In [8]:
```
import math
```

Economics and Python

Philosophy

``Mathematics is the art of giving the same name to different things.''

Henri Poincare

Comments:

That is what economics and statistics and Python are also about

Both reason in terms of abstract concepts and acquire power by applying those concepts to superficially different situations

Instantiation

Economics

1. Theory
* Data

``If you ask the data to speak for themselves, they will be silent.''

Anonymous econometrician

Theory

1. General equilibrium

    * preferences
    * technologies
    * information
    * equilibrium concept

* Game

    * list of players
    * actions available to all players
    * payoffs of all players as functions of actions of all other players
    * Nash equilibrium

Insight about economic theory

Particular economic models are typically **instances** of either

  * a **general equilibrium** or
  * a **game**

that are specified by filling
in particular versions of the items that constitute them.

Their common structure empowers the analyst to construct and understand them

Data

  1. Data set

     * Time series
     * Cross sections
     * Panels

  * Statistical model

     * A joint probability distribution indexed by vector of parameters

     * Data reduction and summary

         * sufficient statistic

           * all information about parameter contained in sample

Marriage between theory and data

    * Likelihood function

    * Prior distribution over parameters

Python

Abstractions

 1. $\ldots$
 * $\ldots$
 * $\ldots$

Instances

 1. $\ldots$

* $\ldots$
* $\ldots$

Operators

1. Function

   * one and only one task

* Class

    * multiple tasks

    * a function inside a Class is called a method

Abstractions

1. string
* list
* tuple
* dictionary
* array
* $\ldots$

Instances

1. "Henri Poincare is correct"
* $[1, 2, round(math.pi), round(math.e)]$
* (1, 3, 3, 5, 10)
* $\ldots$

In [15]:
```
 string = "Henri Poincare is correct"
 x = [1, 2, round(math.pi), round(math.e)]
 y = (1, 2, 3, 5, 10)

 print (x)
print (y)
print (string)
```

```
[1, 2, 3, 3]
(1, 2, 3, 5, 10)
Henri Poincare is correct
```

In [ ]:
```

```