

Multidimensional Array

Remember Numpy?

- So far, we have learned that Numpy can create an array that is like a list

```
>>> import numpy as np
>>> a = np.array([1,2,3])
>>> a
array([1, 2, 3])
>>> a[0]=999
>>> a
array([999,    2,    3])
>>> type(a)
<class 'numpy.ndarray'>
>>> l = [1,2,3]
>>> type(l)
<class 'list'>
```

Array Broadcasting

```
>>> a = np.array([1,2,3,4,5])
>>> a + 1  ←———— “Broadcasting”
array([2, 3, 4, 5, 6])
>>> a * 3  ←———— Different from LIST
array([ 3,  6,  9, 12, 15])
>>> a > 5
array([False, False, False, False, False], dtype=bool)
```

← Create another array with the Boolean results

Array Math

```
>>> v1 = np.array([1,2,3])  
>>> v2 = np.array([4,9,16])
```

```
>>> print(v1+v2) ←—————
```

Different
from list !!!!

```
[ 5 11 19]
```

```
>>> print(v1*v2)
```

```
[ 4 18 48]
```

```
>>> np.sqrt(v2) ←—————  
array([ 2.,  3.,  4.])
```

Directly apply on
EVERY element in the
array

```
>>> v1.dot(v2) ←————— Dot product  
70
```

```
>>> v2.sum()  
29
```

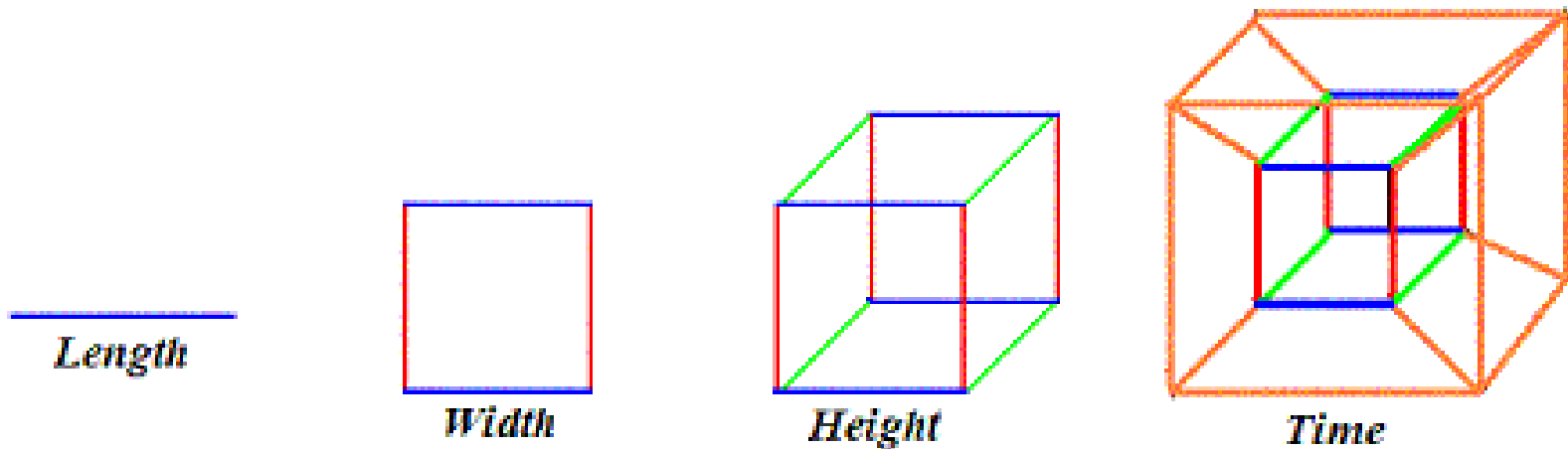
More of these functions:

<https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.math.html>

Dimensions

- Are we living in a three dimensional space?

