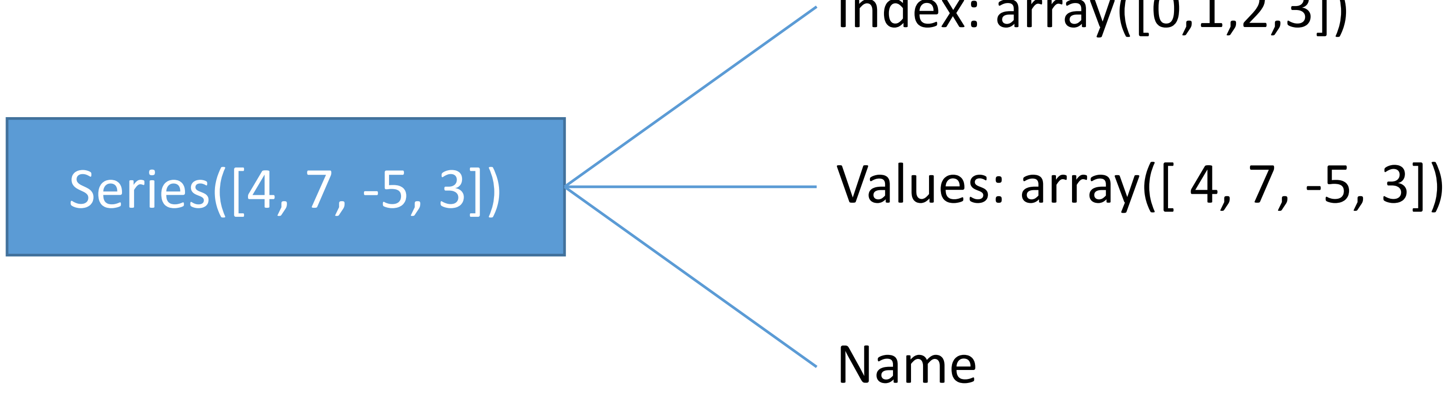
Data Wrangling

Agenda

- **Pandas**
- Web Scraping
  - CSS
  - XPath
  - Hacker News
- Merging Data
- Pandas Input / Output

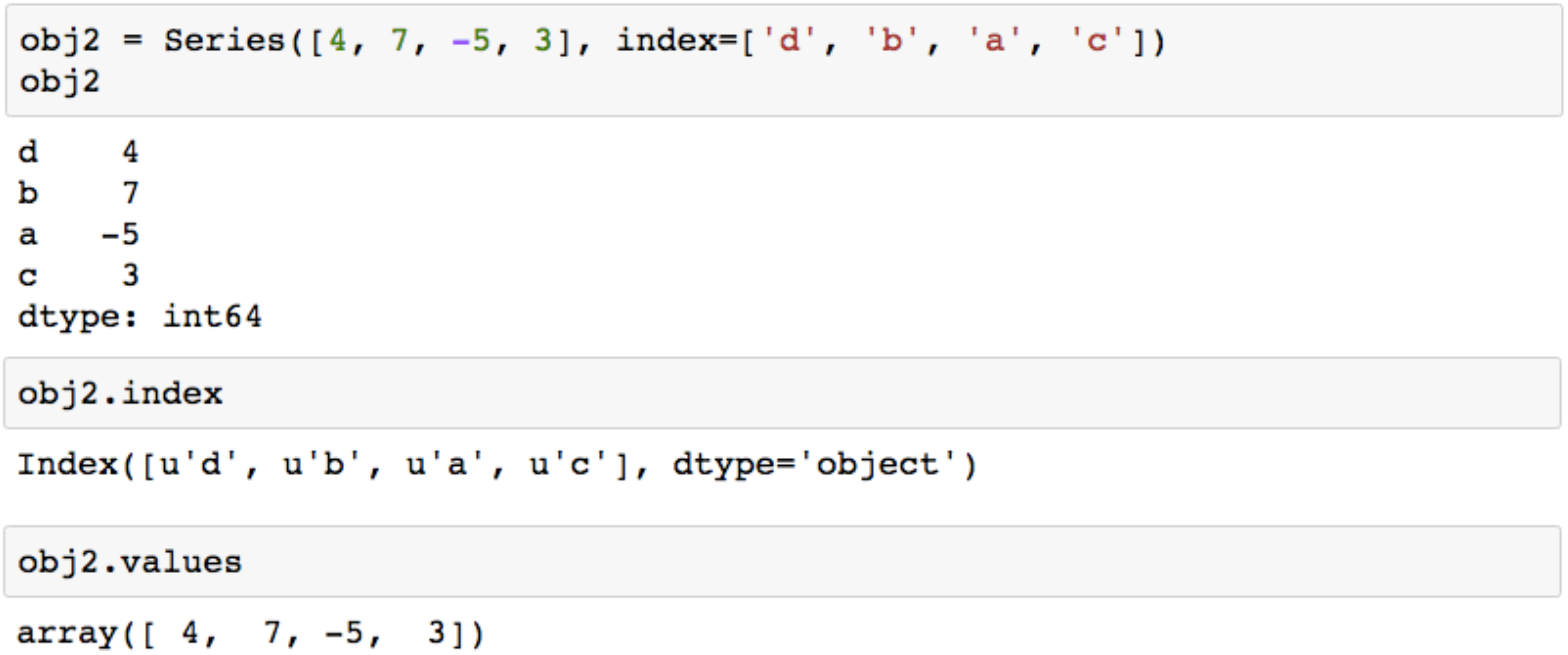
Pandas

Series : pandas 1-D vectors



Series: Index, Values

2 main Series attribues: Index, Values



```
obj2 = Series([4, 7, -5, 3], index=['d', 'b', 'a', 'c'])
obj2
```

```
d    4
b    7
a   -5
c    3
dtype: int64
```

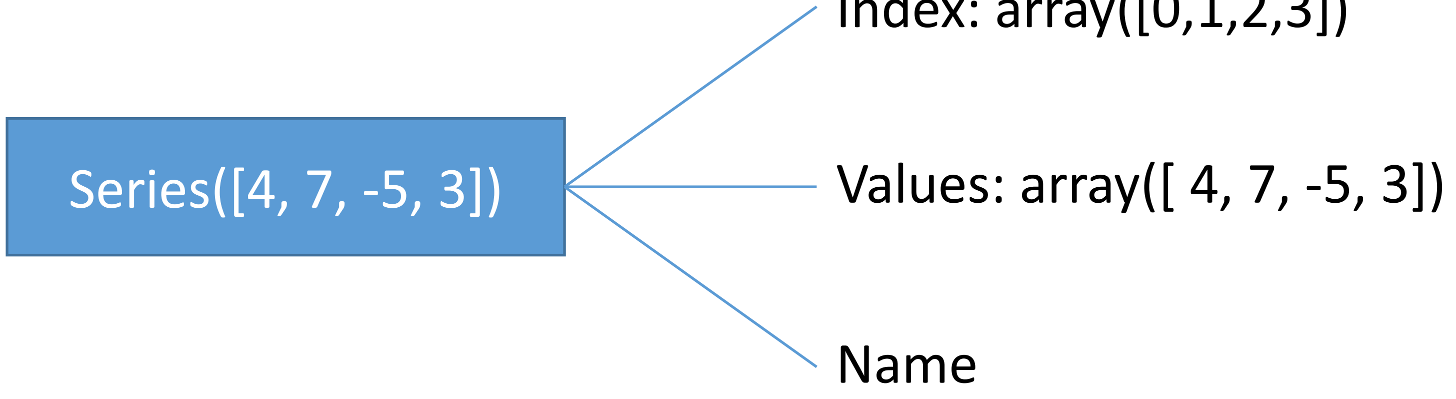
```
obj2.index
```

```
Index([u'd', u'b', u'a', u'c'], dtype='object')
```

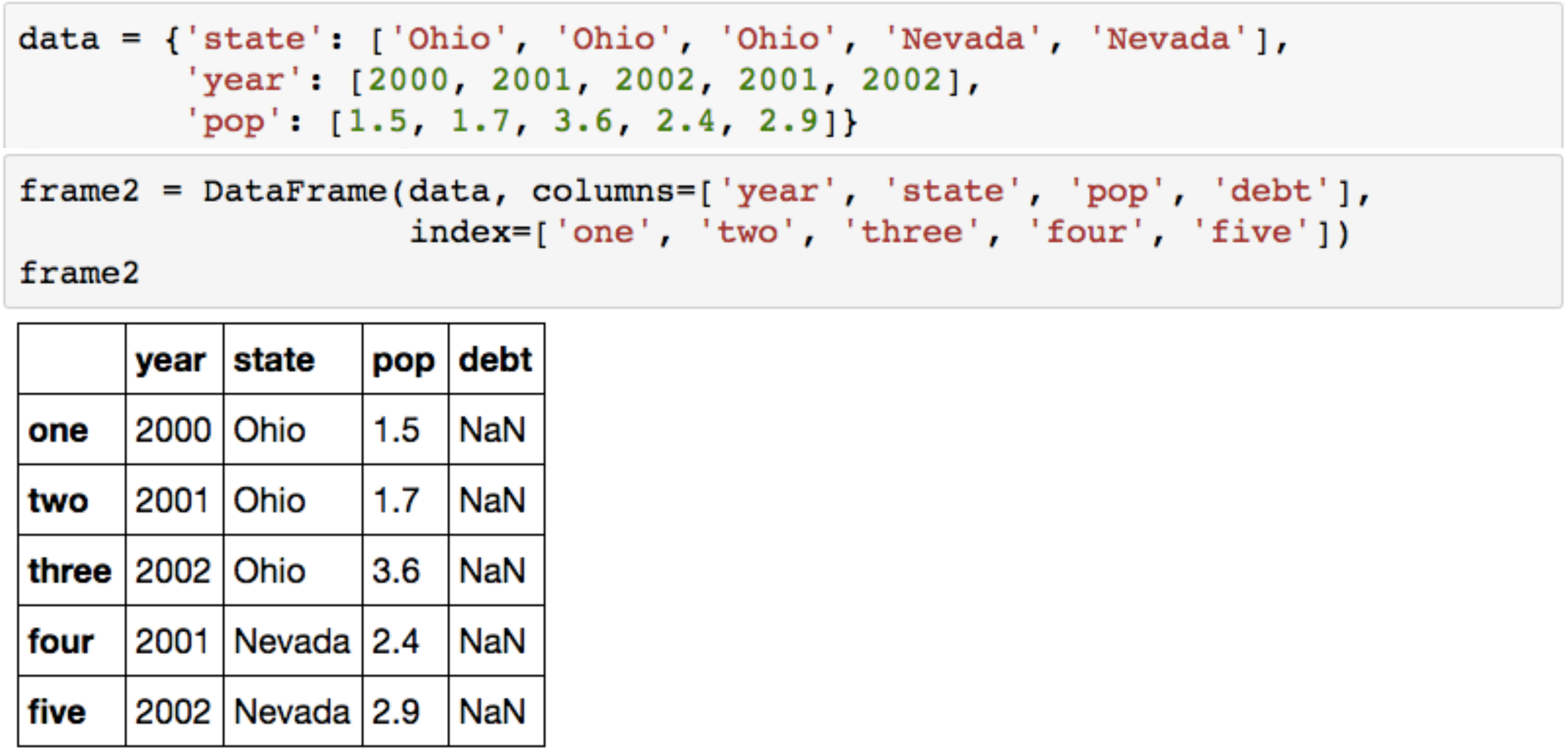
```
obj2.values
```

```
array([ 4,  7, -5,  3])
```

DataFrame: pandas > 1-D vectors



DataFrame: columns of lists with indices



```
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
        'year': [2000, 2001, 2002, 2001, 2002],
        'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
```

```
frame2 = DataFrame(data, columns=['year', 'state', 'pop', 'debt'],
                   index=['one', 'two', 'three', 'four', 'five'])
frame2
```

	year	state	pop	debt
one	2000	Ohio	1.5	NaN
two	2001	Ohio	1.7	NaN
three	2002	Ohio	3.6	NaN
four	2001	Nevada	2.4	NaN
five	2002	Nevada	2.9	NaN

Agenda

- Pandas
- **Web Scraping**
  - **CSS**
  - **XPath**
  - **Hacker News**
- Merging Data
- Pandas Input / Output

Web Scraping

lecture04.web.scraping.ipynb

CSS

What is CSS?

