Announcements
Review: Arrays

An array contains a sequence of values

- All elements of an array should have the same type
- Arithmetic is applied to each element individually
- When two arrays are added, they must have the same size; corresponding elements are added in the result
- A column of a table is an array
Ranges
Ranges

A range is an array of consecutive numbers

- `np.arange(end)`: An array of increasing integers from 0 up to `end`
- `np.arange(start, end)`: An array of increasing integers from `start` up to `end`
- `np.arange(start, end, step)`: A range with `step` between consecutive values

The range always includes `start` but excludes `end`
Tables
Ways to create a table

- `Table.read_table(filename)` - reads a table from a spreadsheet
- `Table()` - an empty table
- and... `select`, `where`, `sort` and so on all create new tables
Example
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
Some of Minard’s 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>
Table Methods

- Creating and extending tables:
  - `Table().with_column` 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`

(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
Break
Table Review
# Table Structure

- A Table is a sequence of labeled columns
- Labels are strings
- Columns are arrays, all with the same length

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Area (m²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>California</td>
<td>CA</td>
<td>163696</td>
</tr>
<tr>
<td>Nevada</td>
<td>NV</td>
<td>110567</td>
</tr>
</tbody>
</table>
Table Methods

- Creating and extending tables:
  - `Table().with_column` 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`
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 for which a column's value equals some particular value
- `t.with_row` makes a new table that has another row
Lists
A list is a sequence of values (just like an array), but the values can all have different types:

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

- Lists can be used to create table rows.
- If you create a table column from a list, it will be converted to an array automatically.

(Demo)
Discussion Questions

The table `nba` has columns `NAME`, `POSITION`, and `SALARY`.

a) Create an array containing the names of all point guards (PG) who make more than $15M/year

\[
\text{nba.where}(1, 'PG').\text{where}(2, \text{are.} \text{above}(15)).\text{column}(0)
\]

b) After evaluating these two expressions in order, what's the result of the second one?

\[
\text{nba.with_row}([\text{'Samosa'}, \text{'Mascot'}, 100])
\text{nba.where}(\text{'NAME'}, \text{are.} \text{containing}(\text{'Samo'}))
\]
Census Data
The Decennial Census

● Every ten years, the Census Bureau counts how many people there are in the U.S.

● In between censuses, the Bureau estimates how many people there are each year.

● Article 1, Section 2 of the Constitution:
  ○ “Representatives and direct Taxes shall be apportioned among the several States … according to their respective Numbers …”
Analyzing Census Data

Leads to the discovery of interesting features and trends in the population

(Demo)
Census Table Description

- Values have column-dependent interpretations
  - The SEX column: 1 is *Male*, 2 is *Female*
  - The POPESTIMATE2010 column: 7/1/2010 estimate
- In this table, some rows are sums of other rows
  - The SEX column: 0 is *Total* (of *Male* + *Female*)
  - The AGE column: 999 is *Total* of all ages
- Numeric codes are often used for storage efficiency
- Values in a column have the same type, but are not necessarily comparable (*AGE* 12 vs *AGE* 999)

Growth Rate

- Growth rate = $g$ (for example 3%, or 0.03)
- Initial value $x$, final value $y$ after $t$ periods of time

Value after 1 period  = $x + xg = x \cdot (1+g)$
Value after 2 periods = $x(1+g)(1+g) = x \cdot (1+g)^2$
Value after $t$ periods = $y = x \cdot (1+g)^t$

So $(1+g)^t = \frac{y}{x}$ and so $1+g = \left(\frac{y}{x}\right)^{\frac{1}{t}}$

So $g = \left(\frac{y}{x}\right)^{\frac{1}{t}} - 1$