The Four Dimensions



[The xkcd guide to the fourth dimension](#)

Motel in US



One Dimension

Hotel

My room is 02-05



Dimensions

One Dimensional Array, A

Index	Contents
0	'Apple'
1	'John'
2	'Eve'
3	'Mary'
4	'Ian'
5	'Smith'
6	'Kelvin'

A[5] = 'Smith'

Two Dimensional Array, M

	0	1	2	3
0	'Apple'	'Lah'	'Cat'	'Eve'
1	'Hello'	'Pay'	'TV'	'Carl'
2	'What'	'Bank'	'Radio'	'Ada'
3	'Frog'	'Peter'	'Sea'	'Eat'
4	'Job'	'Fry'	'Gym'	'Wow'
5	'Walk'	'Fly'	'Cook'	'Look'

M[4][1] = 'Fry'

Locating an Entry

- Creating a 3x5 array with all zeros

```
>>> c = np.zeros((3,5))
```

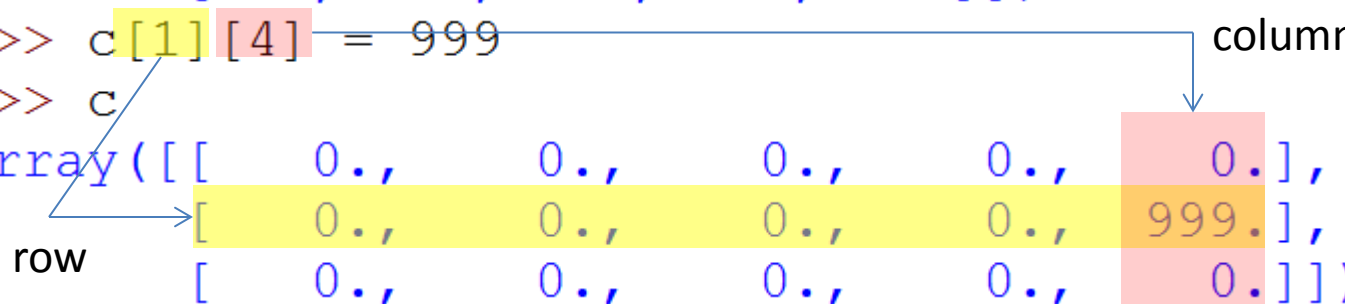
```
>>> c
```

```
array([[ 0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.,  0.]])
```

```
>>> c[1][4] = 999
```

```
>>> c
```

```
array([[ 0.,  0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.,  0., 999.],  
       [ 0.,  0.,  0.,  0.,  0.,  0.]])
```



```
>>> c[1][4]
```

```
999.0
```

```
>>> c[1,4]
```

```
999.0
```

Sub-array (Sub-matrix)

```
>>> a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
```

```
>>> a
```

```
array([[ 1,  2,  3,  4],  
       [ 5,  6,  7,  8],  
       [ 9, 10, 11, 12]])
```

```
>>> a[1:3,2:4]
```

```
array([[ 7,  8],  
       [11, 12]])
```

```
>>> a[0:2,2:]
```

```
array([[3, 4],  
       [7, 8]])
```

```
>>> a[:,0::2]
```

```
array([[ 1,  3],  
       [ 5,  7],  
       [ 9, 11]])
```

```
array([[ 1,  2,  3,  4],  
       [ 5,  6,  7,  8],  
       [ 9, 10, 11, 12]])
```

```
array([[ 1,  2,  3,  4],  
       [ 5,  6,  7,  8],  
       [ 9, 10, 11, 12]])
```

```
array([[ 1,  2,  3,  4],  
       [ 5,  6,  7,  8],  
       [ 9, 10, 11, 12]])
```

