slope of: -0.35
          :
Total Gaze Distance from (1) to (4) is:      141.63
Average Gaze Distance between fixations is: 4.72e+01
```

Further Directions
•You must complete the distance and slope (user-defined) functions. I have started these for you in the StarterKit code. Obviously, your main( ) function should call your distance and slope functions (rather than repeat the calculations over and over).

• You must exhaustively test your program using many different (x,y) coordinates. (See the INPUT section above).

• You must use formatted output to ensure you can see levels of precision. Print (x,y) floating point (**f**) coordinates with one place after the decimal point (**%5.1f**), two places after the decimal for the distances and slopes, and use scientific notation (**e**) for the average gaze distance.

```
print ("Gaze Fixation (1): (%5.1f,%5.1f)" % (x1,y1) )
print ("Average Gaze Distance between fixations is: %7.2e" % avgDistance )
```

• Your program must have initial comments, giving your name, the **PURPOSE** or summary of the program (that is, what does it do or why would someone run this app), a description of the **INPUT** the program uses (from *where* and the *data types*) and a description of the program's **OUTPUT** a user should expect if they run your code. I have given you a template to get you started. **See the grade key in the Starter Kit.**

• Name your Python source file with *your Lastname and the assignment#*, e.g., I would name my source file: **LeBlancA1.py**