

File Input/Output





Data.gov.sg

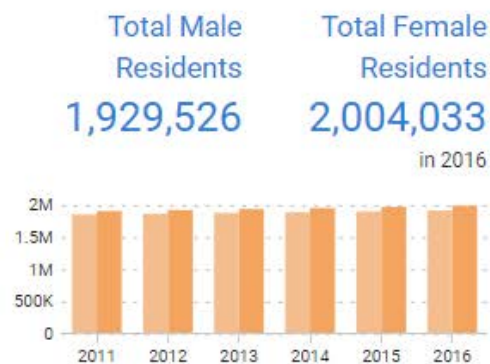
Search Singapore's Public Data

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

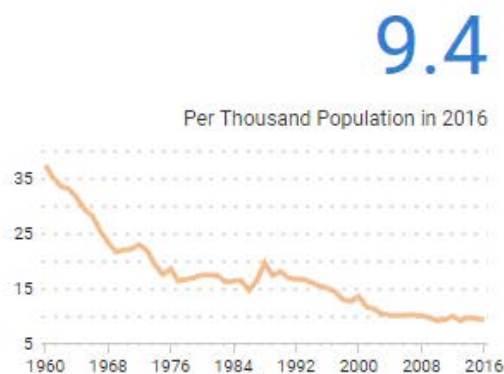


Singapore at a glance

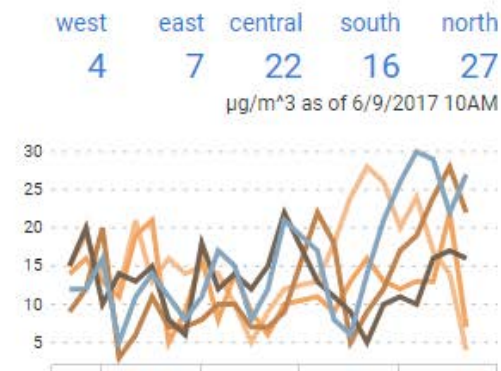
Singapore Residents By Gender, End June,
Annual - Data