Matrix Multiplication

```
>>> m1 = np.array([[2,1],[3,10]])
```

```
>>> m2 = np.array([[4,0],[0,3]])
```

```
>>> m1
```

```
array([[ 2,  1],  
       [ 3, 10]])
```

```
>>> m2
```

```
array([[4, 0],  
       [0, 3]])
```

```
>>> np.matmul(m1,m2)
```

```
array([[ 8,  3],  
       [12, 30]])
```

```
>>> m3 = np.array([[1,2,3,4,5]])
```

```
>>> np.matmul(m1,m3)
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#119>", line 1, in <module>
```

```
    np.matmul(m1,m3)
```

```
ValueError: shapes (2,2) and (1,5) not aligned: 2 (dim 1)  
!= 1 (dim 0)
```

Boolean Array Indexing

```
>>> a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
>>> a
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])

>>> bool_idx = (a > 5)
>>> print(bool_idx)
[[False False False False]
 [False  True  True  True]
 [ True  True  True  True]]
>>> print(a[bool_idx])
[ 6  7  8  9 10 11 12]
>>> print(a[a>10])
[11 12]
```

Dimensions



One Dimensional Array, A



Index	Contents
0	'Apple'
1	'John'
2	'Eve'
3	'Mary'
4	'Ian'
5	'Smith'
6	'Kelvin'

A[5] = 'Smith'

Two Dimensional Array, M



	0	1	2	3
0	'Apple'	'Lah'	'Cat'	'Eve'
1	'Hello'	'Pay'	'TV'	'Carl'
2	'What'	'Bank'	'Radio'	'Ada'
3	'Frog'	'Peter'	'Sea'	'Eat'
4	'Job'	'Fry'	'Gym'	'Wow'
5	'Walk'	'Fly'	'Cook'	'Look'

M[4][1] = 'Fry'

Multi-Dimensional Array

- A **three** dimensional array

The image shows a 3D representation of a multi-dimensional array. The front face is a 10x5 grid of numerical values. The top face shows the array sliced along the first dimension, with indices 0 through 4. The right face shows the array sliced along the second dimension, with indices 0 through 9. Three red arrows indicate the dimensions: a vertical arrow pointing down on the left, a horizontal arrow pointing right at the bottom, and a diagonal arrow pointing up and right from the bottom-left corner.

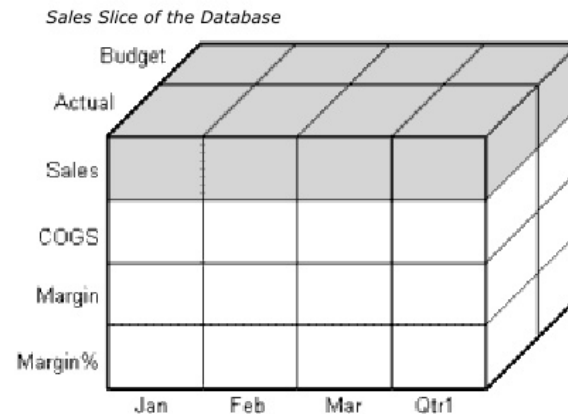
Index	0	1	2	3	4
0	65,340	12,483	138,189	902,960	633,877
1	5,246	424,642	650,380	821,254	866,122
2	89,678	236,781	601,691	329,274	913,534
3	103,902	4,567	733,611	263,010	85,550
4	2,778	658,305	128,788	978,155	620,702
5	45,024	55,058	705,586	89,672	384,605
6	780	47,538	523,784	556,801	617,107
7	32,667	350,890	834,753	638,108	85,188
8	56,083	145,582	775,040	548,322	756,587
9	41,123	543,542	537,738	513,048	418,482