- CSS stands for Cascading Style Sheets
- Styles define how XHTML elements and markup should be displayed by the browser (or user agent)
- Styles can be included in the <head> area of an XHTML document, or placed in external Style Sheet files.
- Multiple style definitions are able to cascade into one…

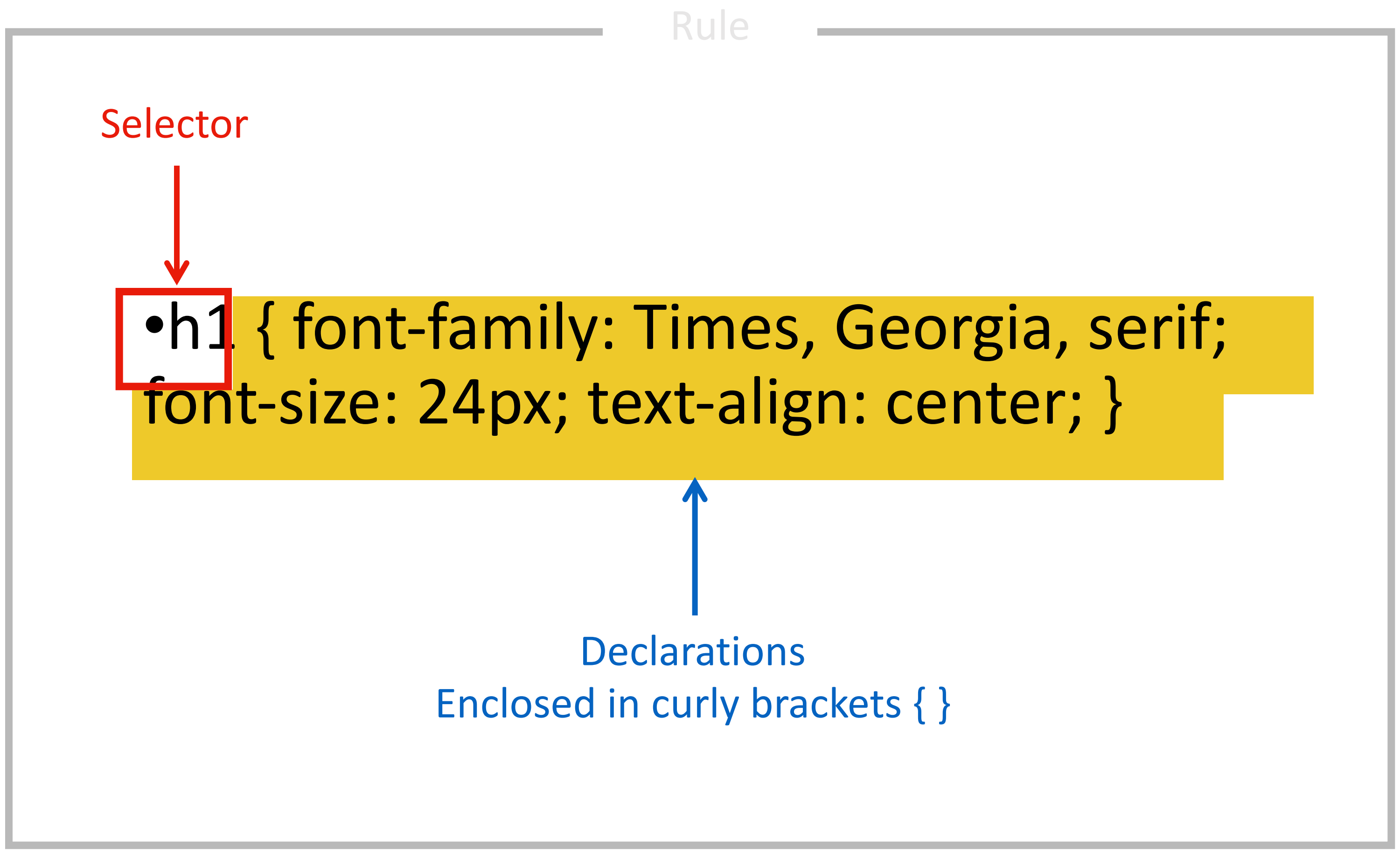
Rules in CSS

- A CSS Style Sheet is basically a collection of rules, describing how the browser should display XHTML elements.
- Each rule contains 2 parts:
  - A **Selector**, stating which element in the XHTML a rule applies to;
  - One or more **Declarations**, which describe how these elements should be displayed.

CSS : Example

```
<html>
<head>
<title>Example page with embedded style</title>
<style type="text/css">
  body    { font-family : sans-serif;
            color : blue;
            background-color : yellow }
  h1      { font-style : italic }
  p       { font-size : 14pt }
  ol      { font-size : 12pt;
            color : red;
            font-family : serif }
</style>
</head>
  ...
</html>
```

CSS Rules: Selectors and Declarations



```
•h1 { font-family: Times, Georgia, serif; font-size: 24px; text-align: center; }
```

CSS Rules: Declaration Parts

Only values with a space in the name need to be enclosed in quotes

- h1 {
  font-family: “Trebuchet MS”, serif;
  font-size: 24px;
  text-align: center;
- }

<u>Note: values</u> do not have to be enclosed in quotation marks, unless the value name includes a space (e.g multi-word name).

CSS Simple, or Element Selectors

- The most basic form of CSS selector is an XHTML element name;

  h1, h2, p, ol, ul, table, etc.

- This is the simple or element selector. Example:

  ```
  p { color: #003366; }
  ```

- This will set every occurrence of content marked up the <p> paragraph tag to be a dark blue colour.

CSS Class Selectors

- However, in XHTML markup, elements can have <u>class</u> attributes assigned to them. The <u>value</u> attached to a class attribute can be one (or more) names, separated by spaces. Example:

  <h1 class=“special”> or

  <h1 class=“special underlined”>

- The actual definition of the value “special” is defined in a CSS class selector…

CSS Class Selectors

```
h1.special { color: #FF0000; }
```

- This will now make all <h1> elements with the class "special" display text with a red colour. <h1> elements that don't have the class attribute "special" will continue to display in the default <h1> colour.
- A CSS class selector can also be defined more generally, without the element name (just the dot):

  ```
  .special { color: #FF0000; }
  ```

- This class can now be applied to any XHTML element that has the class attribute "special".
- *Actually the full CSS syntax is* *.special, *where* * *is a selector that matches anything. However, CSS shorthand means we can drop the* *.

CSS ID Selectors

- XHTML elements can also have id selectors assigned to them. Example:

  `<p id=“summary”>blah, blah, blah.</p>`

- In the CSS, the id “summary” can be styled in a rule, thus:

  `#summary { font-style: italic; }`

- Whereas class selectors can be used across a number of elements in an XHTML document, ID selectors can only be assigned to **one specific element** in any given XHTML document.

CSS ID Selectors

- In the CSS, id selectors are always defined with a # (hash) symbol first:

```
#summary { font-style: italic; }
```

- Again, this is CSS shorthand for…

```
*#summary { font-style: italic; }
```

- …meaning the id #summary can be applied to any XHTML element (but only <u>one</u> element can have that id name in the XHTML document).

Class Selectors vs ID Selectors

ID selectors:

1. As they must be unique to a page, ID selectors are useful for persistent structural elements, such as navigation zones, or key content areas, that occur once on a page, but that are consistent across a site.
2. For example, #mainNav may be the selector to style the the main navigation element, which will likely appear on every page of your site.
3. So, ID selectors are generally applied to conceptually similar elements across a site.

Class Selectors vs ID Selectors

Class selectors:

1. As they can be allied to any number of elements on a page, class selectors are useful for identifying (and targeting) types of content, or similar items.
2. For example, you have a news page with a date at the start of each story. If you use ID selectors, you'd have to give every occurrence of the date a separate ID name. Much easier to give every date one class name and style that one class.

CSS Pseudo-Classes

- Pseudo-classes are CSS classes used to add effects to certain elements. They are used most often to style the anchor elements <a> of hyperlinks. Example:

```
a:link { color: blue; text-decoration: underlined; }
```

- Can also be written without the a (anchor) element:

```
:link { color: blue; text-decoration: underlined; }
```

CSS Pseudo-Classes

- There are four pseudo-class elements provided to make rollover and on-click effects possible:
  - a:link { color: blue; text-decoration: underlined; }
    link not yet visited
  - a:visited { color: green; text-decoration: underlined; }
    visited link
  - a:hover { color: red; text-decoration: none; }
    effect on the link when the mouse hovers over it
  - a:active { color: purple; text-decoration: none; }
    effect on the link when the mouse button is pressed down on it
- Note that pseudo-classes for rollover effects must be written in this order in a CSS file for them to work correctly.

Combining Pseudo-Classes with Classes

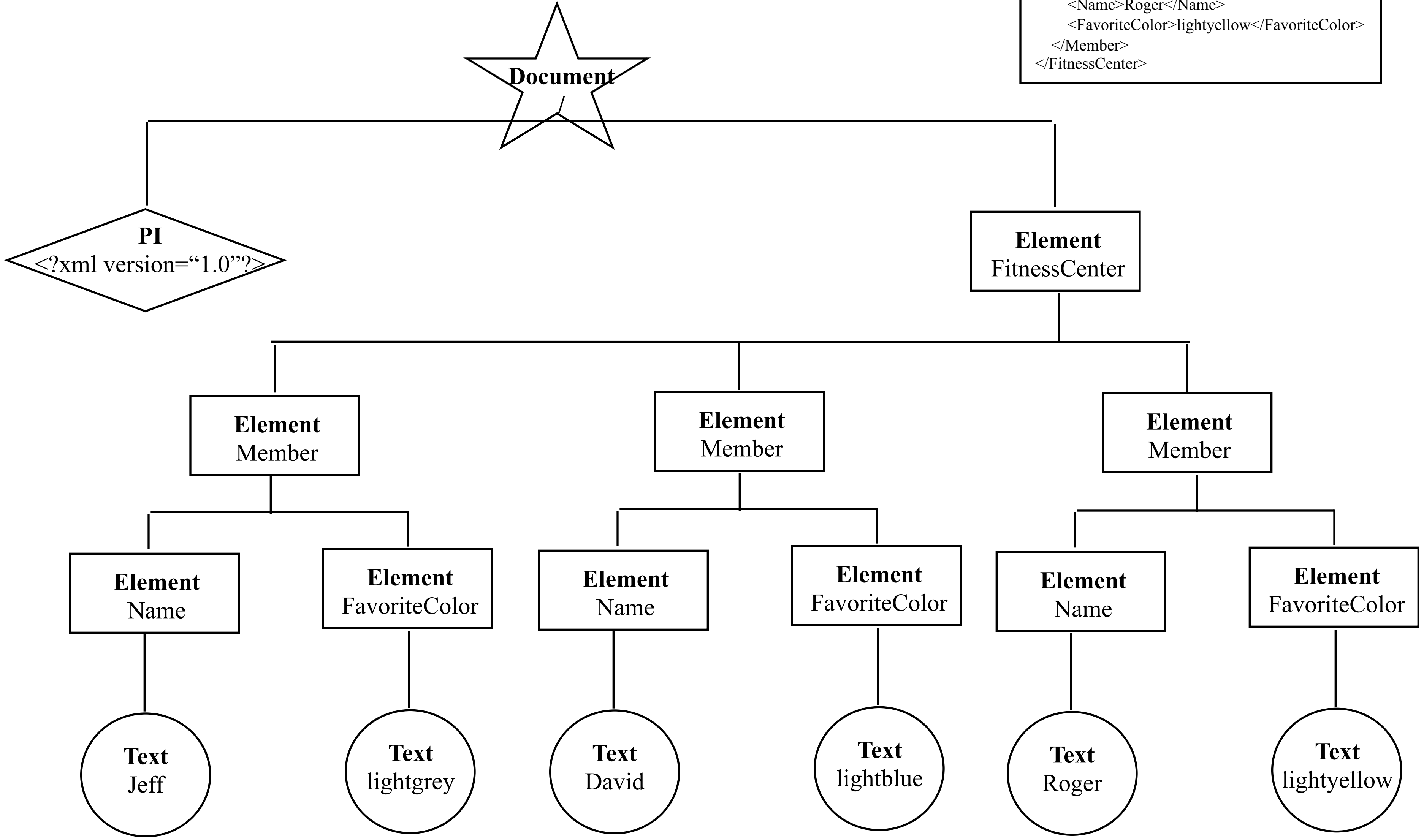
- Pseudo-classes can also be combined with regular CSS class selectors to create multiple link and rollover styles, depending on the parent class. Examples:

```
a.main:link { color: blue; }
a.sidebar:link { color: grey; }
a.footer:link { color: white; }
```