Crude Birth Rate - Data



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

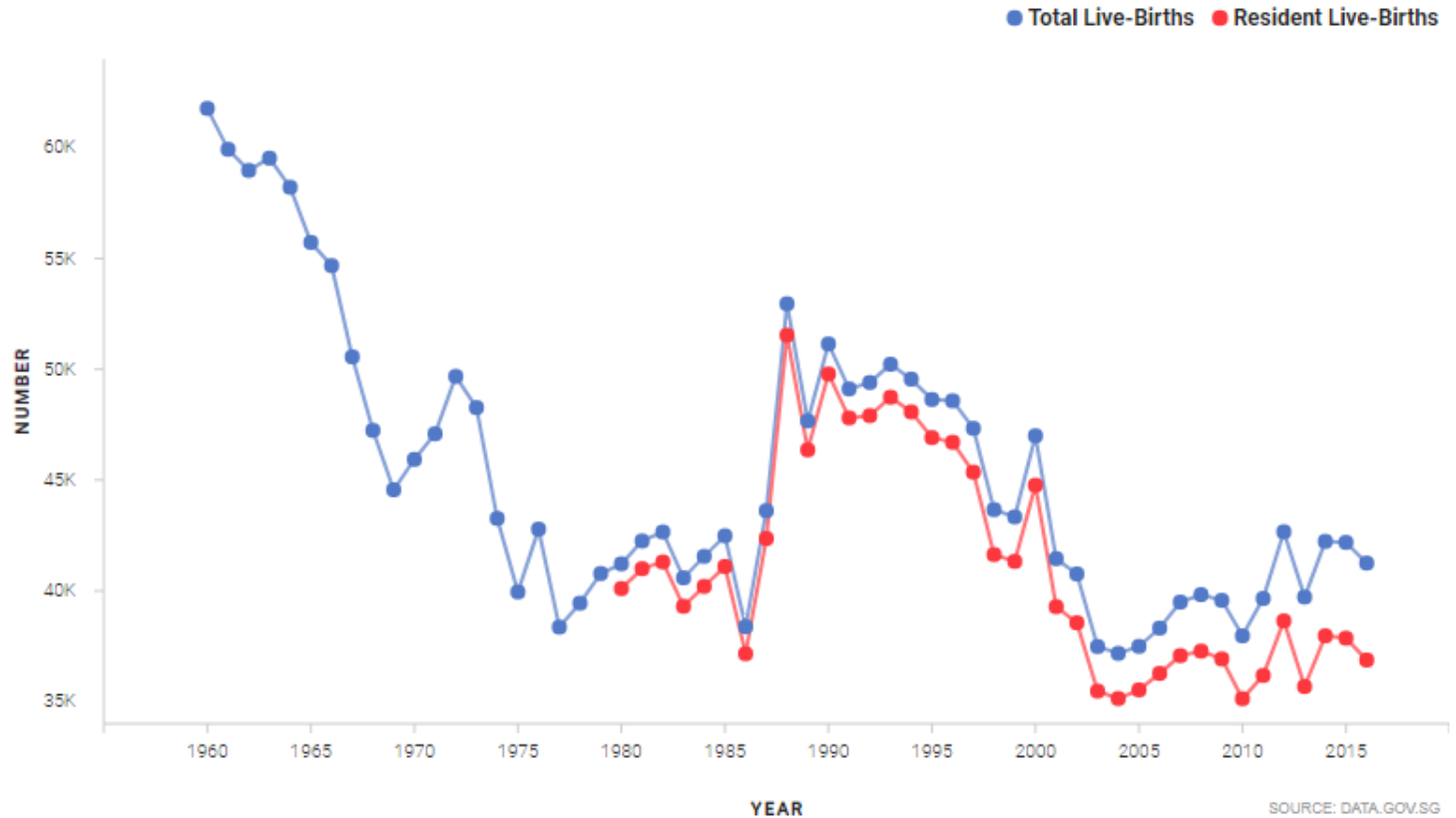


Live-Births

Crude Birth Rate

Total Fertility Rate and
Reproduction Rate

Age-Specific Fertility Rate

Total Fertility Rate by
Ethnic Group

Live-Births ▾

Chart ▾

Embed View

Births and Fertility, Annual

Managed by Ministry of Trade and Industry - Department of Statistics