Fancy Terminology in Business

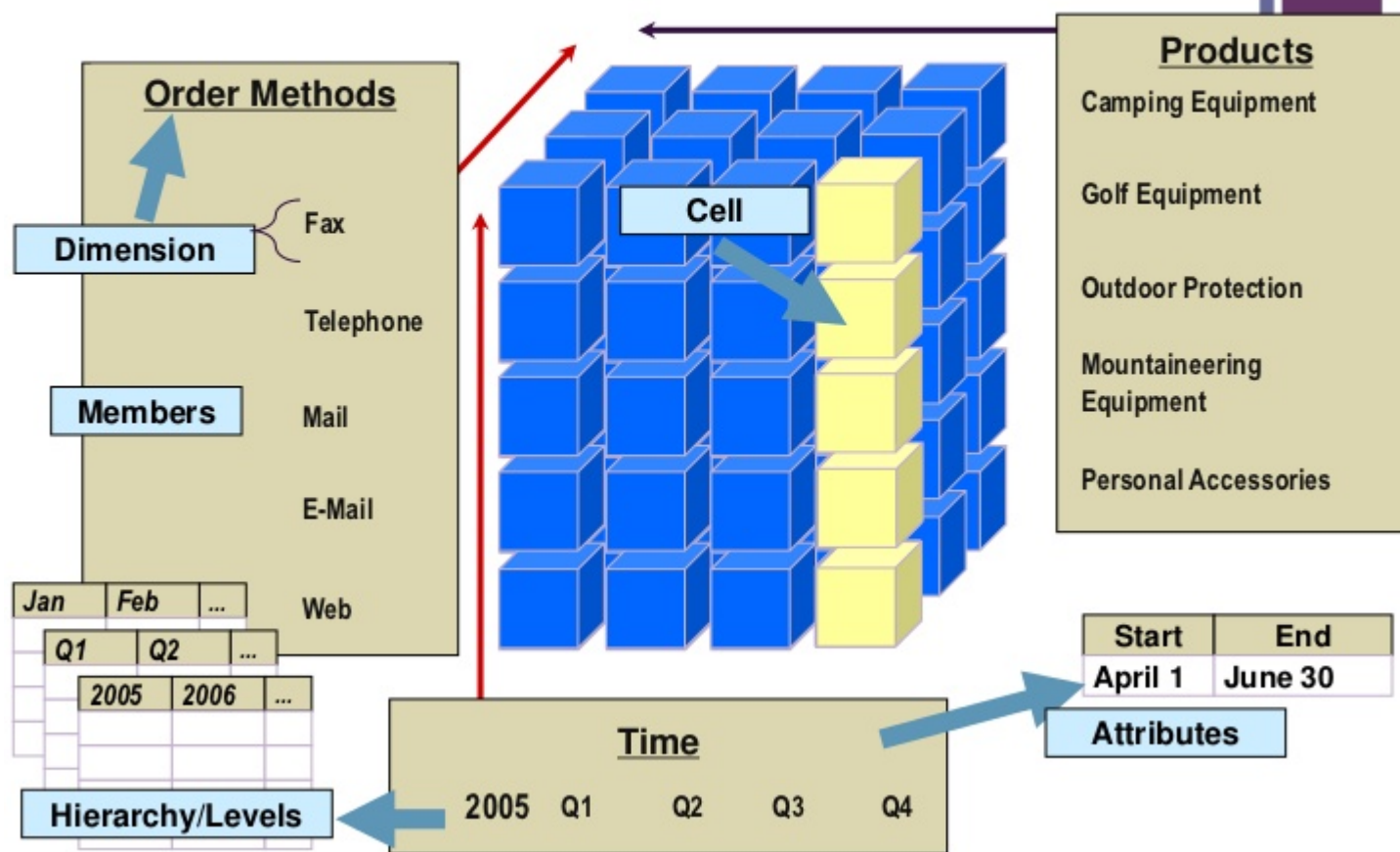
- Multidimensional Data Model
- Multidimensional Analysis



The shaded cells is called a slice illustrate that, when you refer to Sales, you are referring to the portion of the database containing eight Sales values.

