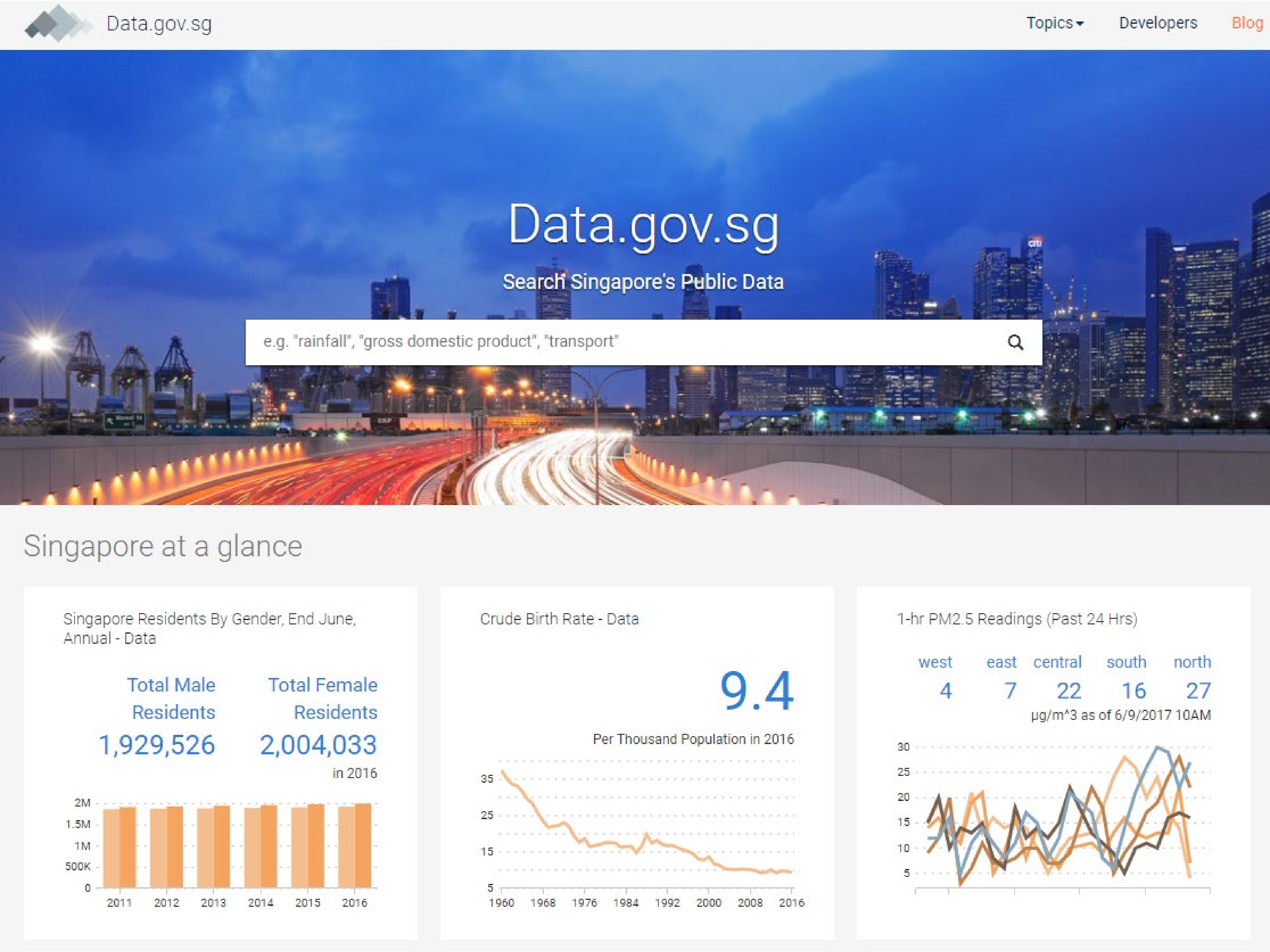
(Text) File Input/Output



Data.gov.sg

Topics Developers Blog

Data.gov.sg

Search Singapore's Public Data

e.g. "rainfall", "gross domestic product", "transport"

Singapore at a glance

Singapore Residents By Gender, End June, Annual - Data

Total Male Residents	Total Female Residents
1,929,526	2,004,033

in 2016

Crude Birth Rate - Data

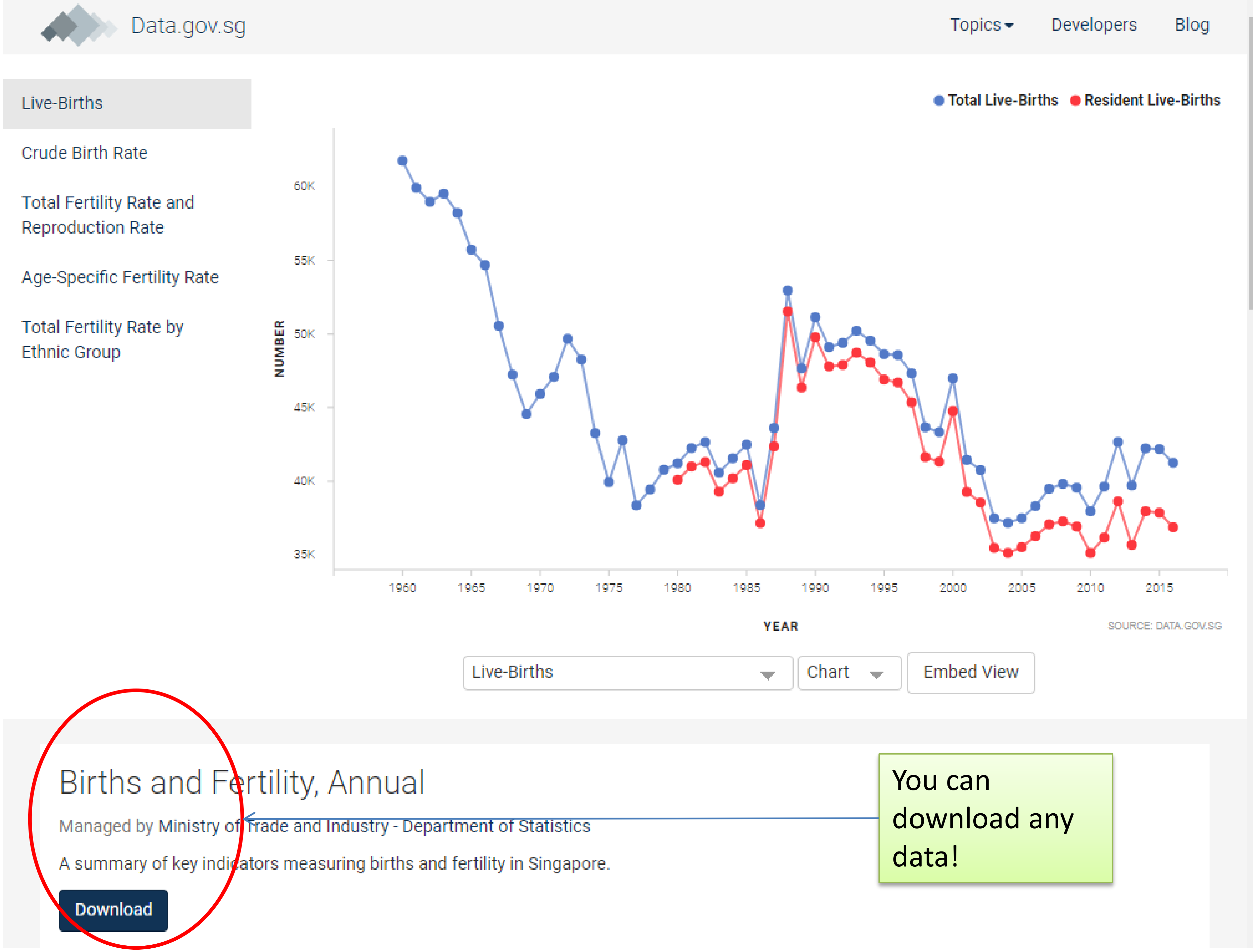
9.4

Per Thousand Population in 2016

1-hr PM2.5 Readings (Past 24 Hrs)

