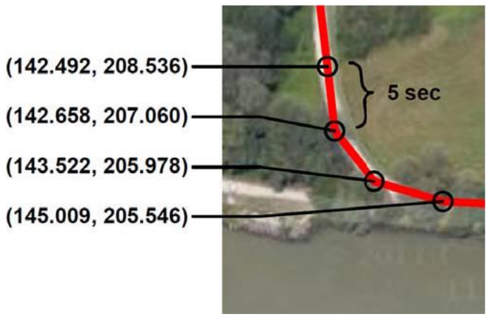
P05 – Lists, Tuples & File I/O

1. Lists & Tuples

1.1 Trail Length

Write a program that computes the total length of a trail tracked with your GPS device.
Let's assume your GPS device saves your coordinates every 5 second into a list of tuples.



For the trail in the right image above, the list of tuples would look like this:
```
trail = [(142.492, 208.536),
         (142.658, 207.060),
         (143.522, 205.978),
         (145.009, 205.546)]
```

At **n** points of time we have recorded the corresponding positions:

$$(x_0, y_0), (x_1, y_1) \dots (x_{n-1}, y_{n-1})$$

The total length **L** of the trail from (x_0, y_0) to (x_{n-1}, y_{n-1}) is the sum of the length all the individu-al line segments:

$$L = \sum_{i=1}^{n-1} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

Create a function **`pathlength(trail)`** for computing **L** according to the formula above. The argument **`trail`** should be a list of **`(x,y)`** coordinate tuples. Test the function on a triangular path with the four points (1, 1), (2, 1), (1, 2), and (1, 1).

Hint: To compute the square root of some value **`x`**, use
```
    import math
    math.sqrt(x)
```

1.2 Largest Product in a grid (optional)

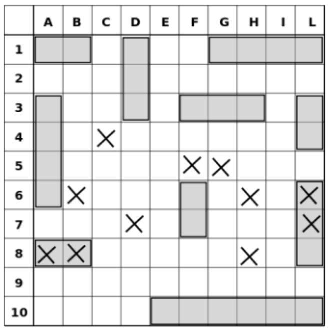
In the 20×20 grid below, four numbers along a diagonal line have been marked in red.

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

The product of these numbers is 26 × 63 × 78 × 14 = 1788696.

What is the largest product of four adjacent numbers in either of the following orientations: horizontally, vertically and diagonally up or down? Write a Python program to find it out (print the product itself and its location within the grid, if you can).

1.3 Battleship[1]



Write a program that is a simplified, one-player version of the classic board game "Battleship". To start, use a 15x10 board, save it as a list of lists and initialize a single boat of size 1x1 in a random location on the board.

Let the player guess at most 10 times the location of the ship (column number and row number). Make sure the guesses of the player are inside the board.

Draw the board in the console after every guess of the player; you could start by printing an '`O`' everywhere (for 'ocean'), then an '**`X`**' for every wrong shot. The game can immediately end when the user hits the ship.

Hint:

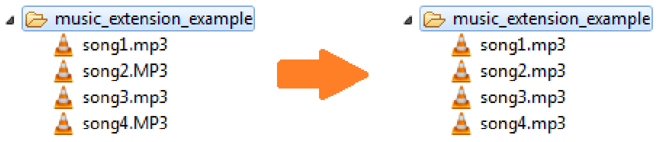
- You can create a list with 10 elements (columns), each containing an `"O"`, with the expression **`row = ["O"] * 10`**
- You can append such a **`row`** to a list called **`board`** with the expression **`board.append(row)`**

[1] Schiffe versenken: http://en.wikipedia.org/wiki/Battleship_(game)

2. File I/O

2.1 Change File Extension

Write a program that unifies the filename endings of MP3 songs (**.mp3** and **.MP3**) for all songs in a given folder to **.mp3**. Download the zip file containing the test data (these are empty files, not actual songs) and extract it to a folder.



music_extension_example
song1.mp3
song2.MP3
song3.mp3
song4.MP3

music_extension_example
song1.mp3
song2.mp3
song3.mp3
song4.mp3

Hints:

- Make sure you import the module **os**: **import os**
- You can get a list of the files in a folder using **os.listdir(<path_to_folder>)**
- You can rename a file using **os.rename(<old_filename>, <new_filename>)**
- Research string methods (**help(str)**), e.g. **replace()**, **endswith()** or **join()**

2.2 Recursive Directory Tree (optional)

Implement a program that asks for a directory and prints all the files in that directory recursively as a tree.

Example output:

```
IT_PROG/
|-- schedule.txt
|-- lectures/
|    |-- V01_Introduction.pdf
|    |-- V02_Variables.pdf
|-- labs/
     |-- P01.pdf
     |-- P02.pdf
```

Hints:

- Use the **os.listdir()** and **os.path.isdir()** functions

➔ In any case, discuss all of your results and findings with your advisor.