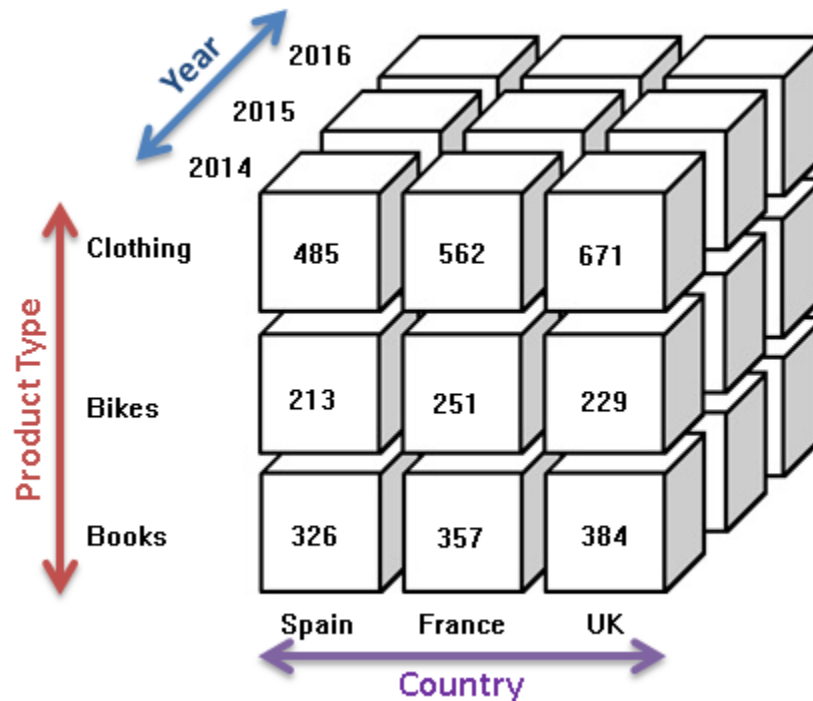


+ What's a "Cube"?



Sales Cube

- Product Type x Year x Country



- 4th Dimensional Sales Cube:
 - Product Type x Year x Country x {Predicted vs actual}

Multi-Dimensional Array

- A **three** dimensional array
 - E.g. a picture:
 - N x M pixels, and each pixel is an array of three colors, (R, G, B) (Dimensions = N x M x 3)
 - Next Week:

```
from scipy import misc, ndimage
import matplotlib.pyplot as plt

cat_pic = misc.imread('cute cat.jpg')
cat_pic2 = 255 - cat_pic
plt.figure(1)
plt.imshow(cat_pic)
plt.figure(2)
plt.imshow(cat_pic2)
plt.show()
```