west	east	central	south	north
4	7	22	16	27

µg/m^3 as of 6/9/2017 10AM



Births and Fertility, Annual

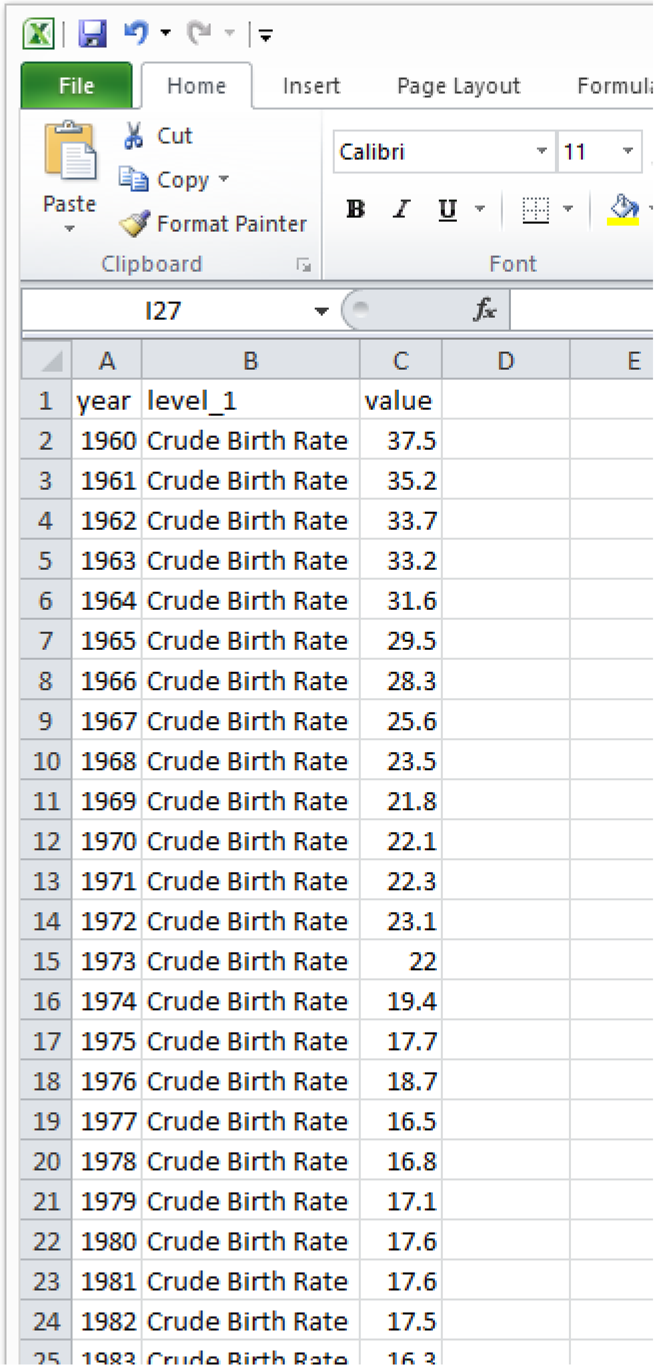
Managed by Ministry of Trade and Industry - Department of Statistics

A summary of key indicators measuring births and fertility in Singapore.

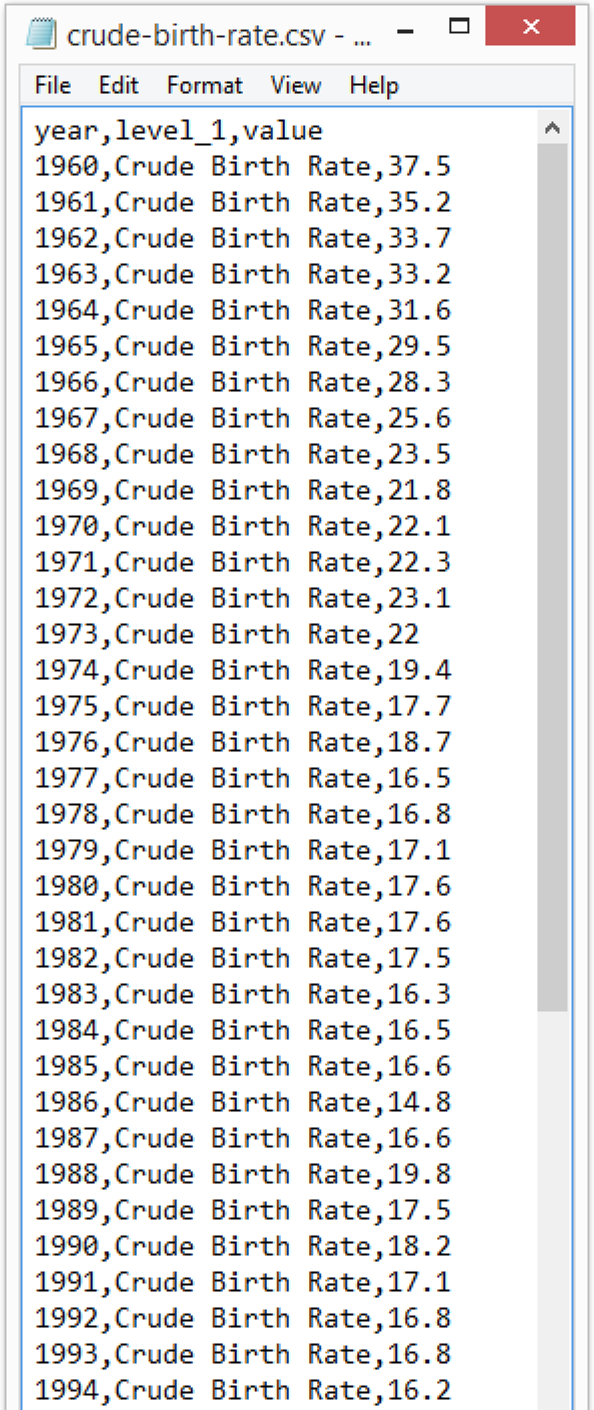
You can download any data!

Open in

- Excel
- Notepad



	A	B	C	D	E
1	year	level_1	value		
2	1960	Crude Birth Rate	37.5		
3	1961	Crude Birth Rate	35.2		
4	1962	Crude Birth Rate	33.7		
5	1963	Crude Birth Rate	33.2		
6	1964	Crude Birth Rate	31.6		
7	1965	Crude Birth Rate	29.5		
8	1966	Crude Birth Rate	28.3		
9	1967	Crude Birth Rate	25.6		
10	1968	Crude Birth Rate	23.5		
11	1969	Crude Birth Rate	21.8		
12	1970	Crude Birth Rate	22.1		
13	1971	Crude Birth Rate	22.3		
14	1972	Crude Birth Rate	23.1		
15	1973	Crude Birth Rate	22		
16	1974	Crude Birth Rate	19.4		
17	1975	Crude Birth Rate	17.7		
18	1976	Crude Birth Rate	18.7		
19	1977	Crude Birth Rate	16.5		
20	1978	Crude Birth Rate	16.8		
21	1979	Crude Birth Rate	17.1		
22	1980	Crude Birth Rate	17.6		
23	1981	Crude Birth Rate	17.6		
24	1982	Crude Birth Rate	17.5		
25	1983	Crude Birth Rate	16.3		



```
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
1969,Crude Birth Rate,21.8
1970,Crude Birth Rate,22.1
1971,Crude Birth Rate,22.3
1972,Crude Birth Rate,23.1
1973,Crude Birth Rate,22
1974,Crude Birth Rate,19.4
1975,Crude Birth Rate,17.7
1976,Crude Birth Rate,18.7
1977,Crude Birth Rate,16.5
1978,Crude Birth Rate,16.8
1979,Crude Birth Rate,17.1
1980,Crude Birth Rate,17.6
1981,Crude Birth Rate,17.6
1982,Crude Birth Rate,17.5
1983,Crude Birth Rate,16.3
1984,Crude Birth Rate,16.5
1985,Crude Birth Rate,16.6
1986,Crude Birth Rate,14.8
1987,Crude Birth Rate,16.6
1988,Crude Birth Rate,19.8
1989,Crude Birth Rate,17.5
1990,Crude Birth Rate,18.2
1991,Crude Birth Rate,17.1
1992,Crude Birth Rate,16.8
1993,Crude Birth Rate,16.8
1994,Crude Birth Rate,16.2
```