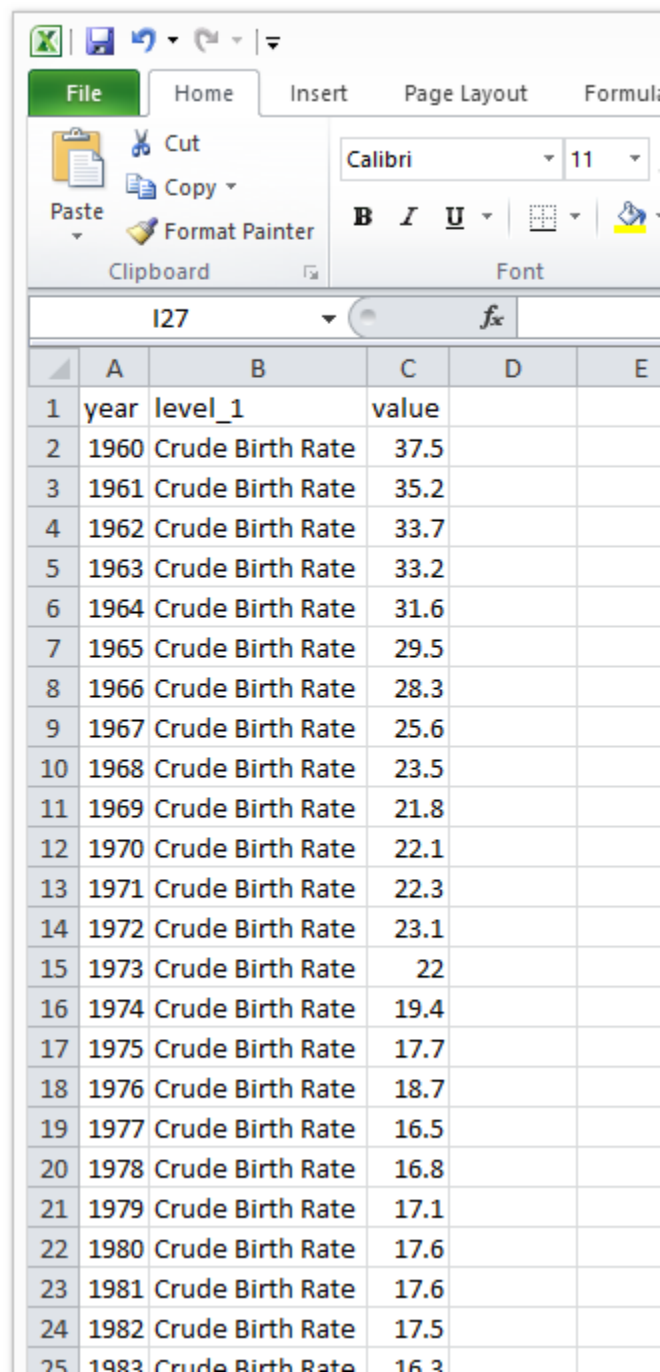
A summary of key indicators measuring births and fertility in Singapore.

Download

You can
download any
data!

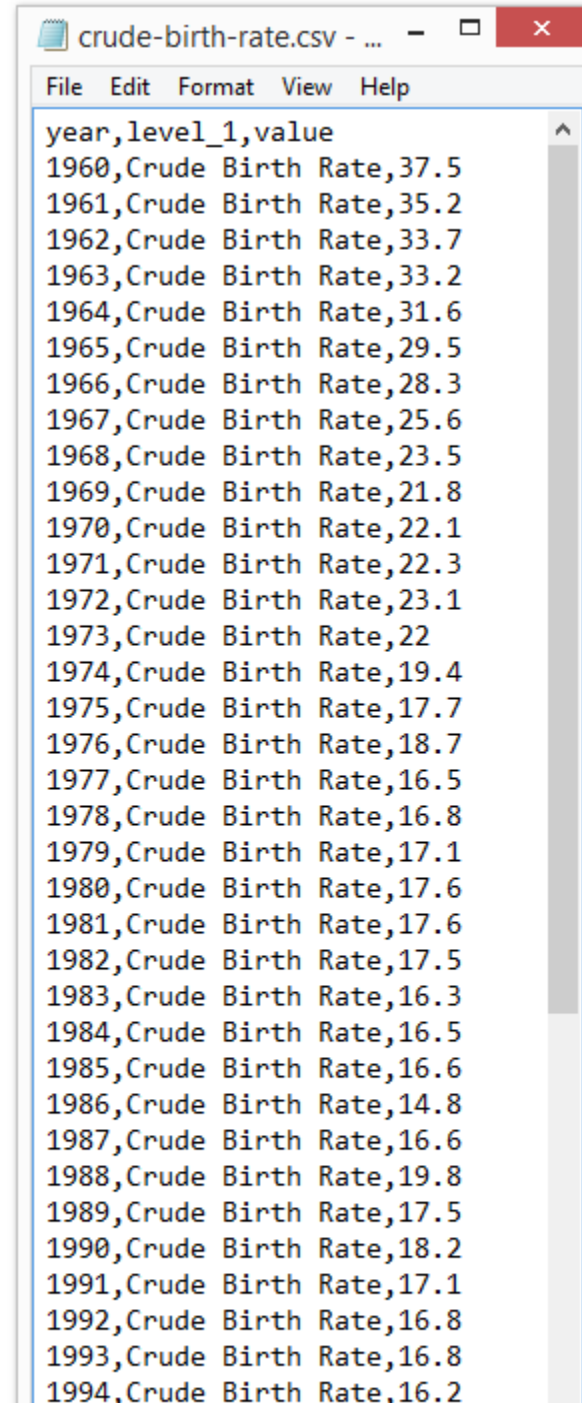
Open in

- Excel
- Notepad



The screenshot shows the Microsoft Excel interface with the 'File' tab selected. The worksheet contains a table with the following data:

	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	16.2		



The screenshot shows a Notepad window titled 'crude-birth-rate.csv'. The text content is as follows:

```
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?
- Usually practice
 - Data file +
 - Python code that can read the file



An Easier Example

Try it out

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

file name

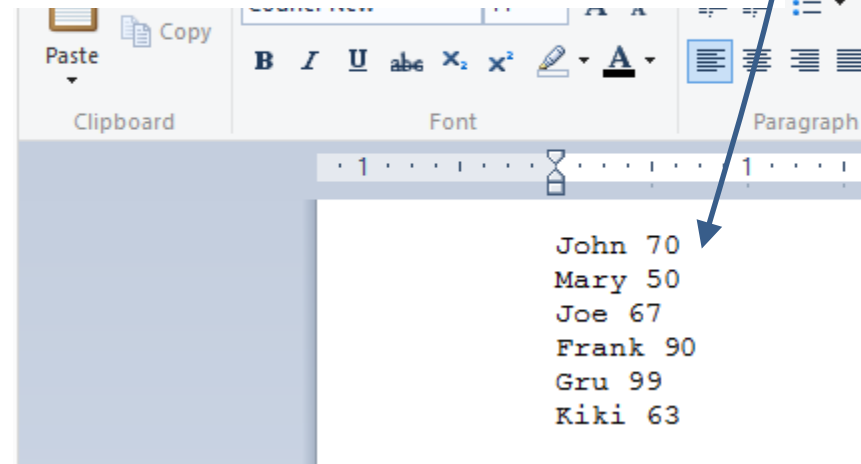
The file object