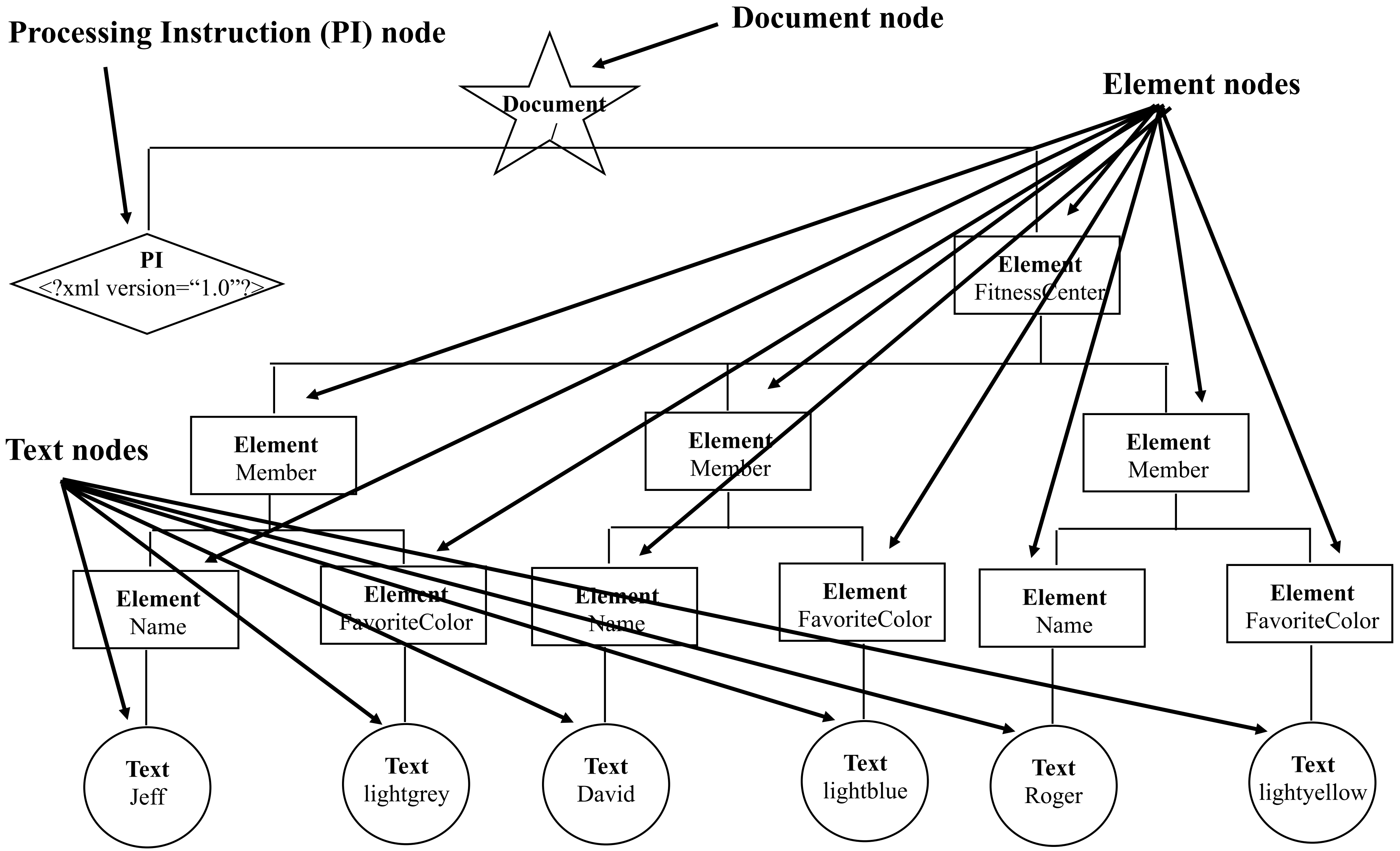
XPath

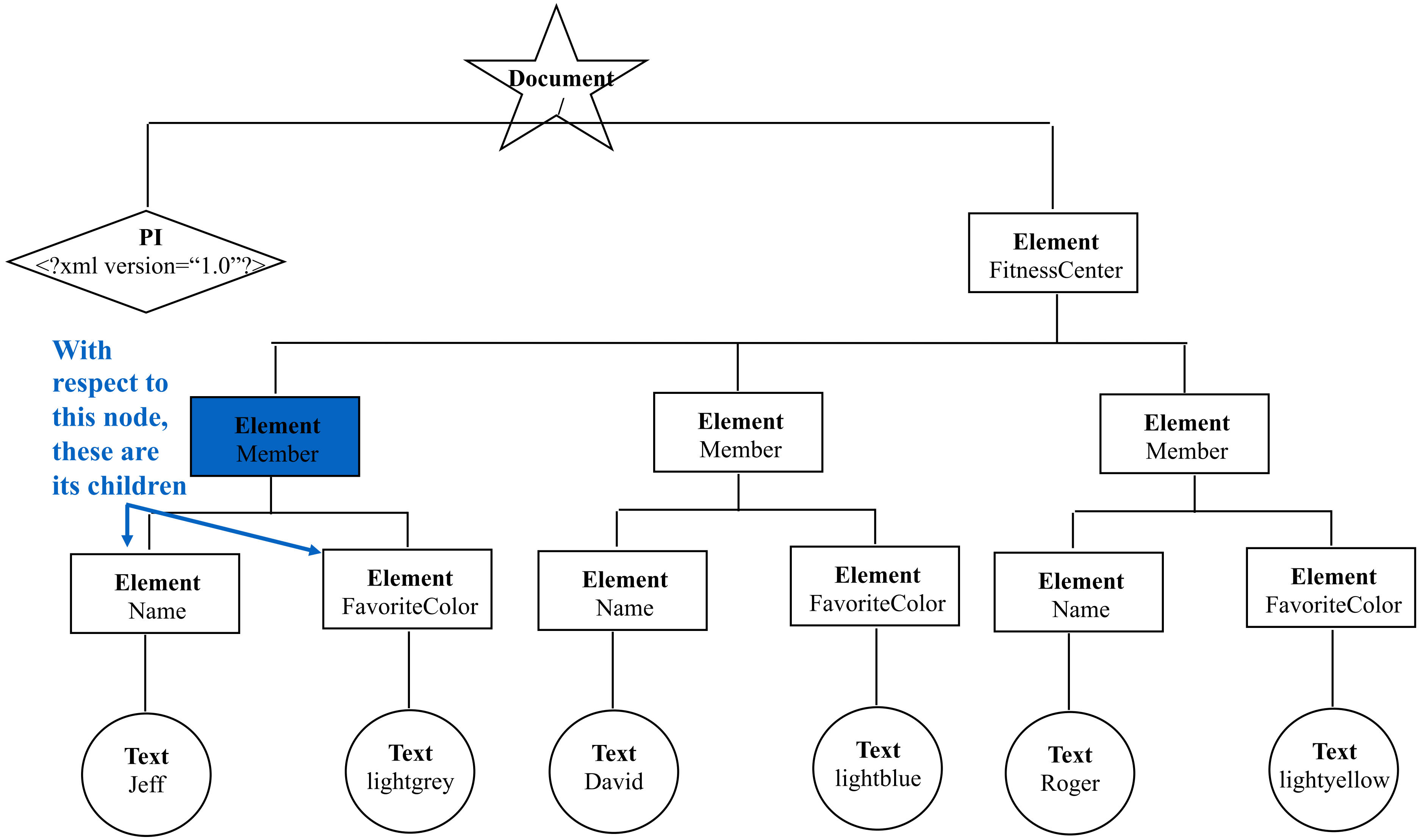
```
<?xml version="1.0"?>
<FitnessCenter>
  <Member>
    <Name>Jeff</Name>
    <FavoriteColor>lightgrey</FavoriteColor>
  </Member>
  <Member>
    <Name>David</Name>
    <FavoriteColor>lightblue</FavoriteColor>
  </Member>
  <Member>
    <Name>Roger</Name>
    <FavoriteColor>lightyellow</FavoriteColor>
  </Member>
</FitnessCenter>
```

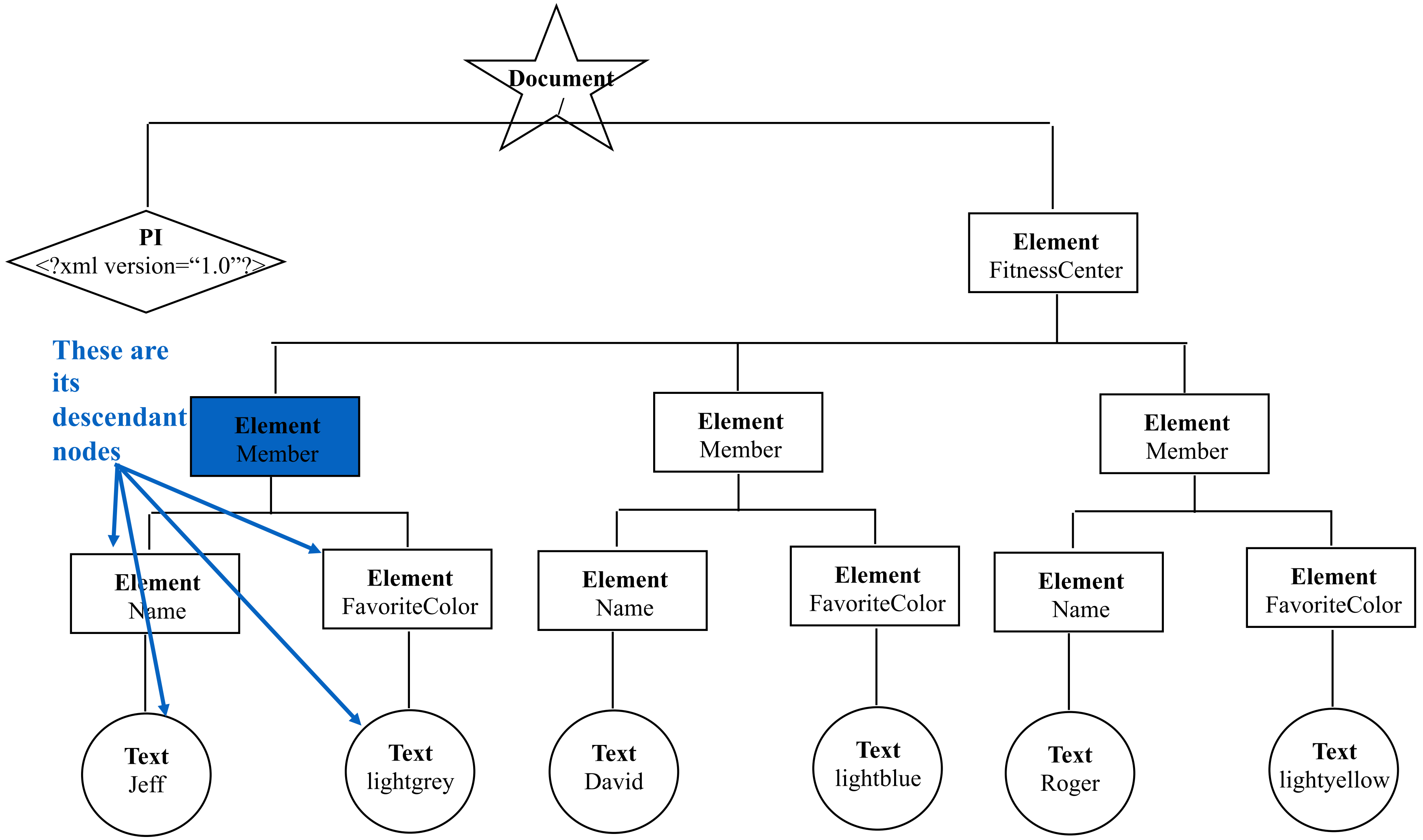
This XML document can be represented as a tree, as shown below