Image Processing

- Original Picture

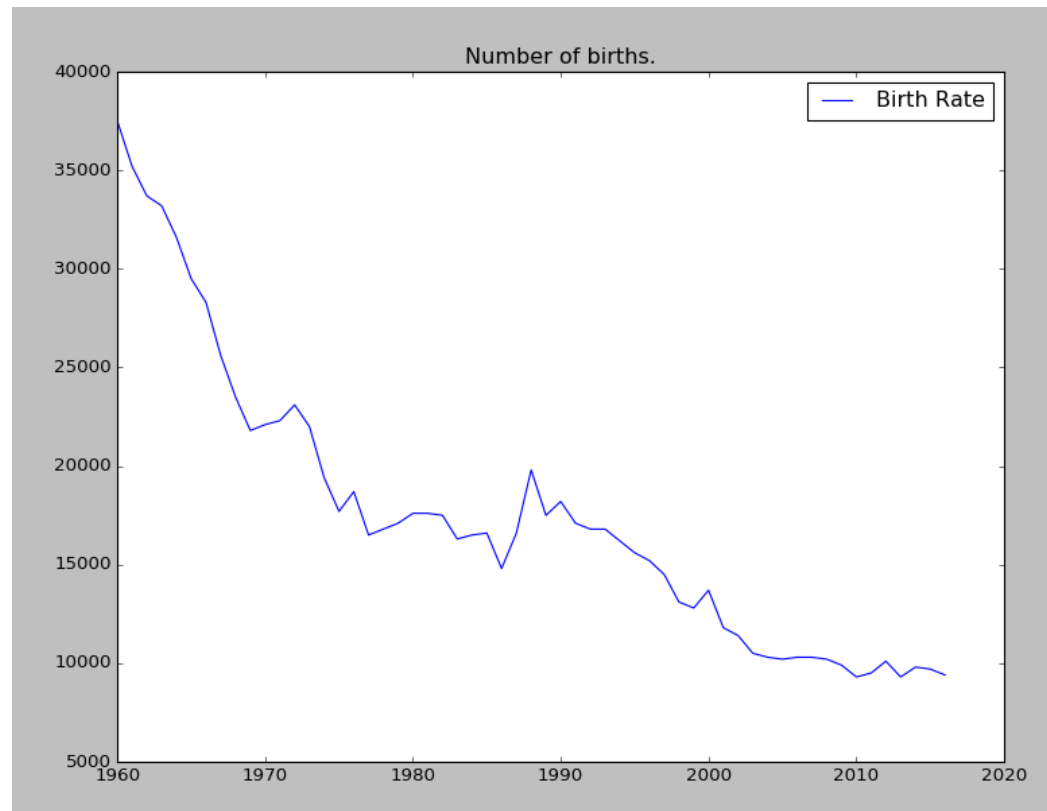


- Negative Image



2D Array x CSV

- Remember last week, we plotted the Singapore birth rate with the data in a CSV file



```
import matplotlib.pyplot as plt
```

```
def plot_birth_rate():
```

```
    with open('crude-birth-rate.csv') as f:
```

```
        f.readline()
```

```
        year = []
```

```
        num_birth = []
```

```
        for line in f:
```

```
            list_form = line.rstrip('\n').split(',')
```

```
            year.append(int(list_form[0]))
```

```
            num_birth.append(float(list_form[2])*1000)
```

```
plt.plot(year,num_birth,label="Birth Rate")
```

```
plt.legend(loc="upper right")
```

```
plt.title('Number of births.')
```

```
plt.show()
```

```
plot_birth_rate()
```