```
>>> 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



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

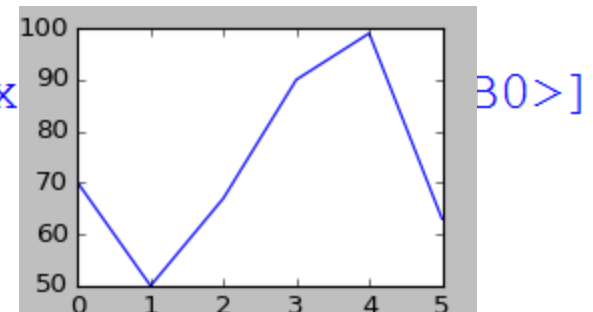
Try it out

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...30>]
>>> plt.show()
```

Convert each string into an integer



Or you can

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

The file as an
“iterable”

console echo

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

Annoying newline
character '\n'

A More Complicated Example

Data.gov.sg

Search Singapore's Public Data

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



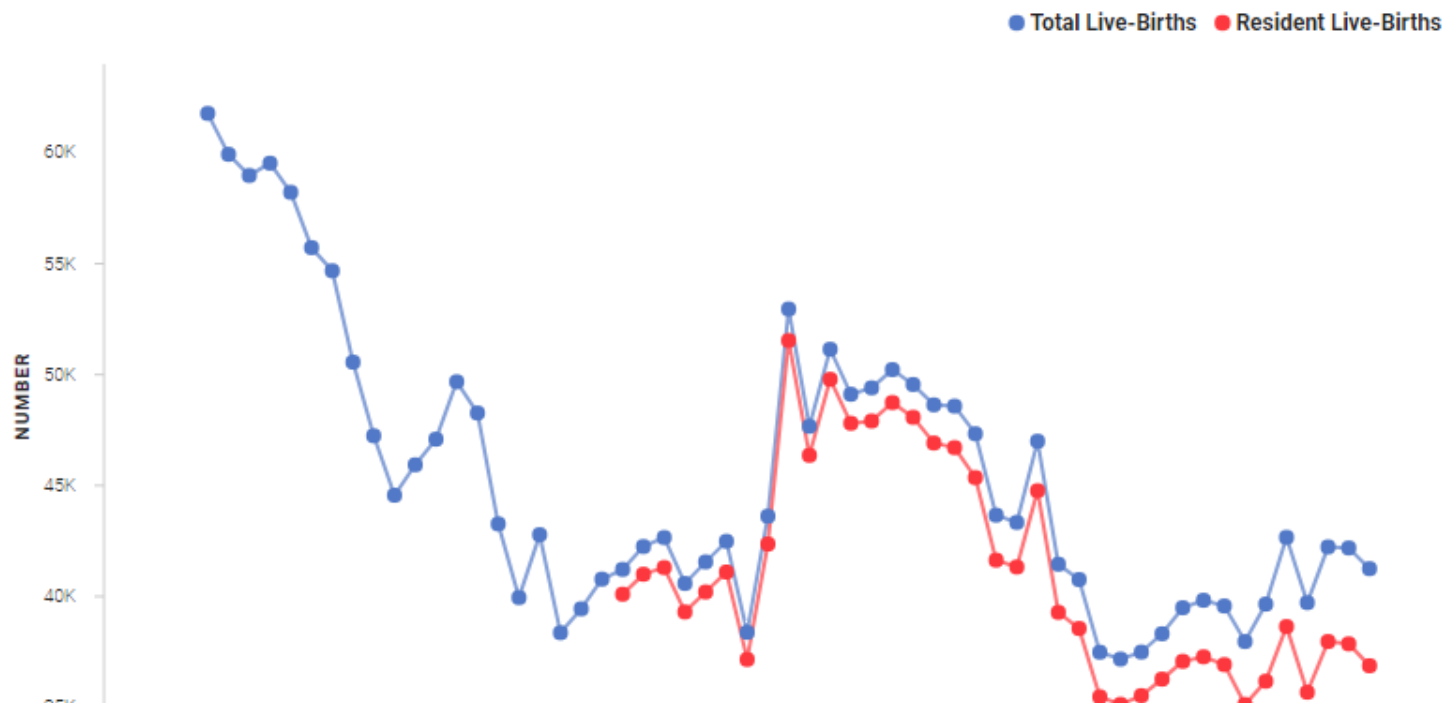
Live-Births

Crude Birth Rate

Total Fertility Rate and
Reproduction Rate