Let’s Do it in Python

- Of course, you are not going to type the data into your Python code
  - one data one code?!
  - change in data = change in code?
  - Called “Hard Coding”
- Usually practice
  - Data file +
  - Python code that can read the file

Writing A File

Actually Easier

Writing A File

Indicate the file object f is for writing

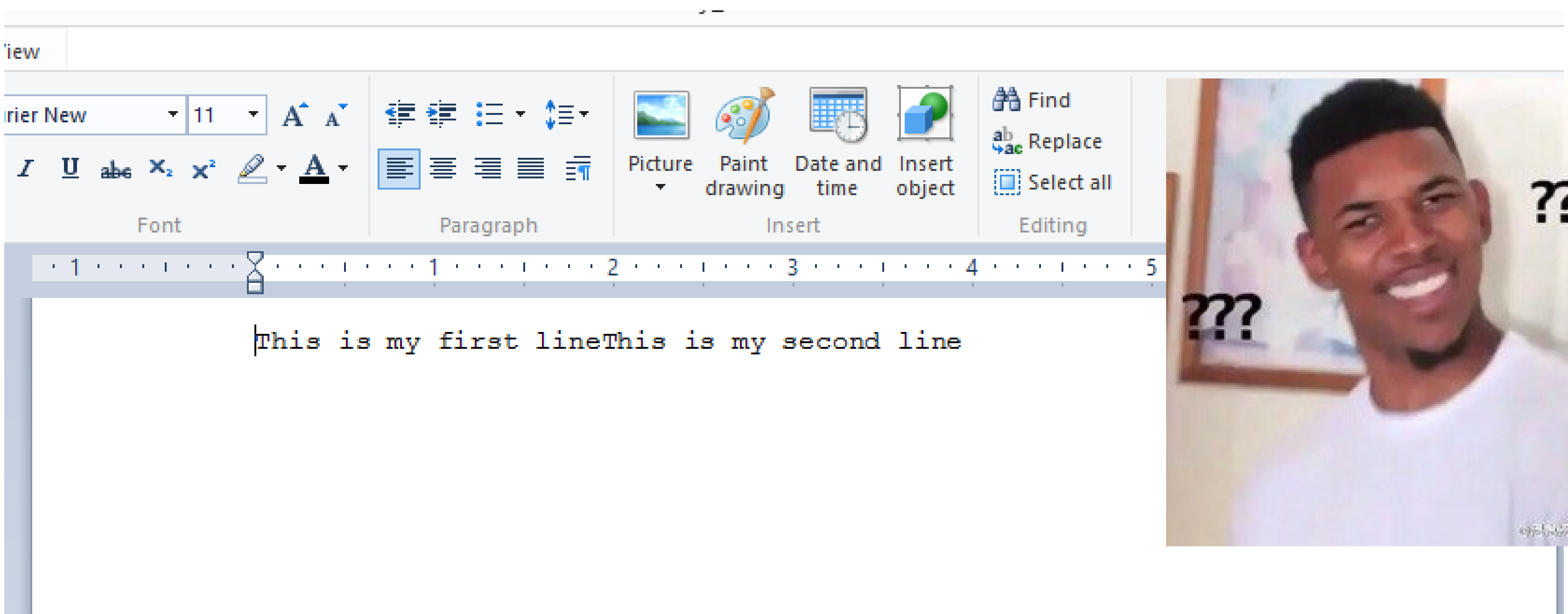
```
def write_something():
    with open('my_file.txt','w') as f:
        f.write('This is my first line')
        f.write('This is my second line')

write_something()
```

The file object that we called it “f” (can be any variable name)

Use the file “f” to write something in it

Writing A File

```
def write_something():
    with open('my_file.txt','w') as f:
        f.write('This is my first line')
        f.write('This is my second line')

write_something()
```

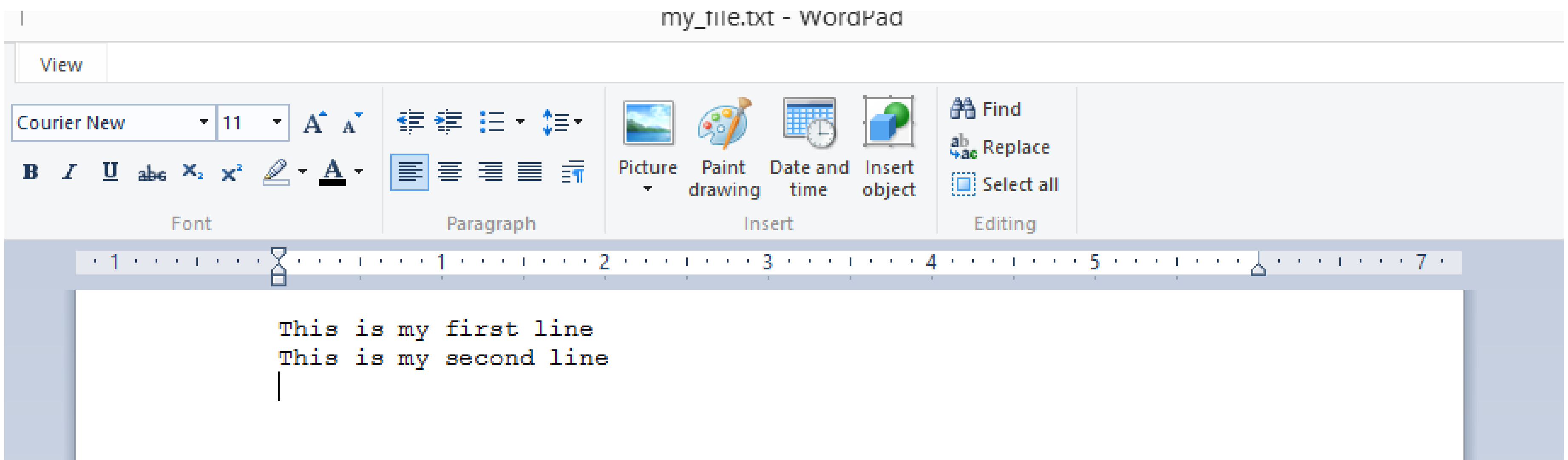


```
This is my first lineThis is my second line
```

Writing a File

The newline character

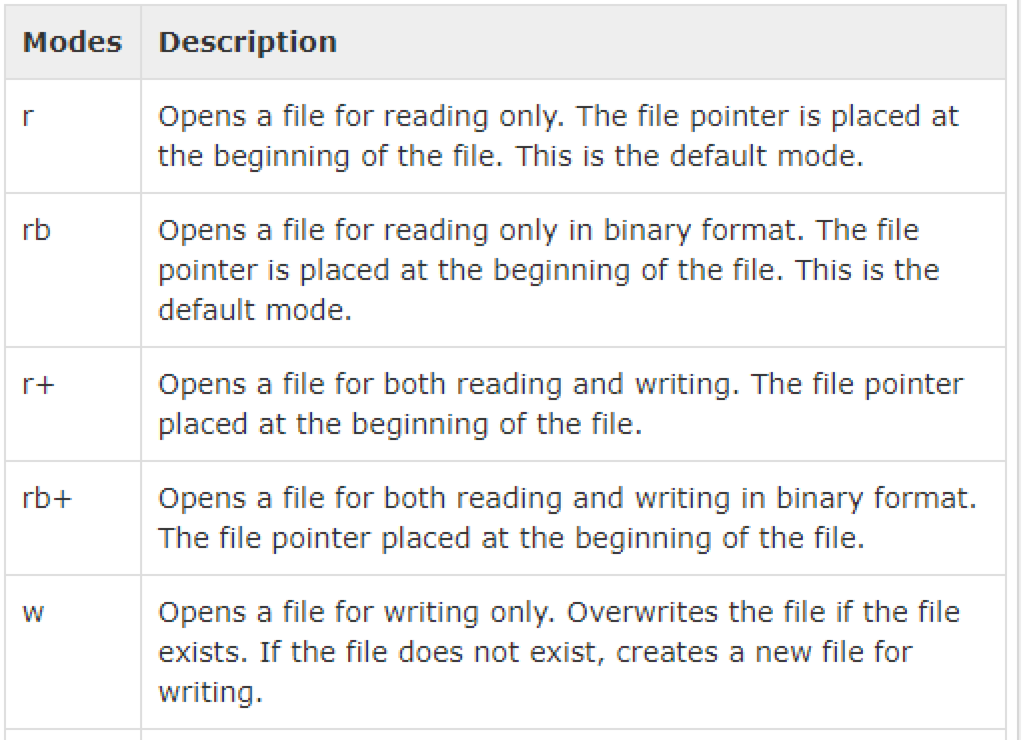
```
def write_something():
    with open('my_file.txt','w') as f:
        f.write('This is my first line'+'\n')
        f.write('This is my second line'+'\n')

write_something()
```



```
This is my first line
This is my second line
```

