1) How do I get the information on the website?

In order to be able in reading and dealing with HTML code, I have to import the following library:

```
import urllib2
```

At this point I create a vector, called platforms, containing the names of all the names of the bike docking stations saved as records at the i-line and 2nd column of the shapefile. These names are not saved as they are but they are modified by adding two pieces of HTML code: >name. I made this choice in order to have a "direct link" throughout the HTML code to the section I am interested in.

```
platforms = []
for i in range (len(shapes)):
      platforms.append('>'+str(records[i][2])+'</span>')
```

After creating this vector of modified names, I have to look for each of them into the HTML code:

```
my_number = []
not_reported = []
f = urllib2.urlopen('http://www.tobike.it/frmLeStazioni.aspx')

for line in f:
    for j in range (len(shapes)):
        if re.search(platforms[j], line):
            splitt = line.split('_i')
            Splitt = splitt[1].split('_')
            my_number.append(int(Splitt[0]))
```

As result I get something very similar to the following image. This is an example made considering the first bike docking station, Paravia:

```
<span id="wcPGoogle_radRotStazioni_i0_Laber22" class="Stazione">Paravia</span><br />
<span id="wcPGoogle_radRotStazioni_i0_Label2" class="TableComune">Corso Principe Eugenio ang. c.so Beccaria</span><br />
<span id="wcPGoogle_radRotStazioni_i0_label23" class="Red">5 bici libere<br>6 posti disponibili</span>
```

The example is useful to understand the last three lines of code: it is necessary to move from the name of the bike docking station to the correspondent ID number in order to catch the right data. The following piece of code has been written in order to create a vector of associations between the name of the bike docking station and its ID number:

```
f = urllib2.urlopen('http://www.tobike.it/frmLeStazioni.aspx')
association = []
n = 0
for line in f:
 for i in my_number:
   if re.search('<span id="wcPGoogle_radRotStazioni_i'+str(i)+'_Laber22" class="Stazione">', line):
      couple = []
      lstrip = line.lstrip()
      lsplitted = lstrip.split('</span>')
      lSplitted = lsplitted[0].split('>')
      couple.append(i)
      couple.append(lSplitted[1])
      association.append(couple)
f.close()
```

At this point, I have a vector containing all the bike docking stations and their ID number. This will be useful in order to verify that the data are correctly reported.

```
>>> association
[[0, 'Paravia'], [2, 'Porta Susa 2'], [4, 'Cernaia'], [5, 'Valdocco 2'], [6, 'So
mmeiller'], [7, 'Valdocco'], [8, 'Consolata'], [10, 'Pietro Micca'], [11, 'Corte
 d`Appello'], [13, 'Statuto 2'], [14, 'XI Febbraio'], [15, 'Cimitero Monumentale
```

Now I skip to the line in which all the data I'm looking for are contained. To do this I use the same idea as before: I analyze the HTML code in order to find a string of code that is repeated in the same line of the information that I need. Notice that, in order to perform this search I imported the re library.

```
f = urllib2.urlopen('http://www.tobike.it/frmLeStazioni.aspx')
lines = []
for line in f:
 for i in my_number:
   if re.search('<span id="wcPGoogle_radRotStazioni_i'+str(i)+'_label23" class="Red">', line):
      lstrip = line.lstrip()
      lsplitted = lstrip.split('<br>')
      lines.append(lsplitted)
f.close()
```

After performing these few lines of code I got all the information I wanted. The only problem is to select the numbers from the line and to correctly associate them to the correspondent bike docking station. I report a small piece of program that is useful to understand the way I used to select the numbers: I simply used the command split repeated more times.

```
for i in range(len(lines)):
 last_to_add = lines[i][len(lines[i])-1].split('<')
 last = lines[i][len(lines[i])-1]
 lines[i].remove(last)
 lines[i].append(last_to_add[0])

for i in range(len(lines)):
 first_to_add = lines[i][0].split('>')
 first = lines[i][0]
 lines[i].reverse()
 lines[i].remove(first)
 lines[i].append(first_to_add[1])
 lines[i].reverse()
```

The data linked to Paravia station is composed by the first group of three numbers:

```
>>> numbers
[6, 5, 0, 0, 14, 0, 3, 12, 0, 2, 10, 0, 7, 5, 2, 6, 6, 2, 6, 9, 0, 10,
```

There are same particular cases in which there are some bikes to be repaired, so I add a third number which is equal to zero if there are not broken bikes.

```
<span id="wcPGoogle_radRotStazioni_i0_Laber22" class="Stazione">Paravia</span><br />
<span id="wcPGoogle_radRotStazioni_i0_Label2" class="TableComune">Corso Principe Eugenio ang. c.so Beccaria</span><br />
<span id="wcPGoogle_radRotStazioni_i0_label23" class="Red">5 bici libere<br>6 posti disponibili</span>
```

2) How is possible to link NetLogo and GIS?