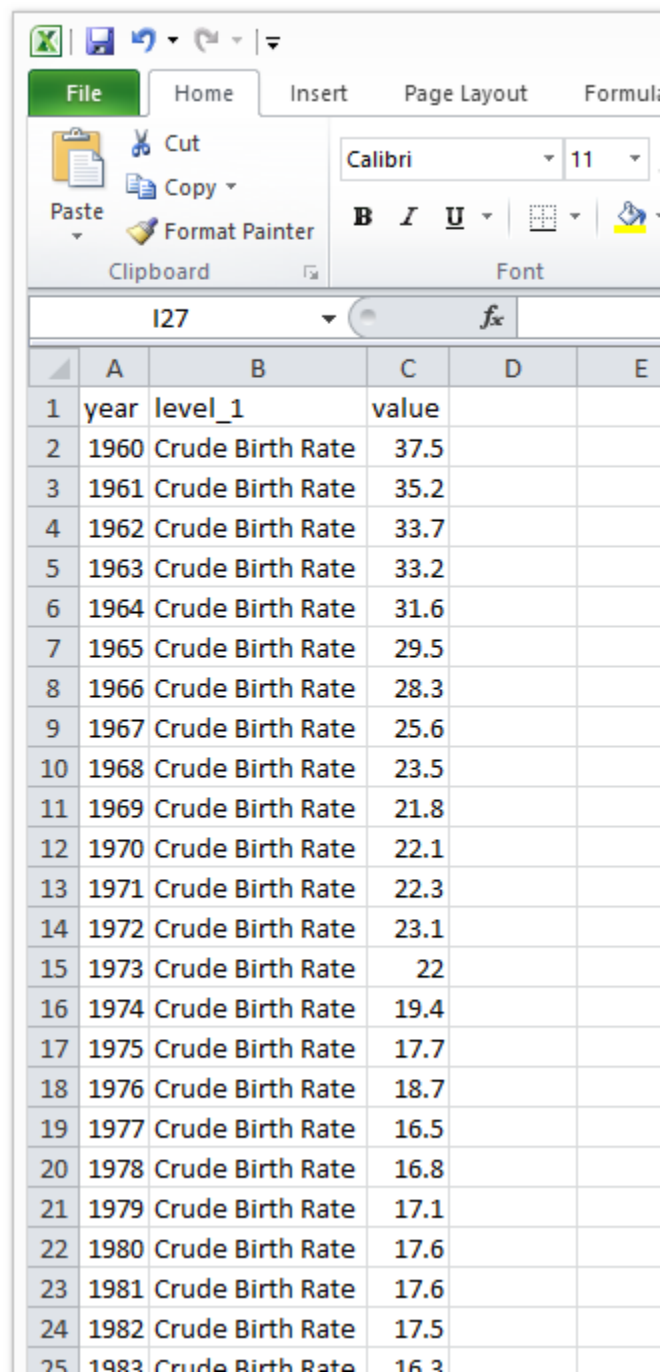
Age-Specific Fertility Rate

Total Fertility Rate by
Ethnic Group



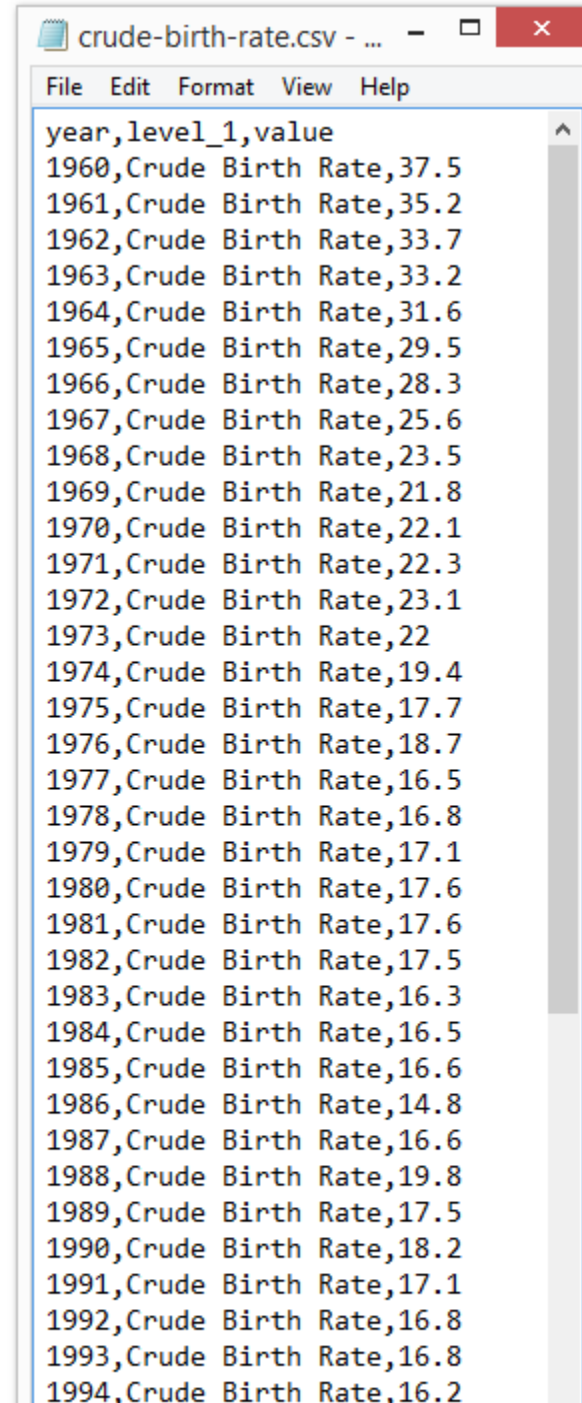
Open in

- Excel
- Notepad



The screenshot shows the Microsoft Excel interface. The 'File' tab is selected in the ribbon. The 'Clipboard' group contains icons for Cut, Copy, Paste, and Format Painter. The 'Font' group shows the font set to Calibri, size 11, with bold, italic, and underline options. The formula bar shows the active cell is I27. The worksheet contains a table with the following data:

	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	16.2		




The screenshot shows a Notepad window titled 'crude-birth-rate.csv'. The text inside is a CSV representation of the data from the Excel spreadsheet:

```
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,16.2
```

Reading Data in Python

- You can start reading a file in Python by

```
'year,level_1,value\n'
>>> 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

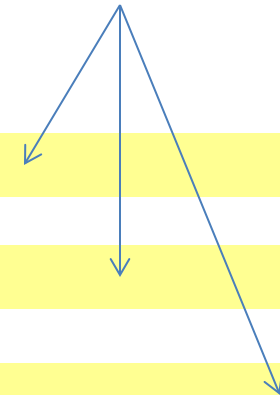
```
>>> with open('crude-birth-rate.csv') as f:
    line1 = f.readline()
    line2 = f.readline()
    line3 = f.readline()
    print(line1)
    print(line2)
    print(line3)
```

year,level_1,value

1960,Crude Birth Rate,37.5

1961,Crude Birth Rate,35.2

extra new line because
of '\n'

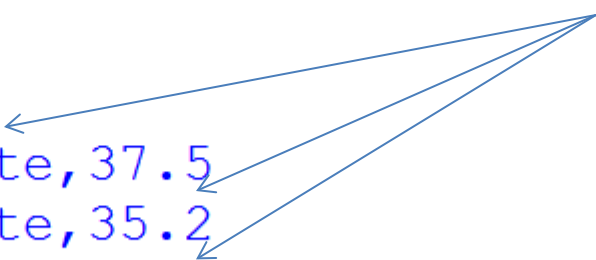


rstrip(): Strip Characters on the Right

```
>>> 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
```



```
>>>
>>>
>>>
```


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'))
```

```
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', 'Rate,31.6']  
['1965,Crude', 'Birth', 'Rate,29.5']  
['1966,Crude', 'Birth', 'Rate,28.3']
```

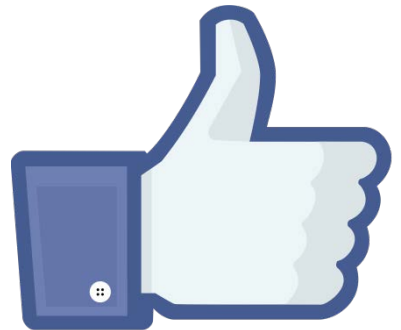
Split by space!!!



Let's Split Commas!