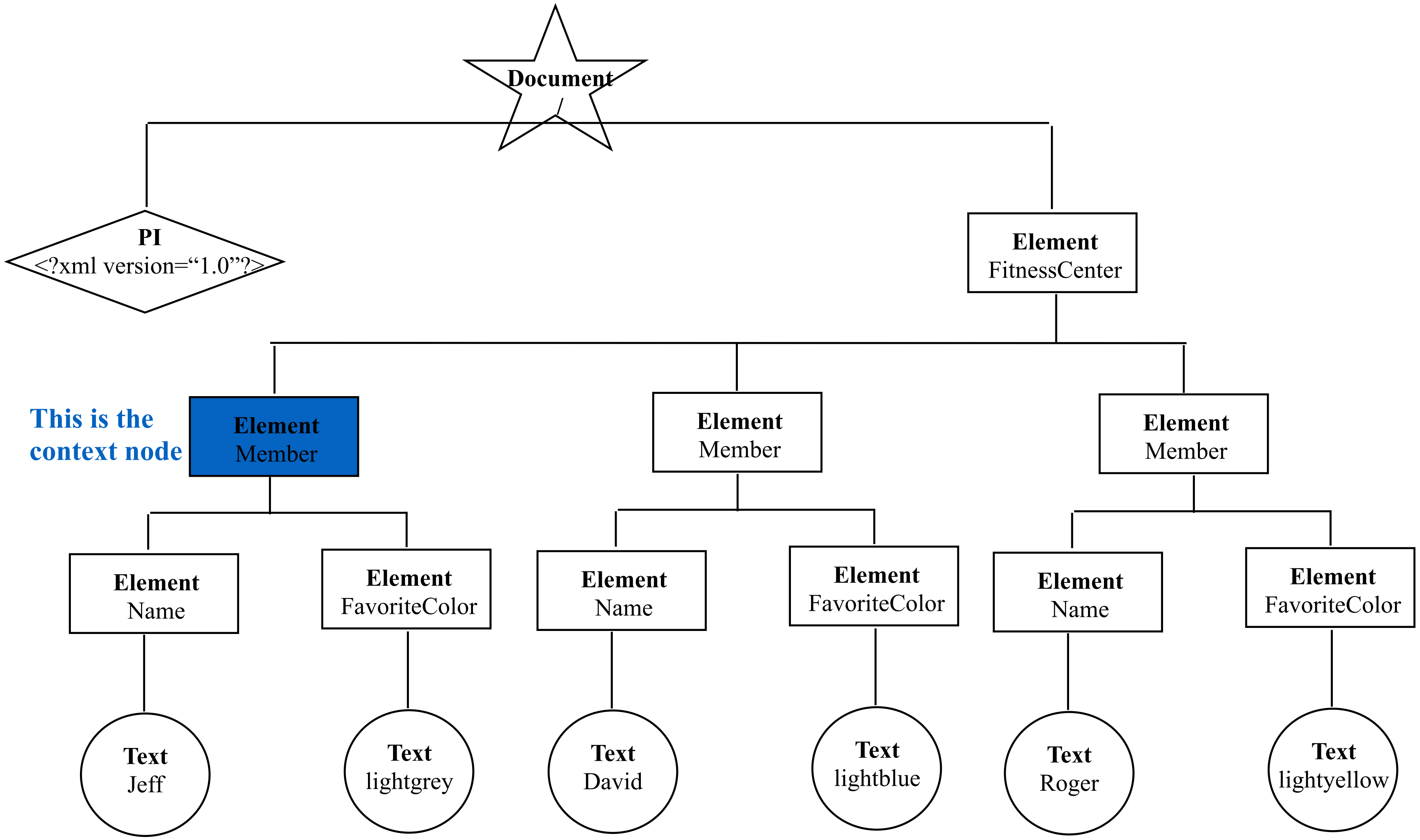


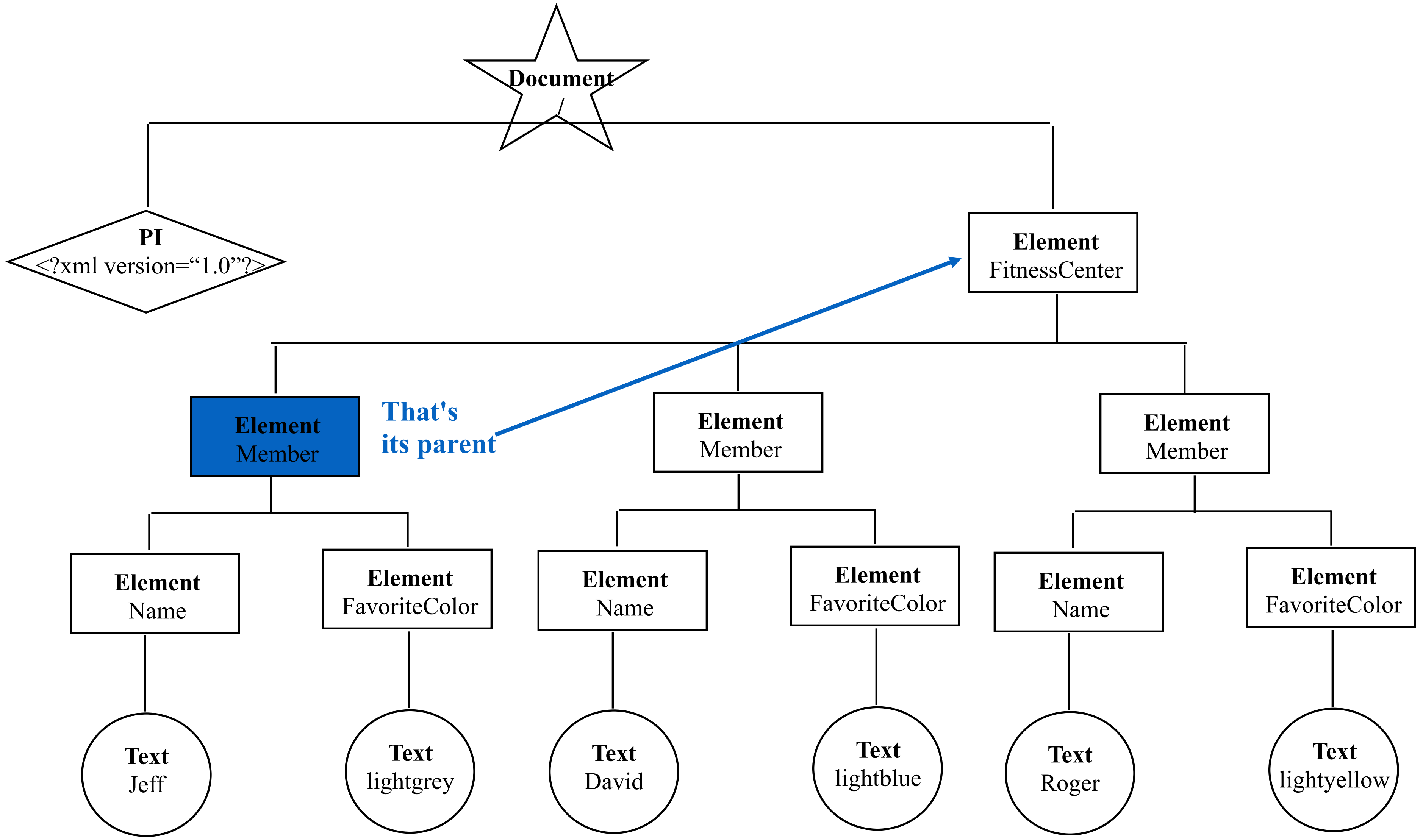
Terminology - node

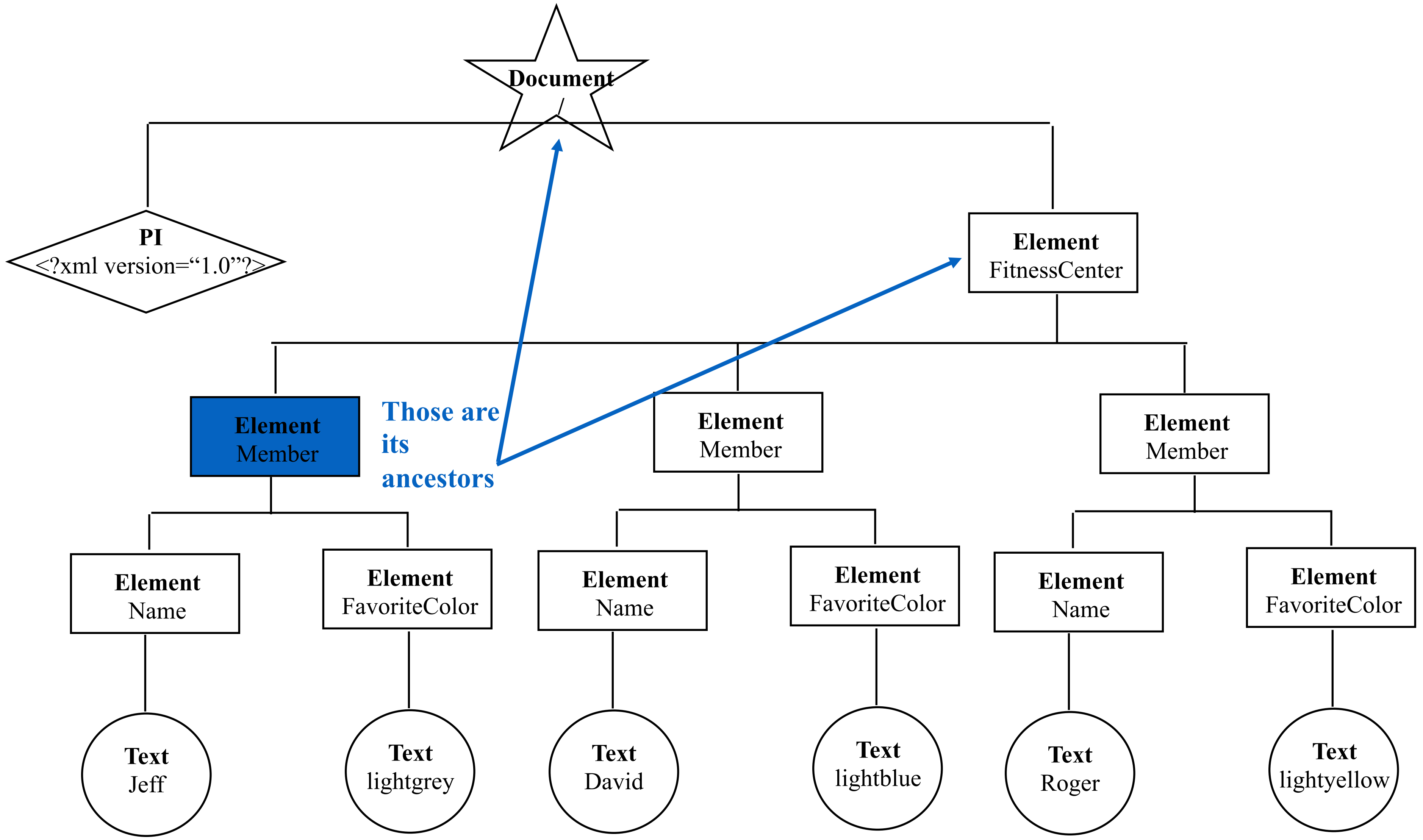


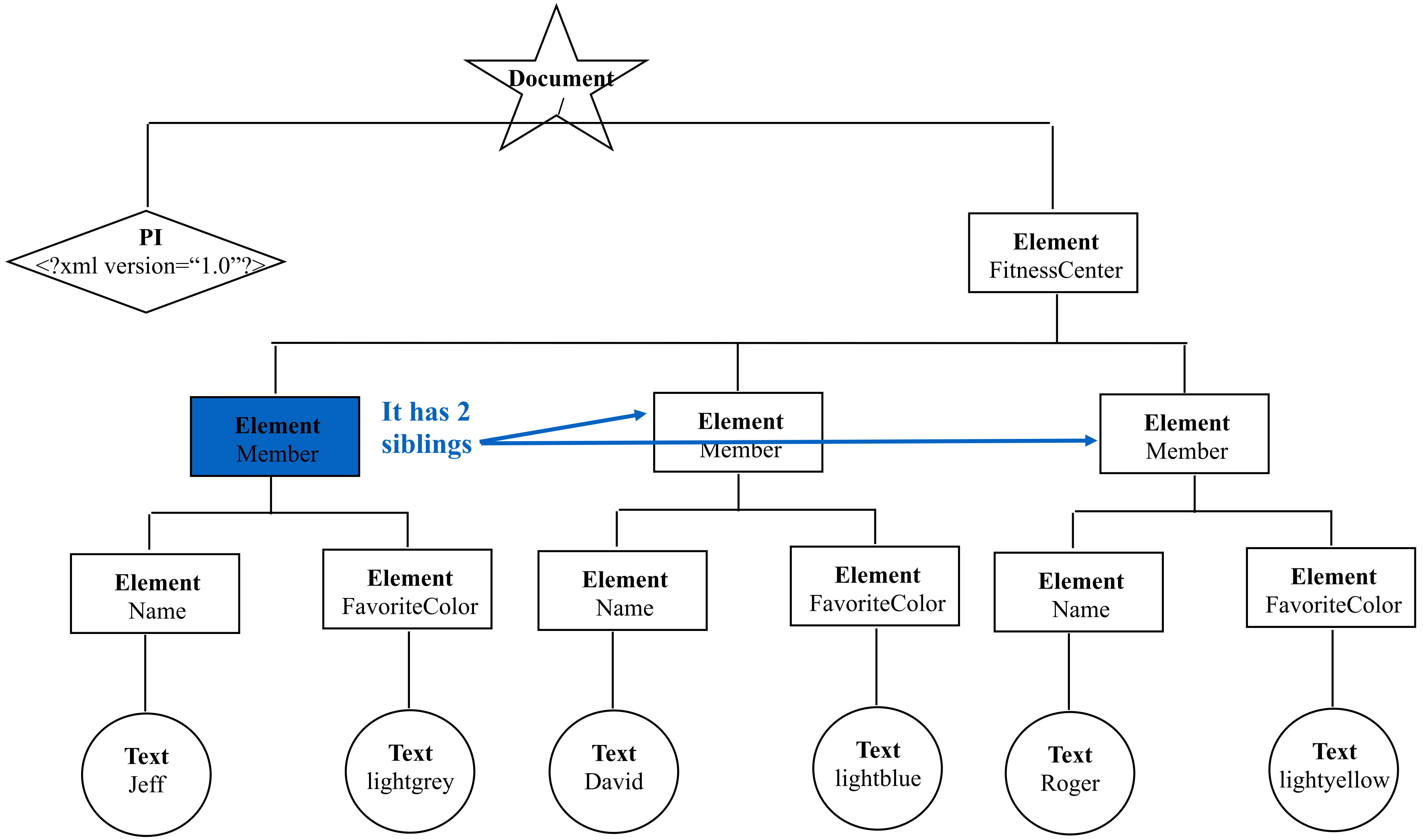

Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
With respect to this node, these are its children
Element
Member
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow

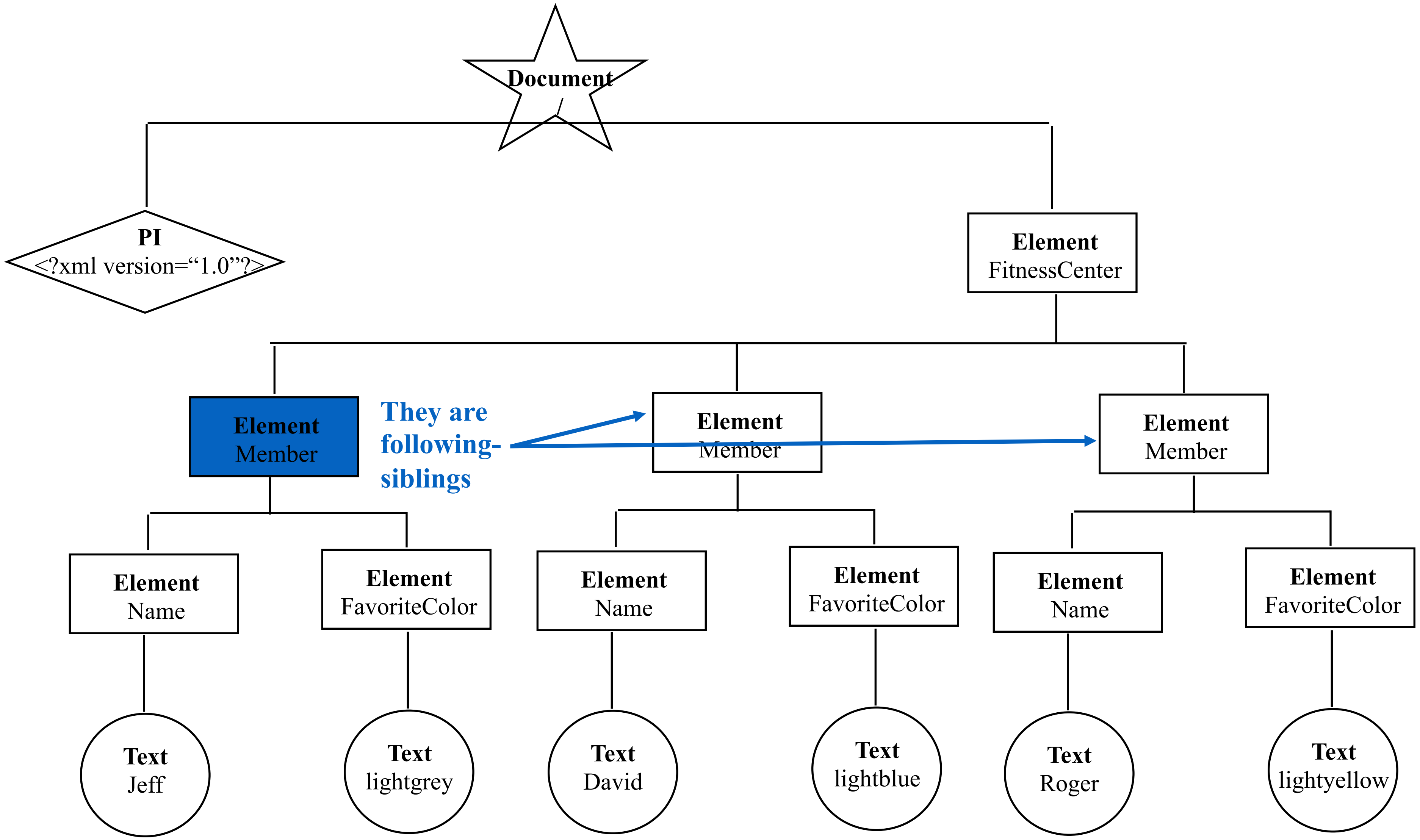

Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
These are its descendant nodes
Element
Member
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow

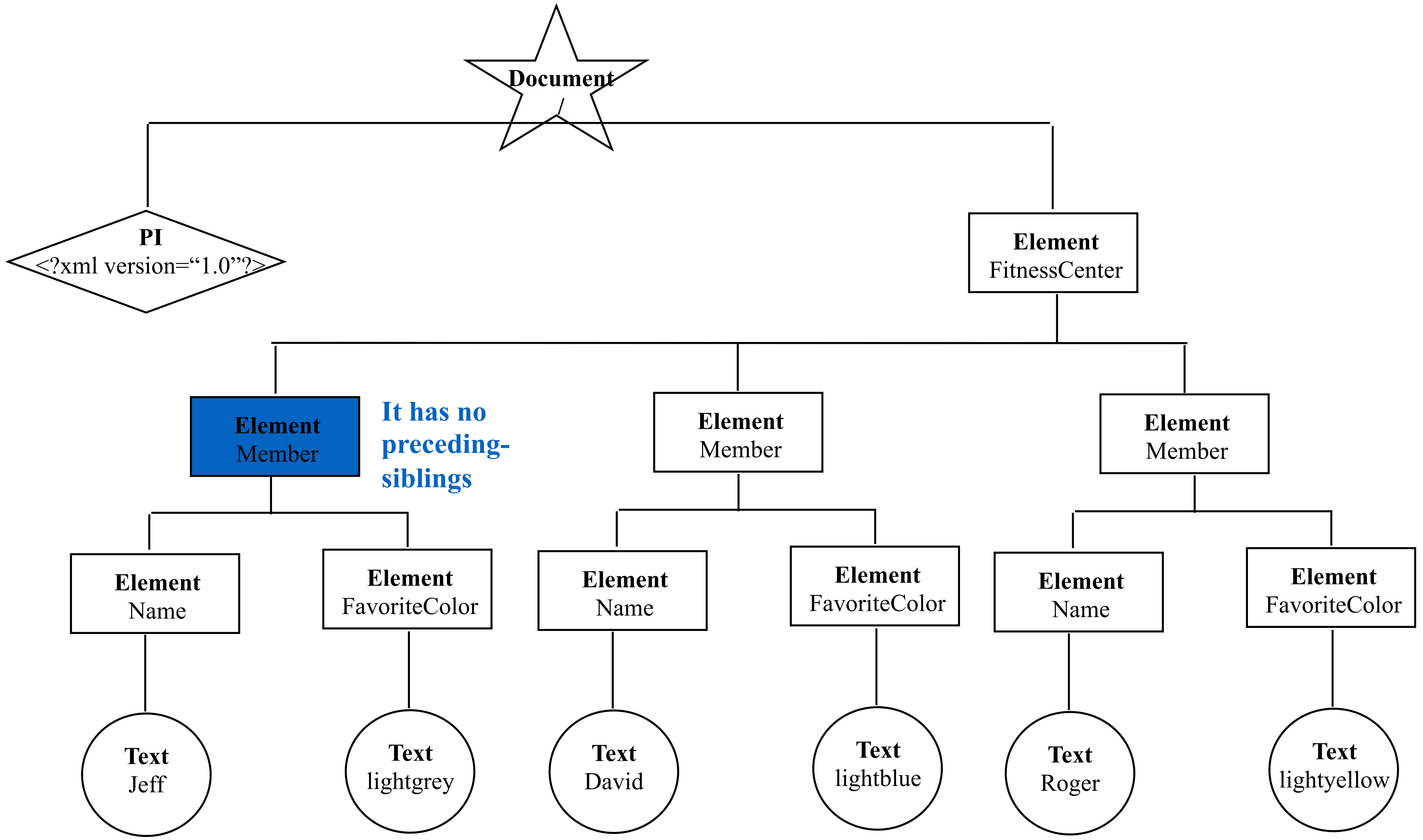

Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
This is the context node
Element
Member
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow


Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Element
Member
That's
its parent
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow


Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Element
Member
Those are its ancestors
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow


Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Element
Member
It has 2 siblings
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow


Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Element
Member
They are following-siblings
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow


Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Element
Member
It has no preceding-siblings
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow

Capabilities of XPath

- XPath provides a syntax for:
  - navigating around an XML document
  - selecting nodes and values
  - comparing node values
  - performing arithmetic on node values
- XPath provides some functions (e.g., concat(), substring(), etc.) to facilitate the above.

Select all Para Elements

//Para

↑

descendents

Select the first Para

```
//Para[1]
```

Select the last Para

```
//Para[last()]
```

Select the classification attribute of the first Para

```
//Para[1]/@classification
```

Is the Document element’s classification top-secret?

```
/Document/@classification = 'top-secret'
```

Is the Document element’s classification top-secret or secret?

```
