Announcements

- Course website: datascienceforall.org
- What if you just added?
- What if you get a 404 Error trying to access a notebook?
- Reminder:
  - Piazza is your first point of contact for questions other people might also have
  - Your section TA is your first point of contact for questions about your personal logistics
Arrays and Ranges

(Leftover from Lecture 04)
Ways to create a table

- `Table.read_table(filename)` - reads a table from a spreadsheet
- `Table()` - an empty table
- and...
Arrays → Tables

● `Table().with_column(label, data) - creates a table with a single column; data is an array`

● `Table().with_columns(label1, data1, ...) - creates a table, with an array of data for each column`
Table Methods

- Creating and extending tables:
  - `Table().with_columns` and `Table.read_table`
- Finding the size: `num_rows` and `num_columns`
- Referring to columns: labels, relabeling, and indices
  - `labels` and `relabeled`; column indices start at 0
- Accessing data in a column
  - `column` takes a label or index and returns an array
- Using array methods to work with data in columns
  - `item`, `sum`, `min`, `max`, and so on
- Creating new tables containing some of the original columns:
  - `select`, `drop`
Examples

The table students has columns Name, ID, and Score. Write one line of code that evaluates to:

a) A table consisting of only the column labeled Name
   
   students.select('Name')

b) The largest score
   
   students.column('Score').max()
   max(students.column('Score'))
Rows
Take Rows, Select Columns

The **select** method returns a table with only some columns.

The **take** method returns a table with only some rows:

- Rows are numbered, starting at 0.
- Taking a single number returns a one-row table.
- Taking a list of numbers returns a table as well.

(Demo)
The where method

- \texttt{t.where(label, condition)} - constructs a new table with just the rows that match the condition

(Demo)
Manipulating Rows

- `t.sort(column)` sorts the rows in increasing order
- `t.take(row_numbers)` keeps the numbered rows
  - Each row has an index, starting at 0
- `t.where(column, are.condition)` keeps all rows for which a column's value satisfies a condition
- `t.where(column, value)` keeps all rows containing a certain value in a column
Minard's Map
Charles Joseph Minard, 1781-1870

- French civil engineer who created one of the greatest graphs of all time
- Visualized Napoleon's 1812 invasion of Russia, including
  - the number of soldiers
  - the direction of the march
  - the latitude and longitude of each city
  - the temperature on the return journey
  - Dates in November and December
Visualization of 1812 March

**FIGURATIVE MAP** of the successive losses in men of the French Army in the RUSSIAN CAMPAIGN OF 1812-1813

Drawn by Mr. Minard, Inspector General of Bridges and Roads in retirement, Paris, 20 November 1869. The numbers of men present are represented by the widths of the colored zones in a ratio of one millimeter for six thousand men; these are also written beside the zones. Red designates men moving into Russia; black those on retreat. — The informations used for drawing the map were taken from the works of M. Hiers, de Ségur, de Vexinac, de Chambry and the unpublished diary of Jacob, pharmacist of the Army since 28 October. In order to facilitate the judgement of the eye regarding the diminution of the army, I suppose that the troops under Prevost, Jerome and under Marshal Duroy, who were sent to Mosk and Mohilow and who rejoined near Orscha and Witebsk, had always marched with the army.

**GRAPHIC TABLE of the temperature in degrees of Réaumur thermometer**

The Cossovo pass the frozen Niemen at a gallop

-26° December 7
-30° December 7
-24 December 1
-20 November 28
-11°
-9° November 9
-22° November 14
-20° November 28

Rain October 24

-10 -13 -16 -19 -22 -25 -28 -31
Different types of data

<table>
<thead>
<tr>
<th>Longitude</th>
<th>Latitude</th>
<th>City</th>
<th>Direction</th>
<th>Survivors</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>54.8</td>
<td>Smolensk</td>
<td>Advance</td>
<td>145000</td>
</tr>
<tr>
<td>33.2</td>
<td>54.9</td>
<td>Dorogobouge</td>
<td>Advance</td>
<td>140000</td>
</tr>
<tr>
<td>34.4</td>
<td>55.5</td>
<td>Chjat</td>
<td>Advance</td>
<td>127100</td>
</tr>
<tr>
<td>37.6</td>
<td>55.8</td>
<td>Moscou</td>
<td>Advance</td>
<td>100000</td>
</tr>
<tr>
<td>34.3</td>
<td>55.2</td>
<td>Wixma</td>
<td>Retreat</td>
<td>55000</td>
</tr>
<tr>
<td>32</td>
<td>54.6</td>
<td>Smolensk</td>
<td>Retreat</td>
<td>24000</td>
</tr>
<tr>
<td>30.4</td>
<td>54.4</td>
<td>Orscha</td>
<td>Retreat</td>
<td>20000</td>
</tr>
<tr>
<td>26.8</td>
<td>54.3</td>
<td>Moiodexno</td>
<td>Retreat</td>
<td>12000</td>
</tr>
</tbody>
</table>

**float:** decimal number

**string:** text

**int:** integer
Lists
Lists are Generic Sequences

A list is a sequence of values (just like an array), but the values can all have different types

```
[2+3, 'four', Table().with_column('K', [3, 4])]
```

If you create a table column from a list, it will be converted to an array automatically

(Demo)