# Introduction to Python II: Number of Passengers & Tic-Tac-Toe

Visualize ComplexCity
Chair of Information Architecture
ETH Zürich

March 7, 2014

## 1 Passengers in Public Transport

In this exercise, you will need to write a script, that reads in the CSV file "passenger-counts.csv" and then calculates the average number of passengers from Kalkbreite to Farbhof. You can find the file on the course website:
`http://www.ia.arch.ethz.ch/category/teaching/fs2014-visualize-complexcity/`

## 2 Tic-Tac-Toe

In this exercise you will implement core parts of the game Tic-Tac-Toe. The exercise consists of two parts. First you will need to calculate the field the player clicked on. Second you will program the logic of finding out if there is already a winner.

To get the program running, you need additionally to the standard Python installation als the "**Python Imaging Library**" (PIL). Some versions of Python have the standard user interface library not included. If this is the case in your version, please install also "**Tkinter**". To test if you have the libraries installed and running, open a Python console and run the following commands

```
import PIL
import Tkinter
```

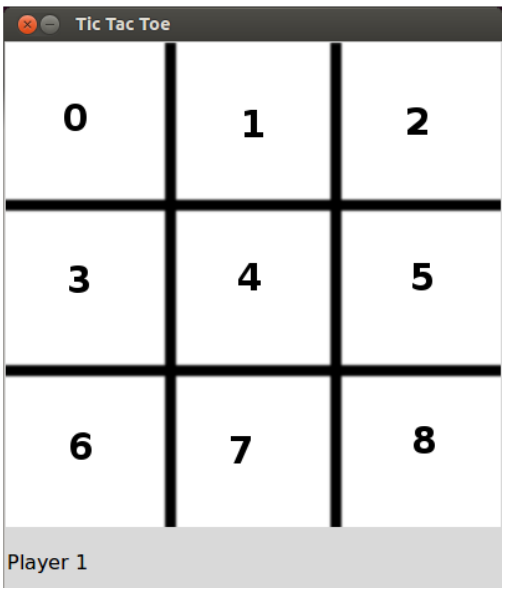
If this runs without any error messages, the libraries are installed and you can start with the exercise.

Please hand in all your Python files in a zip file until the next lecture.

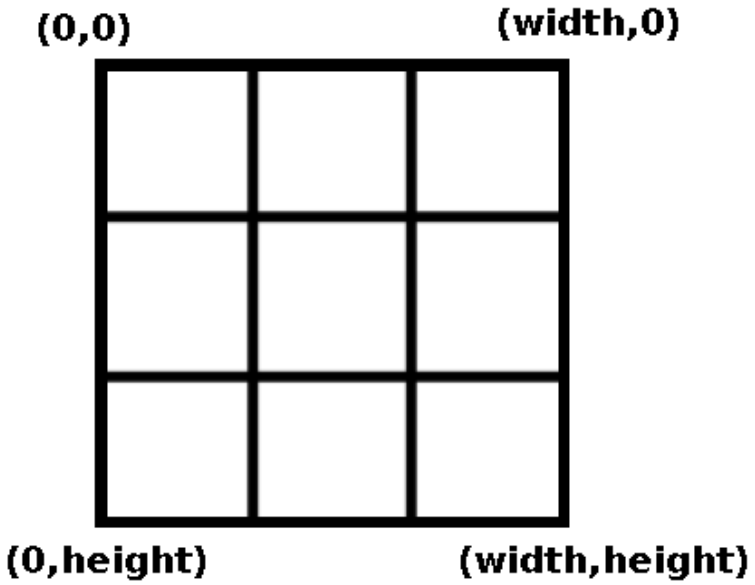
### 2.1 Find the Selected Field

**Function: getIndex**
You have to calculate the index of the corresponding field in the list on which the player clicked. The list you will work with has eight elements and each corresponds to one of the fields. If the player clicked on the top left field, you should return, for example, the value 0. If he clicked on the middle field, you have to return 4.

The first few lines of code in the function gets you the values you will need to find out the value to return. Following a picture showing the index for each field.




The code already given in the function does calculate the size of the playing field (imgWidth, imgHeight) in pixels and the location on the playing field, on which the user clicked (x, y). In the following picture you can see, how the coordinate system is defined in the window.




So the top left corner is $(0, 0)$. The x-value goes from left to right and the y-value from top to bottom. Right now, the function always returns 0, which means, that the rest of the code always assumes that the player clicked on the top left field. This is what you need to change!

## 2.2 Test if there is a Winner

**Function: analyzeFields**
This function should be able to check the *fields*-list for a winner, a draw or if the game continues. It returns

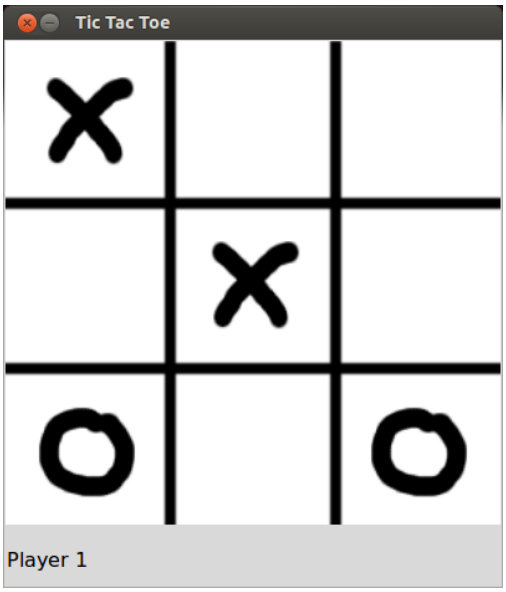
**0:** if nobody has won yet and the playing field is not full.

**1:** if player one has won.

**2:** if player two has won.

**99:** if the playing field is full and nobody has won.

The input of the function is the *fields*-list. It is a list with eight values where every entry corresponds to a field. The first image in the first exercise shows which entry belongs to which index. If player one has already played on a specific field, the list contains a 1 in the corresponding entry. A 2 means that player two played this field and a 0 that the field is still empty. For example in the following case




the list *fields* looks like

```
fields == [1,0,0,0,1,0,2,0,2]
```

In this case, the function *analyzeFields* returns 0.