Before introducing my own experience and my reasoning, I want to report the website I referred to in order to learn the notions I used in my simulation: http://ccl.northwestern.edu/netlogo/docs/gis.html . Thanks!

At the beginning of my program you can find `extensions [gis]`, this command let you import the GIS extension, or library. "In general, you first define a transformation between GIS data space and NetLogo space, then load datasets and perform various operations on them. The easiest way to define a

transformation between GIS space and NetLogo space is to take the union of the envelopes or bounding rectangles of all of your datasets in GIS space and map that directly to the bounds of the NetLogo world." And this is practically what I did in my simulation to deal with GIS data and envelopes:

```
set torino gis:load-dataset "Turin_map.shp"
set postazioni gis:load-dataset "TO_Bike_actual_situation.shp"
```

With these two commands I create two variables – torino and postazioni – made up of two different dataset, the former contains the data of the map of the city, the latter contains the data on the bike docking stations. Each dataset can be considered as a polygon made up of several vertexes, and each one can have several attributes. I report a piece of TO_Bike_actual_situation.shp to better understand what i said.

	A	B	C	E	F	G
1	ID_STRING,C,10	ID_BIKE_SH,N,9,0	DENOMINAZI,C,254	AVAILABLE_BN,N,9,0	AVAILABLE_PN,N,9,0	BROKEN_BIKEN,N,9,0
2	001	1	Paravia	6	5	0
3	003	2	Porta Susa 2	0	14	0
4	005	3	Cernaia	3	12	0
5	006	4	Valdocco 2	2	10	0
6	007	5	Sommeiller	7	5	2
7	008	6	Valdocco	6	6	2
8	009	7	Consolata	6	9	0
9	011	62	Pietro Micca	10	0	1
10	012	8	Corte d'Appello	4	8	0
11	014	9	Statuto 2	3	10	1

I do not want to analyze the code in detail, as this work has already been done, I only focus the attention on the commands which are strictly connected to GIS extension.

```
foreach gis:feature-list-of postazioni [let location gis:location-of (first (first (gis:vertex-lists-of ?)))
```

With this line of code I enter in the dataset of postazioni, considering each single item of the list and I create a temporary variable, called location, in which the coordinates of each vertex are stored, this is done in order to create an agent at the very same position of the vertex.

In order to read the attributes that a vertex has, I use the command `gis:property-value ?` as follows:

```
set bikes_avail_num gis:property-value ? "available_b"
```

To draw the map of the city, and at the same time let the user understand that it must be considered as a background for the simulation, I draw it setting a light grey as colour, while the set of bike docking stations is characterized by a small red point. The code is:

```
gis:set-drawing-color (gray + 1)  gis:draw torino 1
gis:set-drawing-color red gis:draw postazioni 1
```

Once everything has been created and drawn, I decided to manage my world in order to have the dataset called postazioni has main core of the simulation. I performed this will thanks to:

```
gis:set-world-envelope gis:envelope-of postazioni
```

by which I set the envelope of my world with the same envelope of the dataset.

The last command of GIS extension I used in the simulation allows me to compare two different dataset by means of an overlap. I got this result in this line of code:

```
ask patches gis:intersecting torino
     [set street? false]
```

3) Compared to the paper

What I really found interesting of the paper “LARGE-SCALE AGENT-BASED TRAVEL DEMAND OPTIMIZATION APPLIED TO SWITZERLAND, INCLUDING MODE CHOICE”, written by several researchers of ETH Zurich and Axon Active AG Lucerne, is the idea of linking transportation and viability in general, to a utility function defined in order to give a cost to the path and the typology of means of transport chosen by each agent.

The path is computed using a genetic algorithm that allows the agent to choose between several opportunities in a rational way, while in my simulation only half of the choices are taken considering an oriented direction. So, while my simulation is built considering the direction that each agent has to set to reach the objective, the paper’ simulation uses a different strategy: it chooses among all the possible paths the one that satisfies the genetic algorithm and performs the best outcomes. To do this it has to consider a lattice, made up of more than 400.000 nodes.

My simulation considers only bikes and bike docking stations, while in the paper are considered four different means of transport: car, public transit, bikes and people walking. The goals of the two simulations are really different, I just want to implement a simulation in order to verify if the [TO] Bike is structured in a well working way. The paper’s aim is to replan the whole setting of the viability to optimize it.