Reading CSV File

Create a CSV File
Reader

Read in a
CSV file
into a list

```
>>> import csv
>>> birth_file = open('crude-birth-rate.csv')
>>> birth_file_reader = csv.reader(birth_file)
>>> data_in_list = list(birth_file_reader)
>>> print(data_in_list)
[['year', 'level_1', 'value'], ['1960', 'Crude Birth Rate', '37.5'], ['1961', 'Crude Birth Rate', '35.2'], ['1962', 'Crude Birth Rate', '33.7'], ['1963', 'Crude Birth Rate', '33.2'], ['1964', 'Crude Birth Rate', '31.6'], ['1965', 'Crude Birth Rate', '29.5'], ['1966', 'Crude Birth Rate', '28.3'], ['1967', 'Crude Birth Rate', '25.6'], ['1968', 'Crude Birth Rate', '23.5']]
```

```
import matplotlib.pyplot as plt
import numpy as np
import csv
```

```
def plot_birth_rate():
    birth_file = open('crude-birth-rate.csv')
    birth_file_reader = csv.reader(birth_file)

    birth_data = np.array(list(birth_file_reader))
    year = birth_data[1:,0]
    num_birth = birth_data[1:,2]
    birth_file.close()

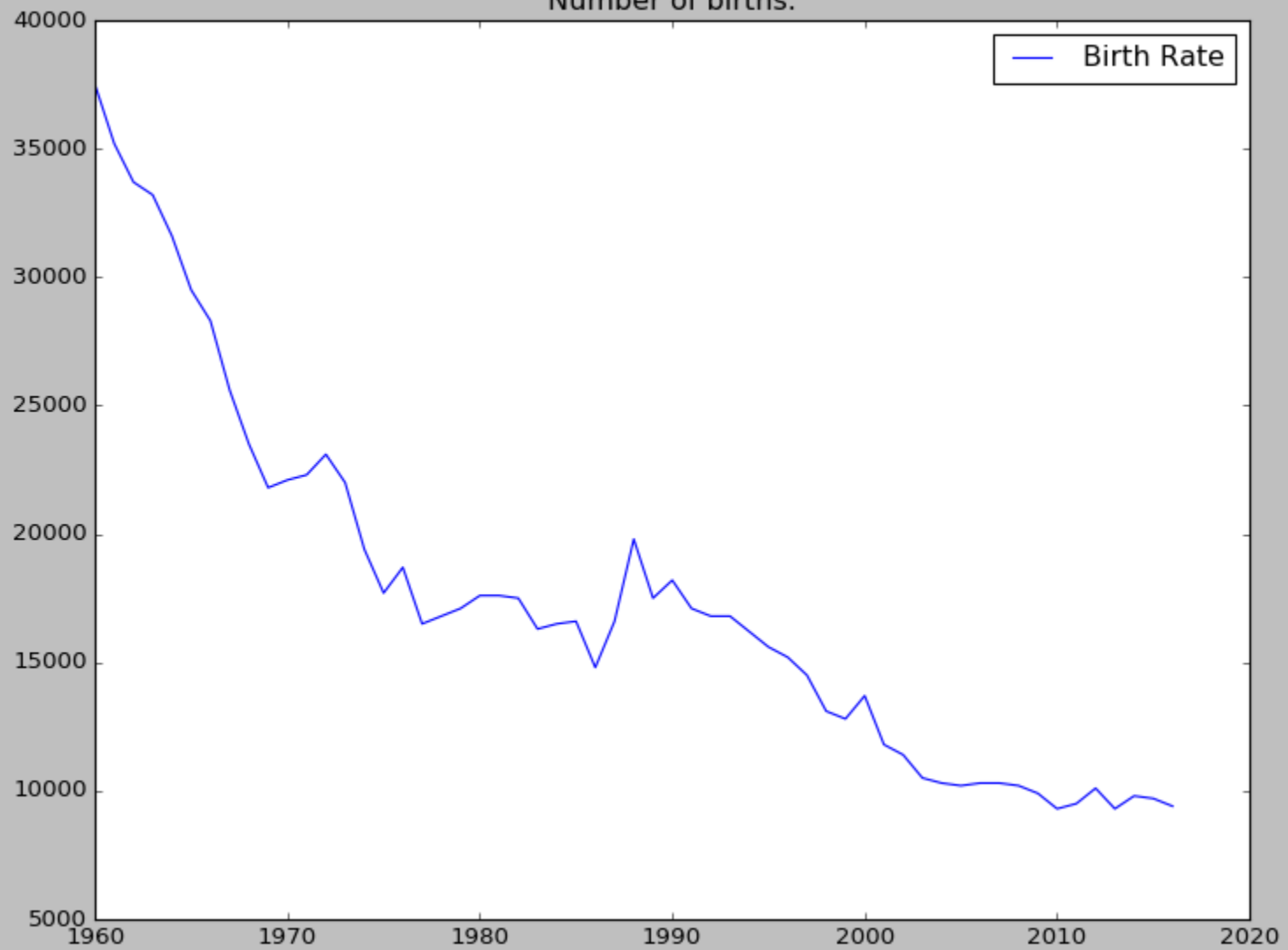
    plt.plot(year, num_birth, label="Birth Rate")
    plt.legend(loc="upper right")
    plt.title('Number of births.')
    plt.show()
```

Because we want
these cool Numpy
features

```
plot_birth_rate()
```

Actually we should close
the file always. But the
“for” loop with file close
it for you automatically

Number of births.



Announcement

- By now, you should have received your Coursemology invitation emails
- There is a Lab 000 for you to warm up with the submission
 - Not graded