(/Document/@classification = 'top-secret') or
(/Document/@classification='secret')
```

Logical Operators

```
A or B
A and B
not(A)
```

Select all Para’s with a secret classification

```
//Para[@classification = 'secret']
```

Check that no Para has a top-secret classification

```
not(//Para[@classification = 'top-secret'])
```

Select the Following Siblings

```
following-sibling::*
```

Select the First Following Sibling

```
following-sibling::*[1]
```

Select the Following <u>Para</u> Siblings

following-sibling::Para

Select all Following Siblings

```
following-sibling::*
```

Select all Preceding Siblings

```
preceding-sibling::*
```

Get parent element's classification

```
../@classification
```

Axis

following-sibling
preceding-sibling
child
parent
ancestor
descendent
self

Count the number of Para elements

```
count(//Para)
```

Count the number of Para elements with secret classification

```
count(//Para[@classification = 'secret'])
```

Does the first Para element contain the string “SCRIPT”?

```
contains(//Para[1], 'SCRIPT')
```

Select all nodes containing the string "SCRIPT"

```
//node()[contains(., 'SCRIPT')]
```

The node() function matches on these nodes:
- element
- text
- comment
- processing instructions (PIs)

Note that it does not match on these nodes:
- attribute
- document

Count the number of nodes containing the string “SCRIPT”

```
count(//node()[contains(., 'SCRIPT')])
```

Select the first 20 characters of the first Para

```
substring(//Para[1], 1, 20)
```

What's the length of the content of the first Para?

```
string-length(//Para[1])
```

Convert Document’s classification to lowercase

```
translate(/Document/@classification, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz')
```

Boolean Operators

eq means equal
ne means not equal
lt means less than
gt means greater than
le means less than or equal to
ge means greater than or equal to

If Document's classification is top-secret then there can be no Para with a classification not equal to top-secret

```
if (/Document/@classification eq 'top-secret') then not(//Para[@classification ne 'top-secret']) else true()
```

Two built-in functions

true()

false()