Different File Opening Modes

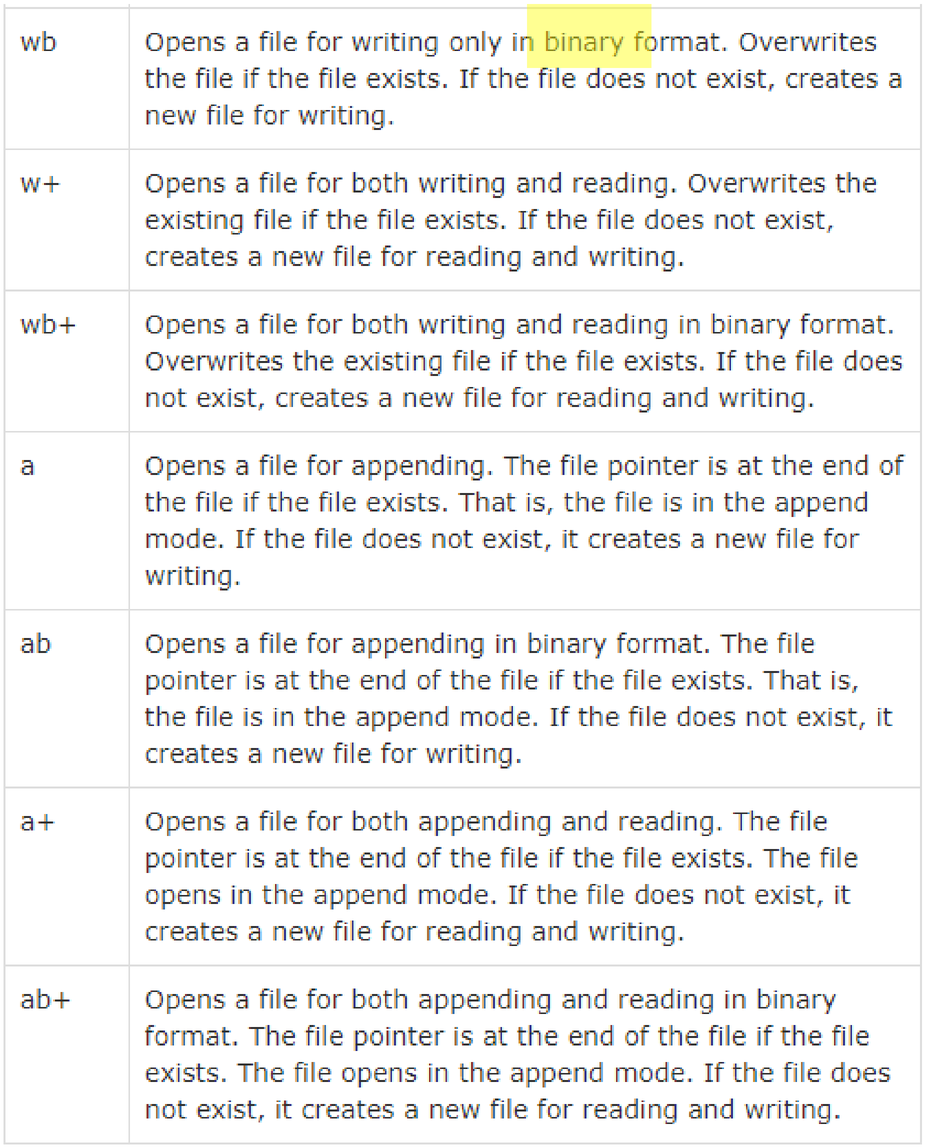
```
def write_something():
    with open('my_file.txt','w') as f:
        f.write('This is my first line'+'\n')
        f.write('This is my second line'+'\n')

write_something()
```



Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
r+	Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
rb+	Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.

Different File Opening Modes

- Default is text format
- Storing in text mode is very space consuming
- E.g. storing the date ‘20180901’
  - Text (ASCII):
    - `50 48 49 56 48 57 48 49`
  - Binary (Integer):
    - `01 33 EF A5`



wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

Reading a File

Try it out

- I have a text file called “student_marks.txt”

file name

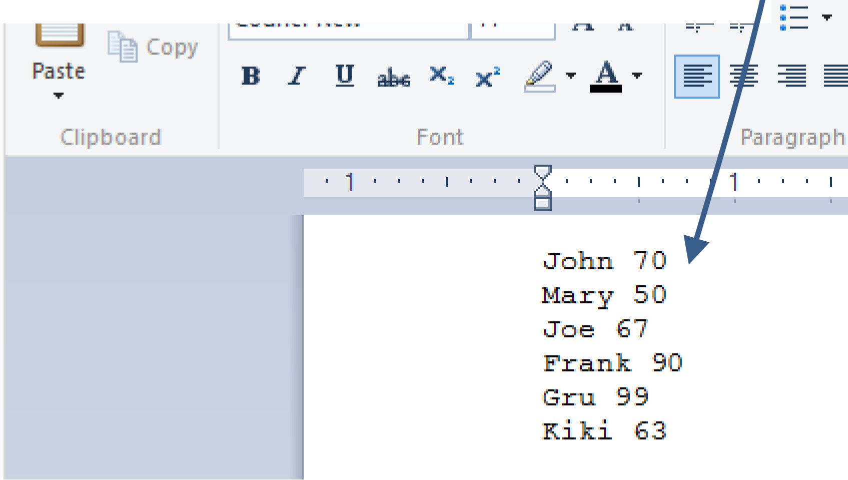
The file object

```
>>> with open('student_marks.txt') as f:
        data = f.read()
```

read the **whole** file into “data” as a string

```
>>> data
'John 70\nMary 50\nJoe 67\nFrank 90\nGru 99\nKiki 63'
>>>
```

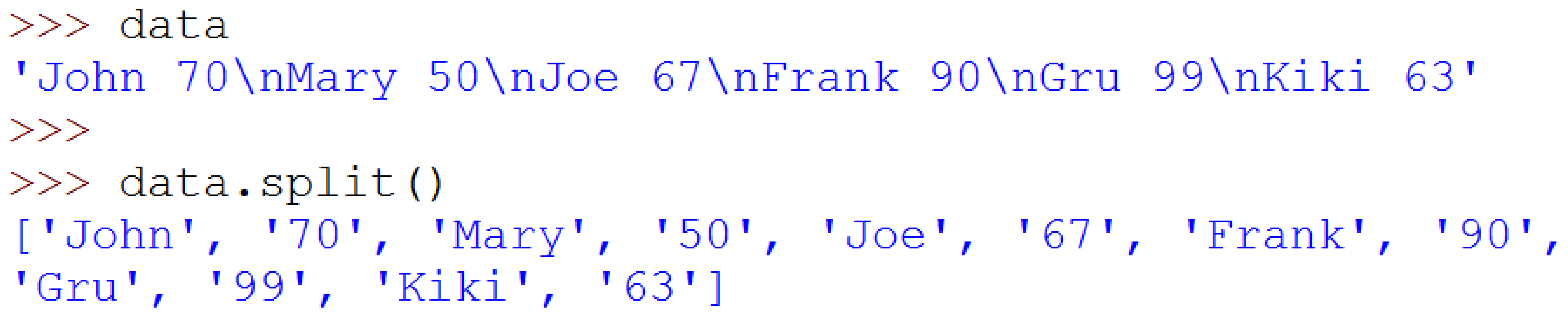
The new line character



```
John 70
Mary 50
Joe 67
Frank 90
Gru 99
Kiki 63
```

String Operation Split

- Use the function split to separate the string into a list of strings by a separator



```
>>> data
'John 70\nMary 50\nJoe 67\nFrank 90\nGru 99\nKiki 63'
>>> 
>>> data.split()
['John', '70', 'Mary', '50', 'Joe', '67', 'Frank', '90',
'Gru', '99', 'Kiki', '63']
```

- If you do not put any argument for split(), the default separators are space and newline

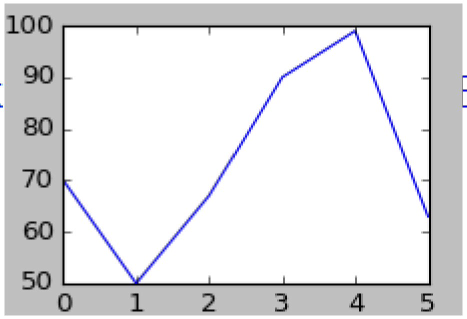
Try it out

Starting from the second position and step two

- Extract all the scores

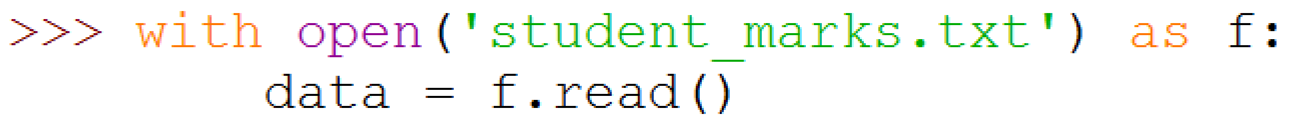
```
>>> data.split()
['John', '70', 'Mary', '50', 'Joe', '67', 'Frank', '90',
 'Gru', '99', 'Kiki', '63']
>>> max(data.split())
'Mary'
>>> all_score = [int(i) for i in data.split()[1::2]]
>>> all_score
[70, 50, 67, 90, 99, 63]
>>> max(all_score)
99
>>> plt.plot(all_score)
[<matplotlib.lines.Line2D object at 0x              B0>]
>>> plt.show()
```

Convert each string into an integer



Reading One Whole File into a String

- That’ is not “healthy”
- Your file can be a few MB or even GB



```
>>> with open('student_marks.txt') as f:
        data = f.read()
```

- Then this line of code will run in a very long time, may even end in crashing the whole program or even the system
- Better way to do is to read the file line-by-line

Reading the File Line-by-line

```
def read_line_by_line():
    with open('student_marks.txt','r') as f:
        for a_line in f:
            print(a_line)
```

```
John 70

Mary 50

Joe 67

Frank 90

Gru 99

Kiki 63


>>>
```

The file type is also "iterable"!!!

If you do it in Python Shell (bad)

```
>>> with open('student_marks.txt') as f:
        for a_line in f:
            a_line

'John 70\n'
'Mary 50\n'
'Joe 67\n'
'Frank 90\n'
'Gru 99\n'
'Kiki 63'
```

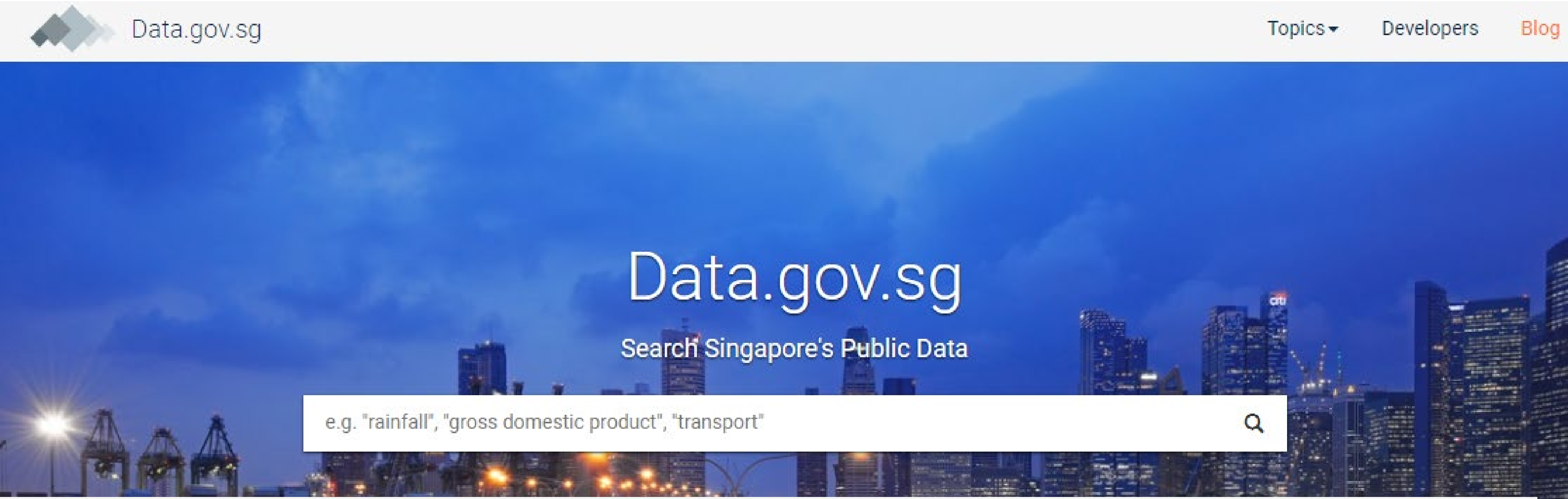
The file as an "iterable"

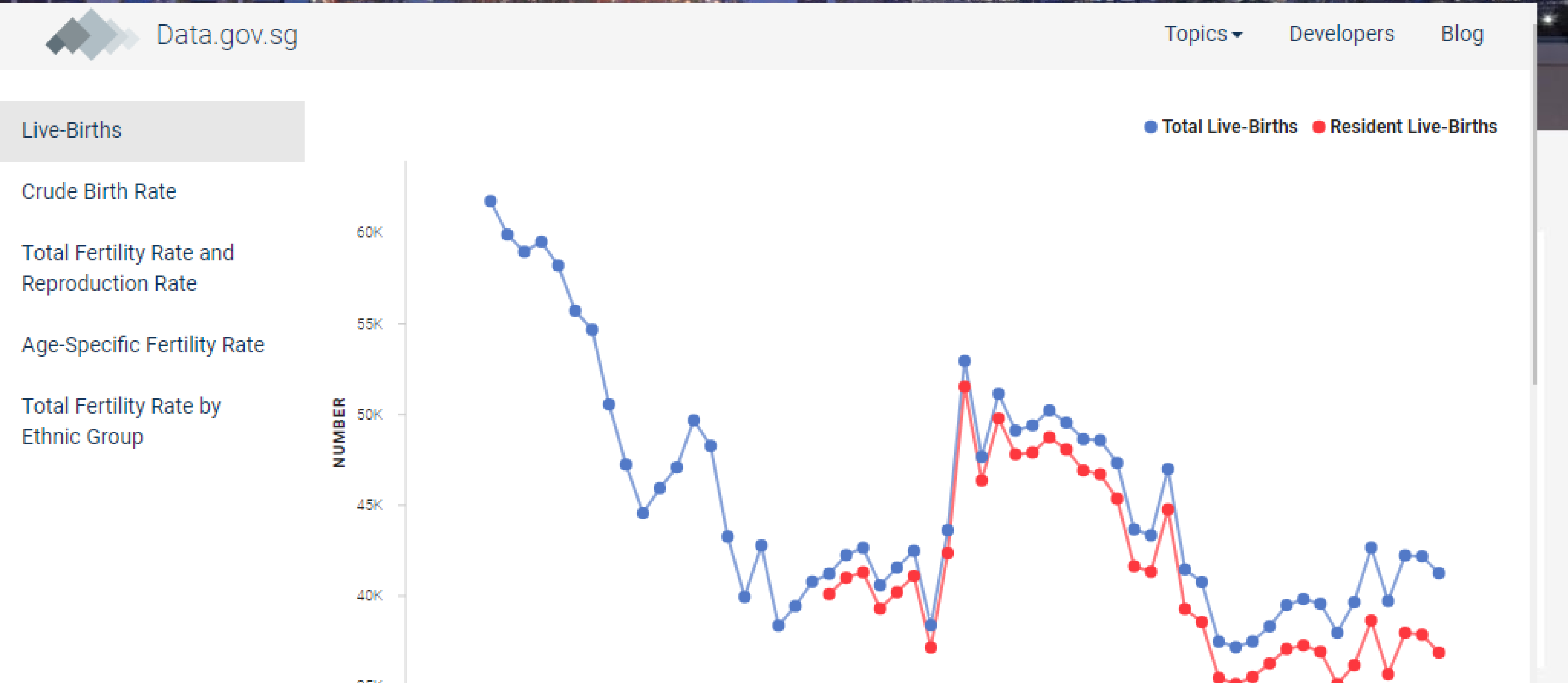
console echo

Annoying newline character '\n'

- How should we deal with these "\n"?