Cast a value to a numeric type

```
number(Cost)
```

The sum() function

```
<?xml version="1.0"?>
<numbers>
    <number>23</number>
    <number>5</number>
    <number>-41</number>
    <number>50</number>
    <number>12</number>
</numbers>
```

sum(//number)
→ returns 49.0

lecture04.web.scraping.ipynb

Assignment: Hacker News

- Hacker News: `https://news.ycombinator.com/`
- Please upload your Jupyter notebook here that contains the implementation of this function:

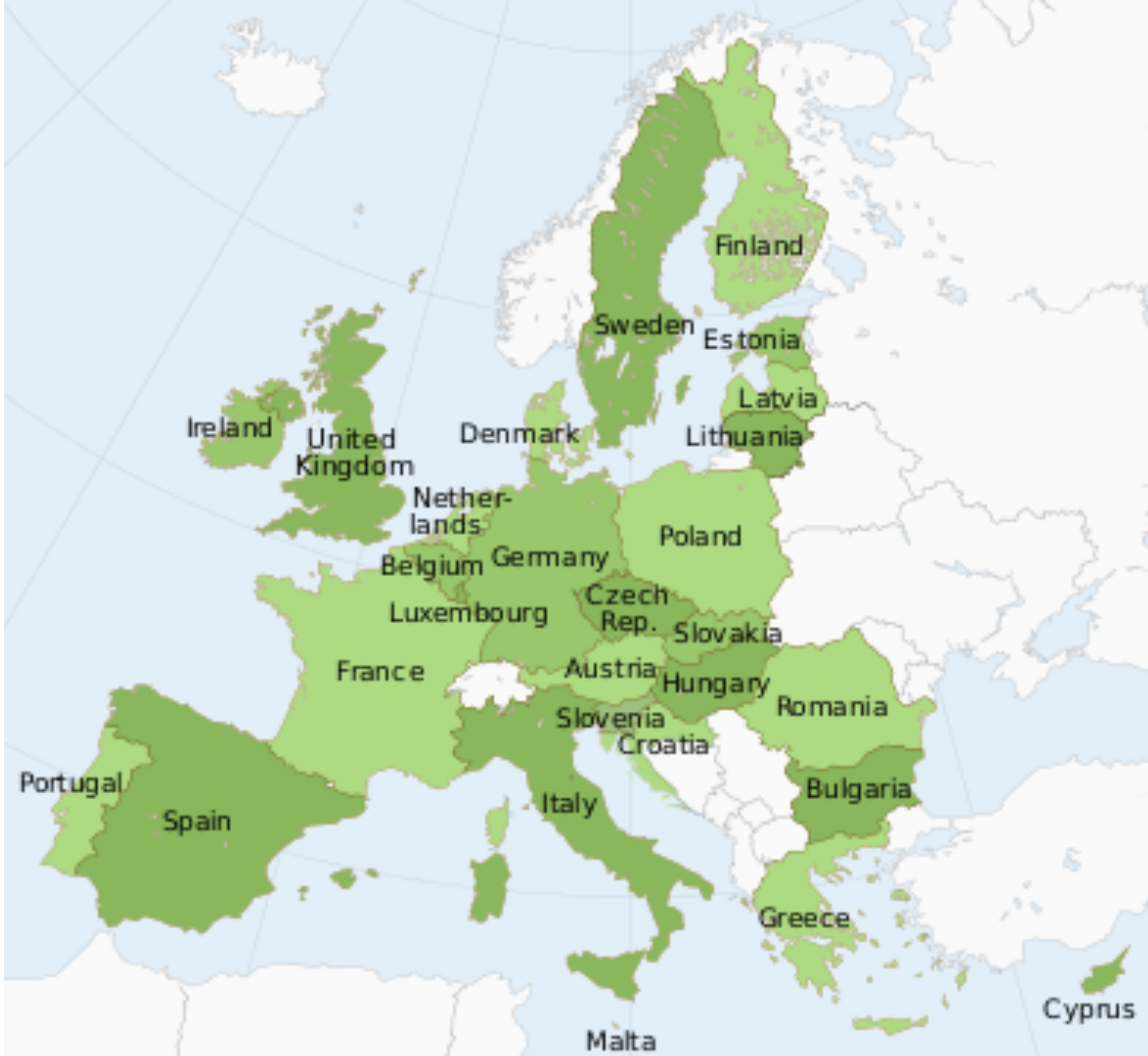
```
# Print out the average points of the posters to frontpage of HackerNews

def getExperiencePointsNow():
    # stuff here
    print "Average experience now: %.2f" % (avgScore)

getExperiencePointsNow()
```

- Explain how articles get ranked & pushed to frontpage of Hacker News (`https://news.ycombinator.com/`)

Agenda

- Pandas
- Web Scraping
  - CSS
  - XPath
  - Hacker News
- **Merging Data**
- Pandas Input / Output

European Union
circa 2016



Extra-EU Trade Data for 2010, 2012, 2014

http://ec.europa.eu/eurostat/web/products-datasets/-/ext_lt_invcur

“Extra-EU trade” statistics cover the trading of goods between Member States and a non-member countries.

SITC : Standard International Trade Classification

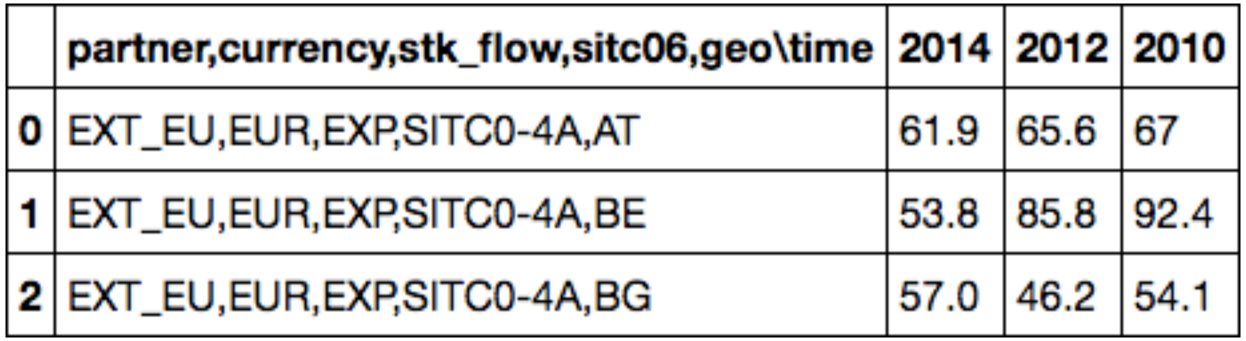
SITC0-4A

Extra-EU Trade Data for 2010, 2012, 2014

```
partner,currency,stk_flow,sitc06,geo\time       2014    2012    2010
EXT_EU,EUR,EXP,SITC0-4A,AT      61.9    65.6    67
EXT_EU,EUR,EXP,SITC0-4A,BE      53.8    85.8    92.4
EXT_EU,EUR,EXP,SITC0-4A,BG      57      46.2    54.1
EXT_EU,EUR,EXP,SITC0-4A,CY      79.1    60.7    61.4
EXT_EU,EUR,EXP,SITC0-4A,CZ      58.3    66.7    59.1
EXT_EU,EUR,EXP,SITC0-4A,DE      62.5    61.5    65.9
EXT_EU,EUR,EXP,SITC0-4A,DK      12.8    14      12.2
EXT_EU,EUR,EXP,SITC0-4A,EA      60.7    65.4    64.1
EXT_EU,EUR,EXP,SITC0-4A,EE      67.9    62.8    51.8
EXT_EU,EUR,EXP,SITC0-4A,EL      60.3    58.4    59
EXT_EU,EUR,EXP,SITC0-4A,ES      61.8    63.7    75.6
EXT_EU,EUR,EXP,SITC0-4A,EU      50.1    53.5    53.2
EXT_EU,EUR,EXP,SITC0-4A,FI      42.4    40.7    47.7
EXT_EU,EUR,EXP,SITC0-4A,FR      63.8    62.3    58.4
EXT_EU,EUR,EXP,SITC0-4A,HR      77.3    :       :
EXT_EU,EUR,EXP,SITC0-4A,HU      45.4    45.5    67.8
...
```

Read in by chunk of 100 rows

```
df = pd.DataFrame()

for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
        df = pd.concat([df, chunk])
```



	partner,currency,stk_flow,sitc06,geo\time	2014	2012	2010
0	EXT_EU,EUR,EXP,SITC0-4A,AT	61.9	65.6	67
1	EXT_EU,EUR,EXP,SITC0-4A,BE	53.8	85.8	92.4
2	EXT_EU,EUR,EXP,SITC0-4A,BG	57.0	46.2	54.1

Transforming column 1 : step 1 (splitting)

```
df = pd.DataFrame()

for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
        data_rows = [row for row in chunk.ix[:,0].str.split(',')]
        data_cols = chunk.columns[0].split(',')
        print(data_rows[:2], data_cols)
        break;
```

```
([['EXT_EU', 'EUR', 'EXP', 'SITC0-4A', 'AT'], ['EXT_EU', 'EUR', 'EXP',
'SITC0-4A', 'BE']], ['partner', 'currency', 'stk_flow', 'sitc06',
'geo\\time'])
```

Transforming column 1 : step 2 (fixing colname)

```
df = pd.DataFrame()

for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
       data_rows = [row for row in chunk.ix[:,0].str.split(',')]
       data_cols = [col.split('\\')[0] for col in chunk.columns[0].split(',')]
       print(data_rows[:2], data_cols)
       break;
```

```
([['EXT_EU', 'EUR', 'EXP', 'SITC0-4A', 'AT'], ['EXT_EU', 'EUR', 'EXP',
'SITC0-4A', 'BE']], ['partner', 'currency', 'stk_flow', 'sitc06', 'geo'])
```

Transforming column 1 : step 3 (merge)

```
df = pd.DataFrame()
for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
      data_rows = [row for row in chunk.ix[:,0].str.split(',')]
      data_cols = [col.split('\\')[0] for col in chunk.columns[0].split(',')]
      clean_df = pd.DataFrame(data_rows, columns=data_cols)

      # now we can concat by "column" which means axis=1
      new_df = pd.concat([clean_df, chunk], axis=1)
      print(new_df)
      break;
```

```
partner currency stk_flow sitc06 geo \
0 EXT_EU EUR EXP SITC0-4A AT
1 EXT_EU EUR EXP SITC0-4A BE
2 EXT_EU EUR EXP SITC0-4A BG
3 EXT_EU EUR EXP SITC0-4A CY
4 EXT_EU EUR EXP SITC0-4A CZ
```

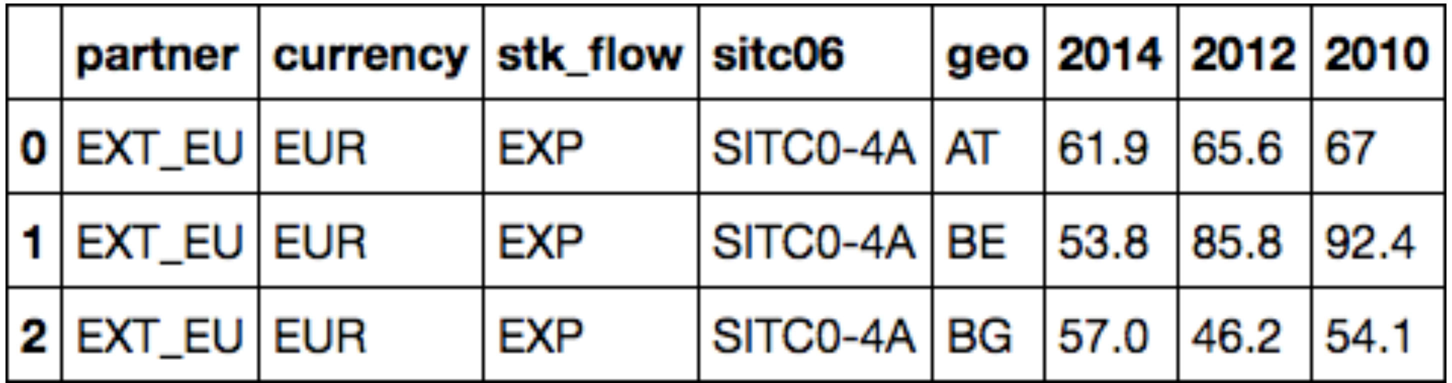
Transforming column 1 : step 4 (clean)

```
df = pd.DataFrame()
for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
        data_rows = [row for row in chunk.ix[:,0].str.split(',')]
        data_cols = [col.split('\\')[0] for col in chunk.columns[0].split(',')]
        clean_df = pd.DataFrame(data_rows, columns=data_cols)

        # now we can concat by "column" which means axis=1
        new_df = pd.concat([clean_df, chunk.drop(chunk.columns[0], axis=1)],
                                axis=1)
        print(new_df)
        break;
```

```
partner currency stk_flow sitc06 geo
0 EXT_EU EUR EXP SITC0-4A AT
1 EXT_EU EUR EXP SITC0-4A BE
2 EXT_EU EUR EXP SITC0-4A BG
3 EXT_EU EUR EXP SITC0-4A CY
4 EXT_EU EUR EXP SITC0-4A CZ
```