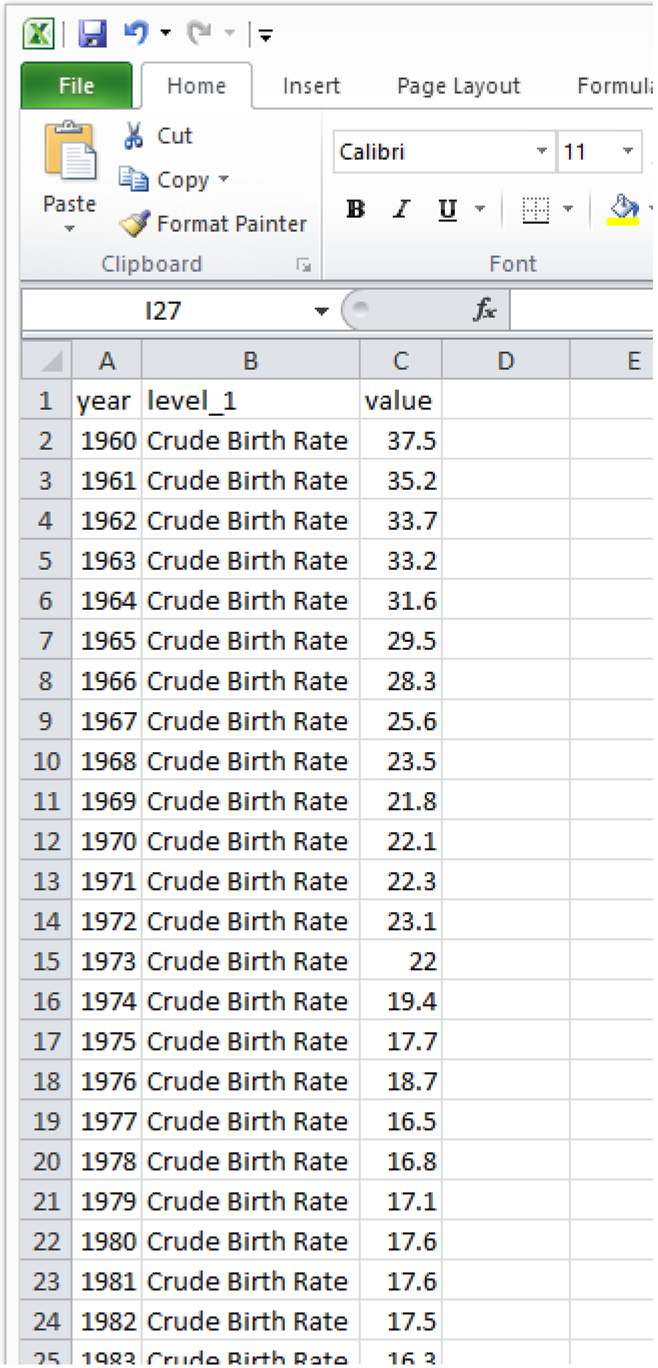
A More Complicated Example


Data.gov.sg
Topics
Developers
Blog
Data.gov.sg
Search Singapore's Public Data
e.g. "rainfall", "gross domestic product", "transport"

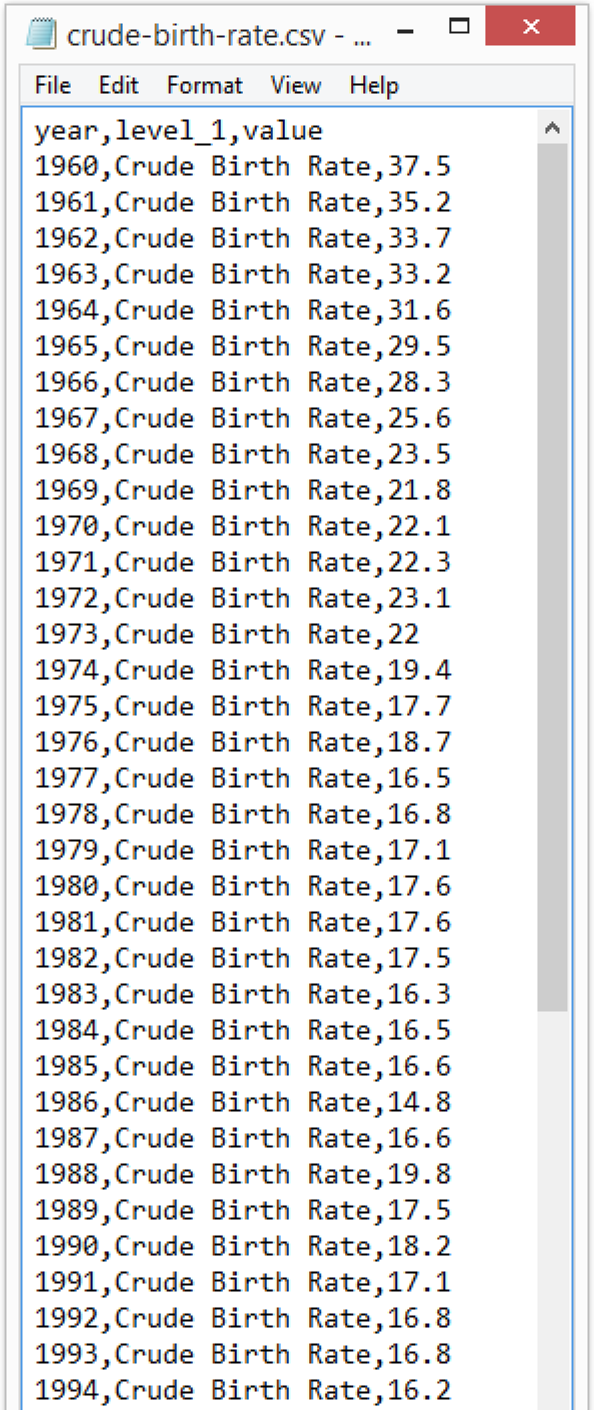

Data.gov.sg
Topics
Developers
Blog
Live-Births
Crude Birth Rate
Total Fertility Rate and Reproduction Rate
Age-Specific Fertility Rate
Total Fertility Rate by Ethnic Group
● Total Live-Births
● Resident Live-Births
NUMBER
60K
55K
50K
45K
40K

Open in

- Excel
- Notepad



	A	B	C	D	E
1	year	level_1	value		
2	1960	Crude Birth Rate	37.5		
3	1961	Crude Birth Rate	35.2		
4	1962	Crude Birth Rate	33.7		
5	1963	Crude Birth Rate	33.2		
6	1964	Crude Birth Rate	31.6		
7	1965	Crude Birth Rate	29.5		
8	1966	Crude Birth Rate	28.3		
9	1967	Crude Birth Rate	25.6		
10	1968	Crude Birth Rate	23.5		
11	1969	Crude Birth Rate	21.8		
12	1970	Crude Birth Rate	22.1		
13	1971	Crude Birth Rate	22.3		
14	1972	Crude Birth Rate	23.1		
15	1973	Crude Birth Rate	22		
16	1974	Crude Birth Rate	19.4		
17	1975	Crude Birth Rate	17.7		
18	1976	Crude Birth Rate	18.7		
19	1977	Crude Birth Rate	16.5		
20	1978	Crude Birth Rate	16.8		
21	1979	Crude Birth Rate	17.1		
22	1980	Crude Birth Rate	17.6		
23	1981	Crude Birth Rate	17.6		
24	1982	Crude Birth Rate	17.5		
25	1983	Crude Birth Rate	16.3		



```
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
1969,Crude Birth Rate,21.8
1970,Crude Birth Rate,22.1
1971,Crude Birth Rate,22.3
1972,Crude Birth Rate,23.1
1973,Crude Birth Rate,22
1974,Crude Birth Rate,19.4
1975,Crude Birth Rate,17.7
1976,Crude Birth Rate,18.7
1977,Crude Birth Rate,16.5
1978,Crude Birth Rate,16.8
1979,Crude Birth Rate,17.1
1980,Crude Birth Rate,17.6
1981,Crude Birth Rate,17.6
1982,Crude Birth Rate,17.5
1983,Crude Birth Rate,16.3
1984,Crude Birth Rate,16.5
1985,Crude Birth Rate,16.6
1986,Crude Birth Rate,14.8
1987,Crude Birth Rate,16.6
1988,Crude Birth Rate,19.8
1989,Crude Birth Rate,17.5
1990,Crude Birth Rate,18.2
1991,Crude Birth Rate,17.1
1992,Crude Birth Rate,16.8
1993,Crude Birth Rate,16.8
1994,Crude Birth Rate,16.2
```