Transforming column 1 : step 5 (finalize)

```
df = pd.DataFrame()
for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
       data_rows = [row for row in chunk.ix[:,0].str.split(',')]
       data_cols = [col.split('\\')[0] for col in chunk.columns[0].split(',')]
       clean_df = pd.DataFrame(data_rows, columns=data_cols)

       # now we can concat by "column" which means axis=1
       new_df = pd.concat([clean_df, chunk.drop(chunk.columns[0], axis=1)],
                            axis=1)
       df = pd.concat([df, new_df])
```



	partner	currency	stk_flow	sitc06	geo	2014	2012	2010
0	EXT_EU	EUR	EXP	SITC0-4A	AT	61.9	65.6	67
1	EXT_EU	EUR	EXP	SITC0-4A	BE	53.8	85.8	92.4
2	EXT_EU	EUR	EXP	SITC0-4A	BG	57.0	46.2	54.1

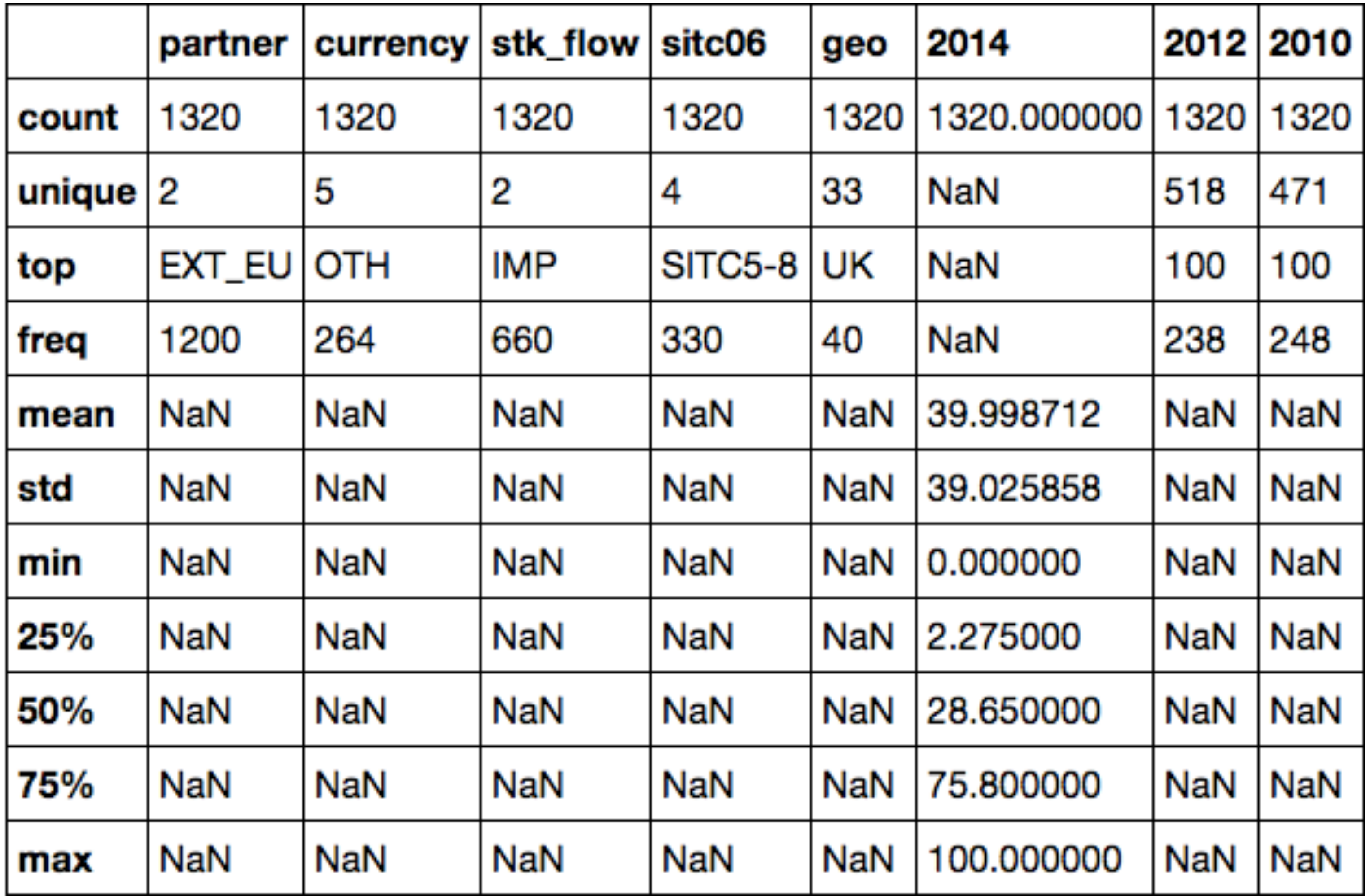
Data Exploration

```
df.shape()
```

```
(1320, 8)
```

Data Exploration

```
df.describe(include='all')
```



	partner	currency	stk_flow	sitc06	geo	2014	2012	2010
count	1320	1320	1320	1320	1320	1320.000000	1320	1320
unique	2	5	2	4	33	NaN	518	471
top	EXT_EU	OTH	IMP	SITC5-8	UK	NaN	100	100
freq	1200	264	660	330	40	NaN	238	248
mean	NaN	NaN	NaN	NaN	NaN	39.998712	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	39.025858	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	2.275000	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	28.650000	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	75.800000	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	100.000000	NaN	NaN

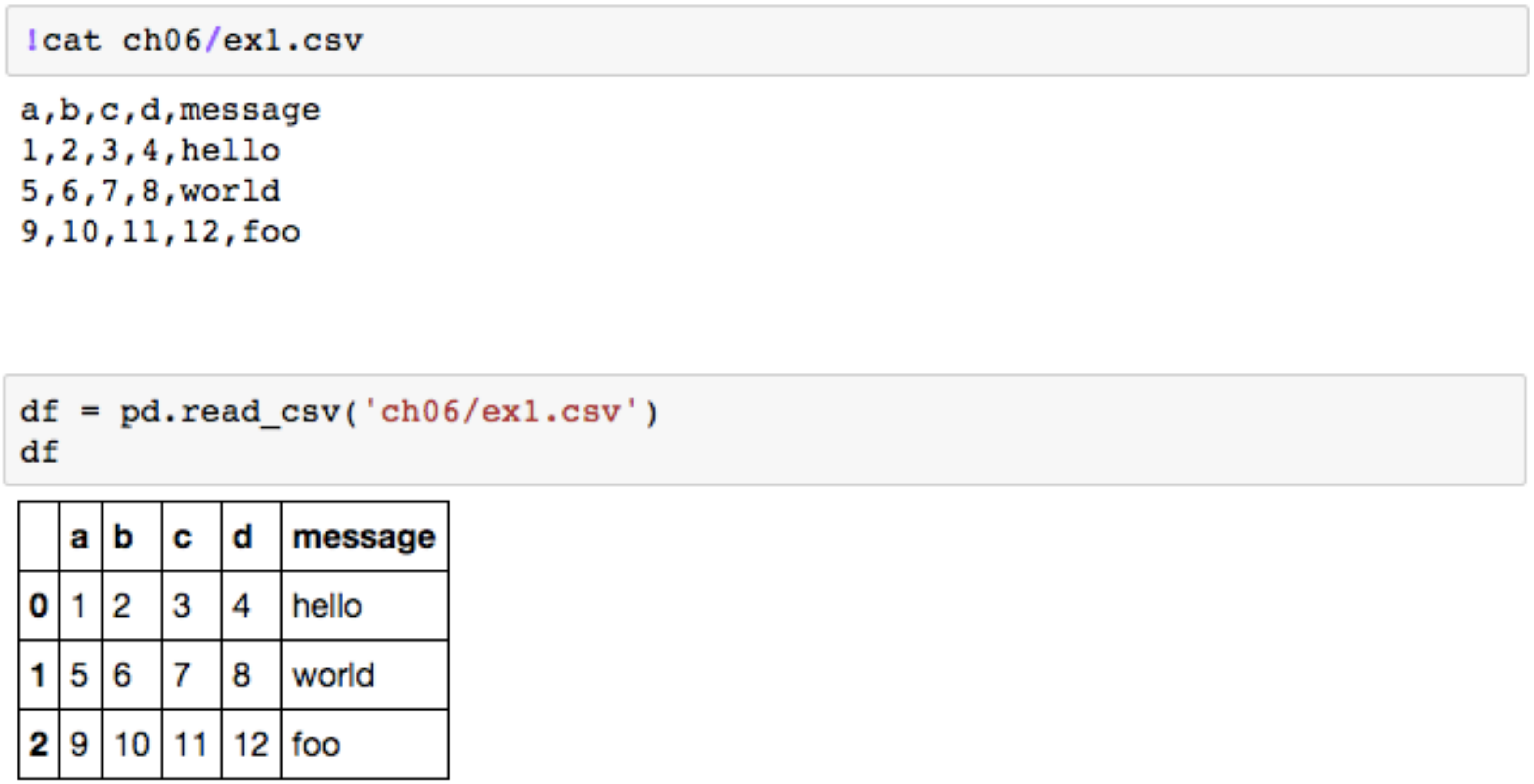
Group Exercise

- Find the “mean” 2014 EU export % to Extra-EU states with:
  - sitc06==“SITC33”   #petroleum products
  - currency==“EUR”   # euro currency
  - Stk_flow==“EXP”   # export only

Agenda

- Pandas
- Web Scraping
  - CSS
  - XPath
  - Hacker News
- Merging Data
- **Pandas Input / Output**

Reading CSV into DataFrame



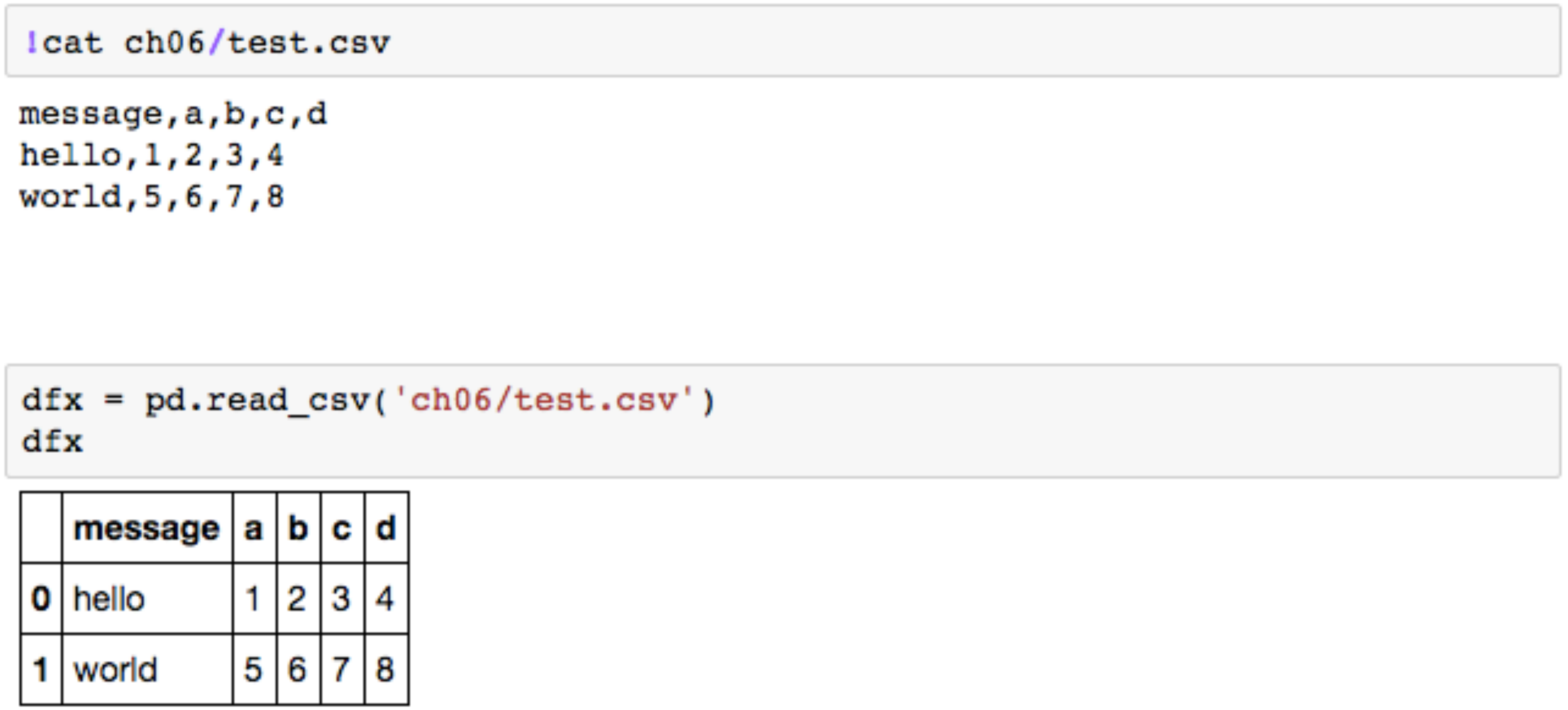
```
!cat ch06/ex1.csv
```

```
a,b,c,d,message
1,2,3,4,hello
5,6,7,8,world
9,10,11,12,foo
```

```
df = pd.read_csv('ch06/ex1.csv')
df
```