Reading Data in Python

- You can start reading a file in Python by

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


'year,level_1,value\n'
'1960,Crude Birth Rate,37.5\n'
'1961,Crude Birth Rate,35.2\n'
```

- The line is read with a ‘\n’ (newline)

Reading Data

```
>>> with open('crude-birth-rate.csv') as f:
        line1 = f.readline()
        line2 = f.readline()
        line3 = f.readline()
        print(line1)
        print(line2)
        print(line3)
```

extra new line because of '\n'

```
year,level_1,value

1960,Crude Birth Rate,37.5

1961,Crude Birth Rate,35.2

```

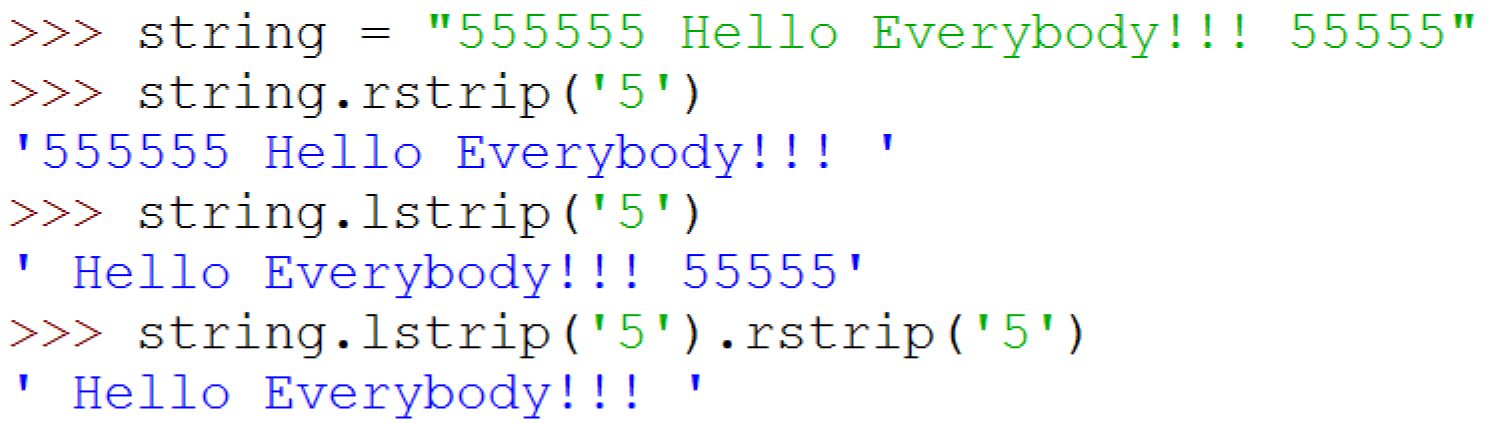
rstrip(): Strip Characters on the Right

```
>>> with open('crude-birth-rate.csv') as f:
        line1 = f.readline().rstrip('\n')
        line2 = f.readline().rstrip('\n')
        line3 = f.readline().rstrip('\n')
        print(line1)
        print(line2)
        print(line3)
```

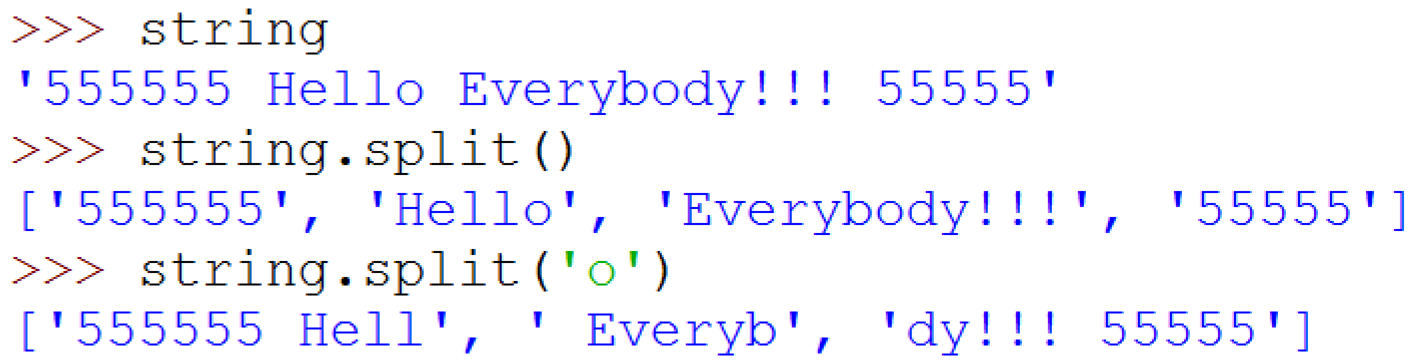
no more extra new line

```
year,level_1,value
1960,Crude Birth Rate,37.5
1961,Crude Birth Rate,35.2
>>>
>>>
>>>
```

String rstrip() and split()



```
>>> string = "555555 Hello Everybody!!! 55555"
>>> string.rstrip('5')
'555555 Hello Everybody!!! '
>>> string.lstrip('5')
' Hello Everybody!!! 55555'
>>> string.lstrip('5').rstrip('5')
' Hello Everybody!!! '
```



```
>>> string
'555555 Hello Everybody!!! 55555'
>>> string.split()
['555555', 'Hello', 'Everybody!!!', '55555']
>>> string.split('o')
['555555 Hell', ' Everyb', 'dy!!! 55555']
```

Let’s start writing the code

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

        for line in f:
            print(line.rstrip('\n'))
```

- The file object ‘f’ is an iterable
- Every iteration you have a hidden

```
line = f.readline()
```

Let’s start writing the code

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

        for line in f:
            print(line.rstrip('\n'))
```

```
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
1969,Crude Birth Rate,21.8
1970,Crude Birth Rate,22.1
1971,Crude Birth Rate,22.3
```

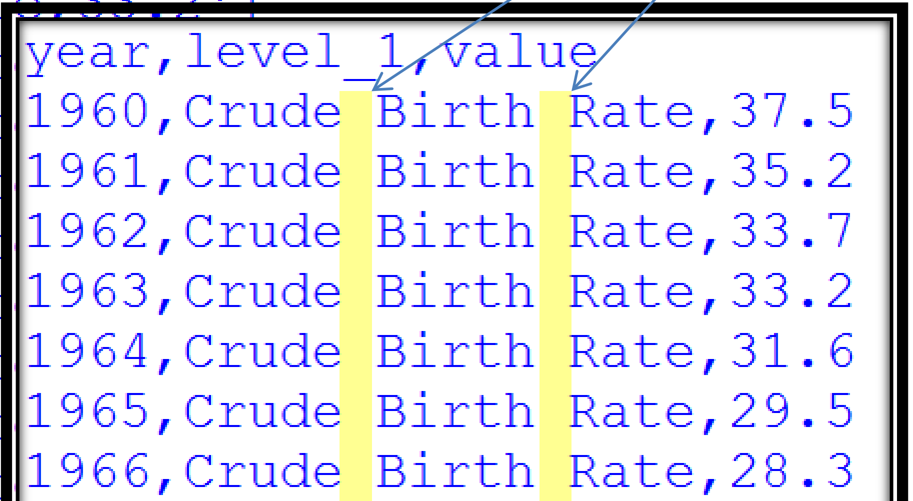
A string

Let’s Split!

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

        for line in f:
            print(line.rstrip('\n').split())
```

```
['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', 'Rat
               e', 'Birth', 'Rat
               e', 'Birth', 'Rat
               e', 'Birth', 'Rat
               e', 'Birth', 'Rat
               e', 'Birth', 'Rat
               e', 'Birth', 'Rat
               e', 'Birth', 'Rat
```

Split by space!!!



```
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
```

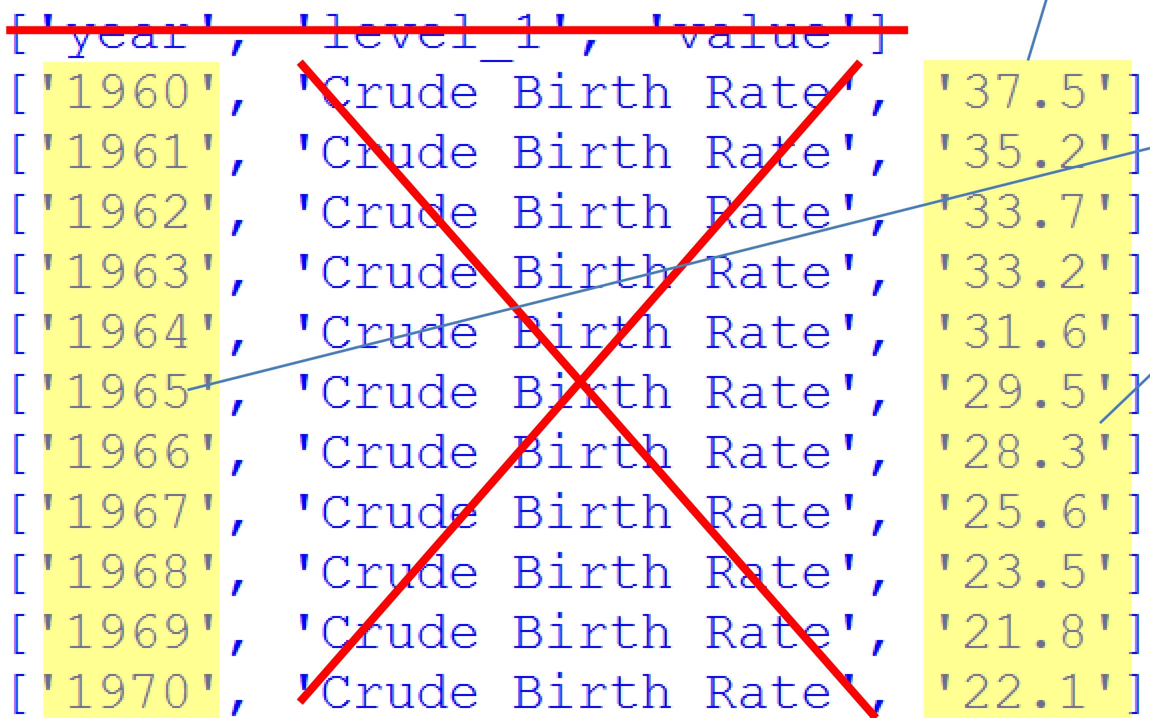
Let’s Split Commas!

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

        for line in f:
            print(line.rstrip('\n').split(','))
```

```
['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']
['1969', 'Crude Birth Rate', '21.8']
['1970', 'Crude Birth Rate', '22.1']
```

Let’s manage our data

- We want to plot the year against the birthday
  - The value of the birth rate is x per thousand
  - So the actual no. of birth is x times 1000



```
~~['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']
['1969', 'Crude Birth Rate', '21.8']
['1970', 'Crude Birth Rate', '22.1']
```

Convert
into integers

```
import matplotlib.pyplot as plt

def plot_birth_rate():
    with open('crude-birth-rate.csv') as f:
        f.readline()


        for line in f:
            list_form = line.rstrip('\n').split(',')
```

Discard the first line

```
['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']
```

"line"

```python
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)
```

Discard the first line

```
~~['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']
```

“line”

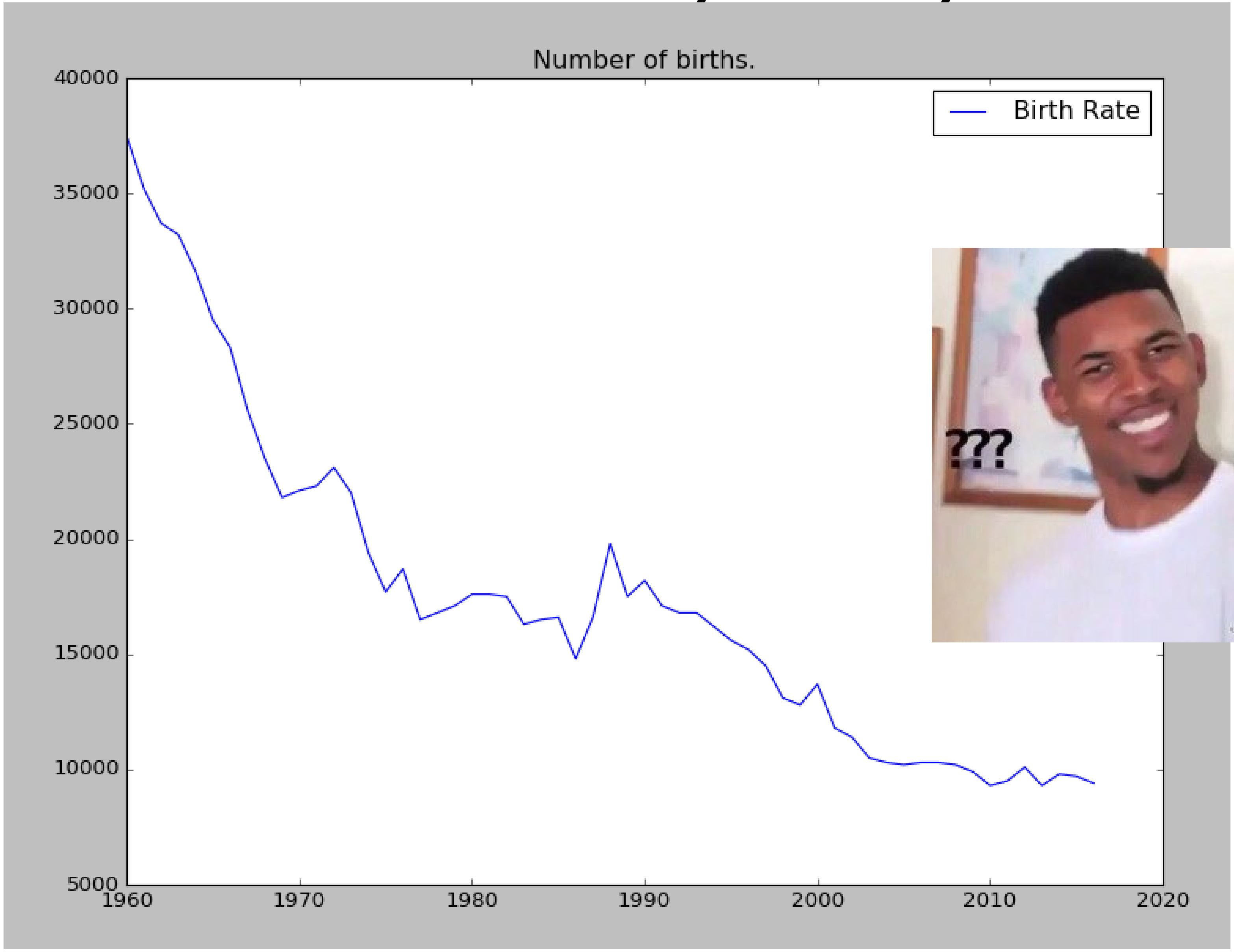
```python
import matplotlib.pyplot as plt

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()
```

Discard the first line

~~['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']

“line”

Now You Know Why “Baby Bonus”



Reading CSV Files

Read in a CSV file into a list

Create a CSV File Reader

```
>>> from pprint import pprint
>>> birth_file = open('crude-birth-rate.csv')
>>> birth_file_reader = csv.reader(birth_file)
>>> birth_data = list(birth_file_reader)
>>> pprint(birth_data)
[['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'],
 ['1969', 'Crude Birth Rate', '21.8'],
```

Remember these four lines of code

No need for all those string strip(), split() etc.

Today

- You have learned how to read and write a file
  - Or more precisely, reading or writing a general file
  - In fact, we got an easier way to read a CSV file
    - Wait until we learn multi-dimensional arrays
- You can say that you “finished” the (most of the) “core” Python Language
- The rest is extra packages, features