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

Reading CSV into DataFrame



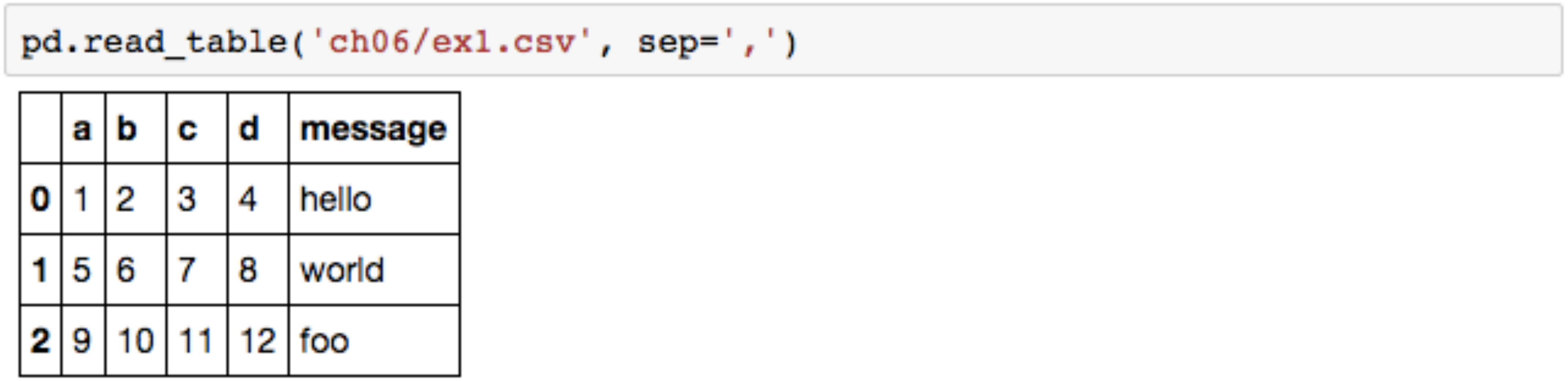
```
!cat ch06/test.csv
```

```
message,a,b,c,d
hello,1,2,3,4
world,5,6,7,8
```

```
dfx = pd.read_csv('ch06/test.csv')
dfx
```

	message	a	b	c	d
0	hello	1	2	3	4
1	world	5	6	7	8

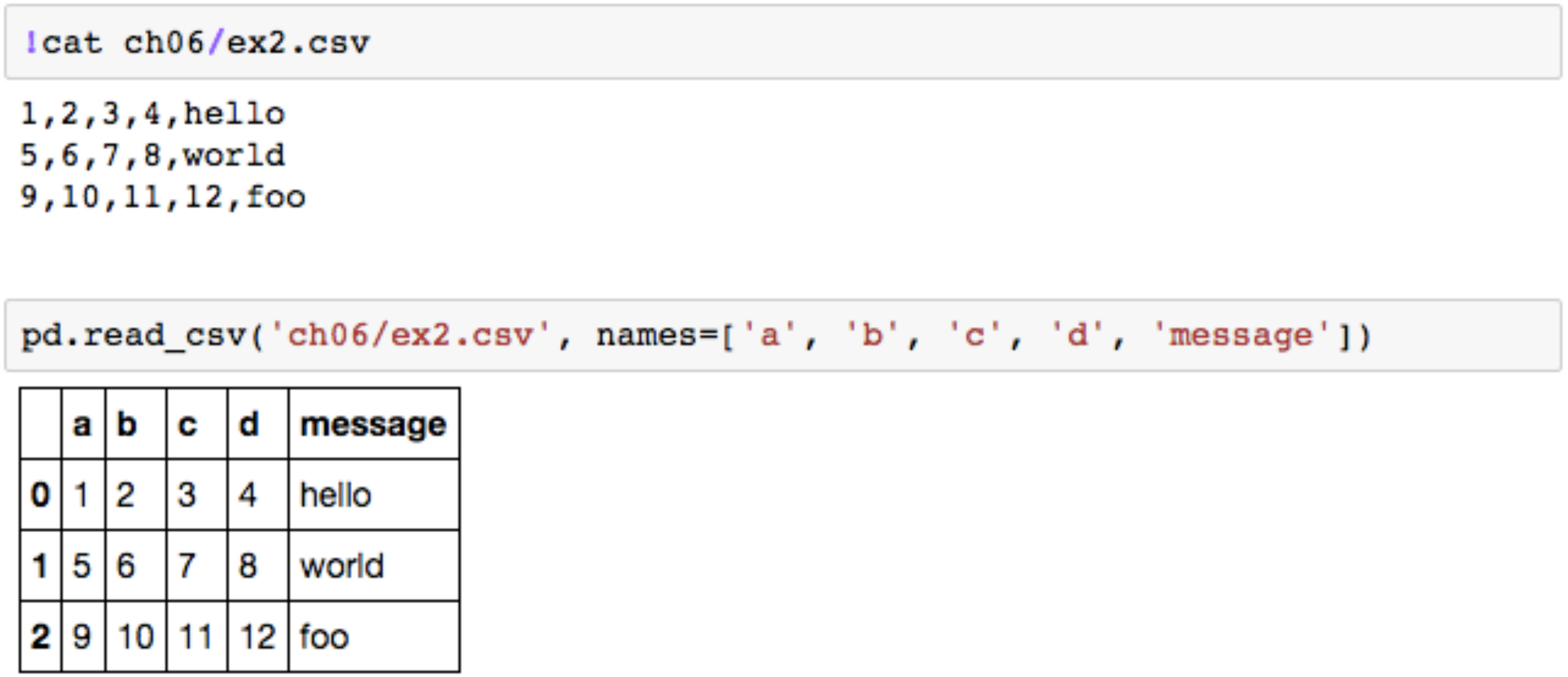
Reading CSV into DataFrame



```
pd.read_table('ch06/ex1.csv', sep=',')
```

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

Reading CSV into DataFrame



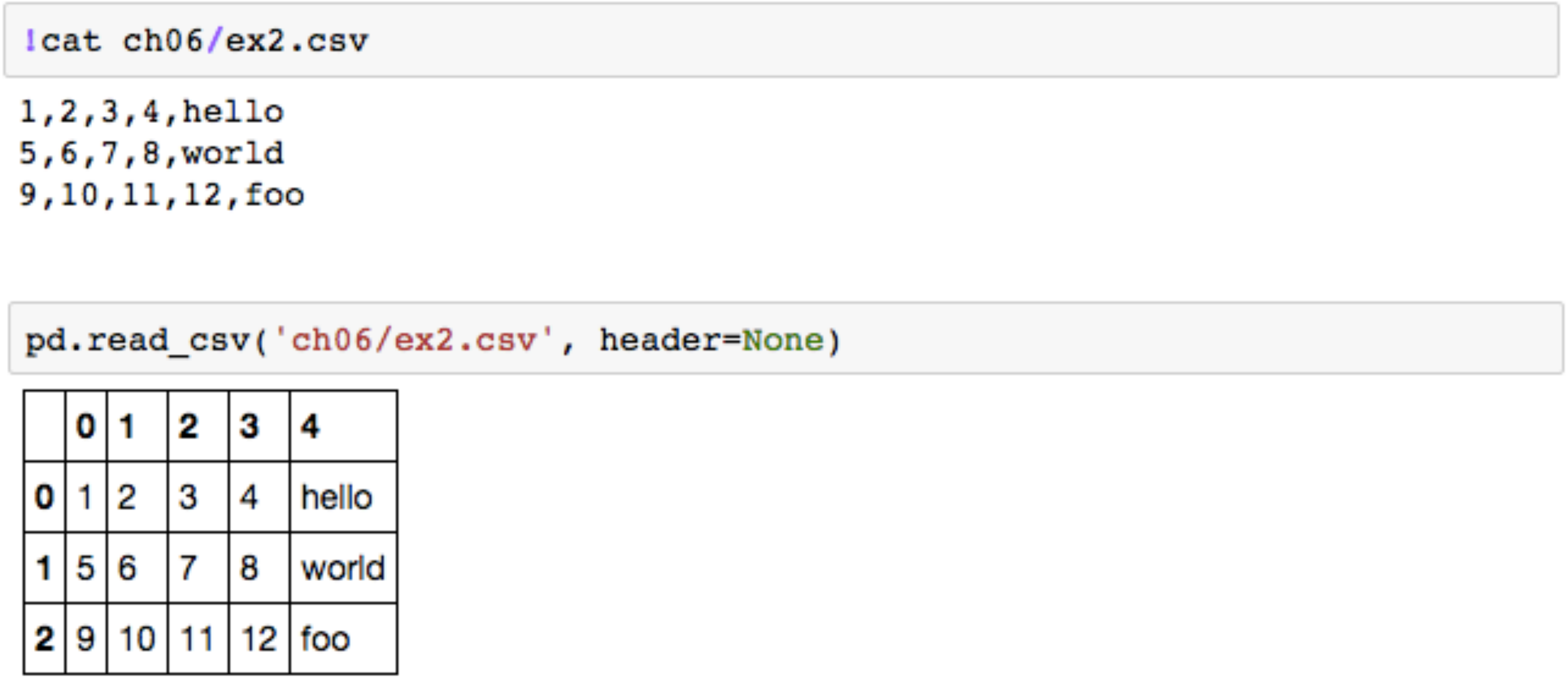
```
!cat ch06/ex2.csv
```

```
1,2,3,4,hello
5,6,7,8,world
9,10,11,12,foo
```

```
pd.read_csv('ch06/ex2.csv', names=['a', 'b', 'c', 'd', 'message'])
```

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

Reading CSV into DataFrame



```
!cat ch06/ex2.csv
```

```
1,2,3,4,hello
5,6,7,8,world
9,10,11,12,foo
```

```
pd.read_csv('ch06/ex2.csv', header=None)
```

	0	1	2	3	4
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

Reading CSV into DataFrame

```
!cat ch06/ex2.csv
```

```
1,2,3,4,hello
5,6,7,8,world
9,10,11,12,foo
```

```
names = ['a', 'b', 'c', 'd', 'message']
pd.read_csv('ch06/ex2.csv', names=names, index_col='message')
```

	a	**b**	**c**	**d**
message				
hello	1	2	3	4
world	5	6	7	8
foo	9	10	11	12

Assignment: Hacker News

- Hacker News: `https://news.ycombinator.com/`
- Please upload your Jupyter notebook here that contains the implementation of this function:

```
# Print out the average points of the posters to frontpage of HackerNews

def getExperiencePointsNow():
    # stuff here
    print "Average experience now: %.2f" % (avgScore)

getExperiencePointsNow()
```

- Explain how articles get ranked & pushed to frontpage of Hacker News (`https://news.ycombinator.com/`)

- Return 2 lists: (1) top 10 articles by points, (2) top 10 articles by comments
- For these 2 lists, find the Pearson correlation coefficient for these 2 lists
- Ask a good question & answer it from the Hacker News front page

Reminder: Midterm Next Week

- In-class (1 hour)
- Take-home (1 dataset)