```
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()  ← Discard the first line
        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()
```

['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” →

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

```
def plot_birth_rate():
```

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

```
        f.readline() ← Discard the first line
```

```
        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()
```

```
['year', 'level 1', 'value']
```

```
['1960', 'Crude Birth Rate', '37.5']
```

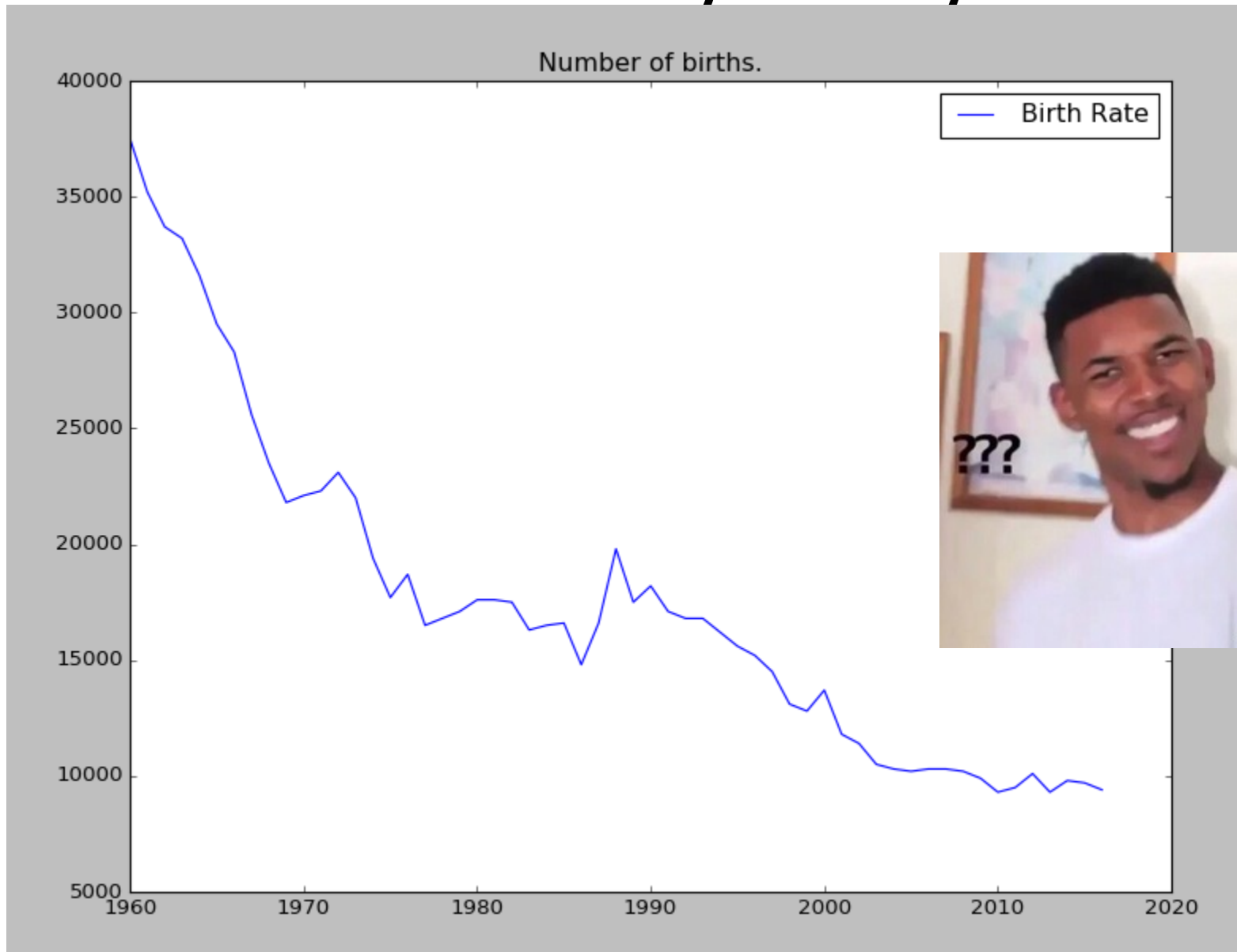
```
['1961', 'Crude Birth Rate', '35.2']
```

```
['1962', 'Crude Birth Rate', '33.7']
```

```
['1963', 'Crude Birth Rate', '33.2']
```

"line"

Now You Know Why “Baby Bonus”



Writing A File

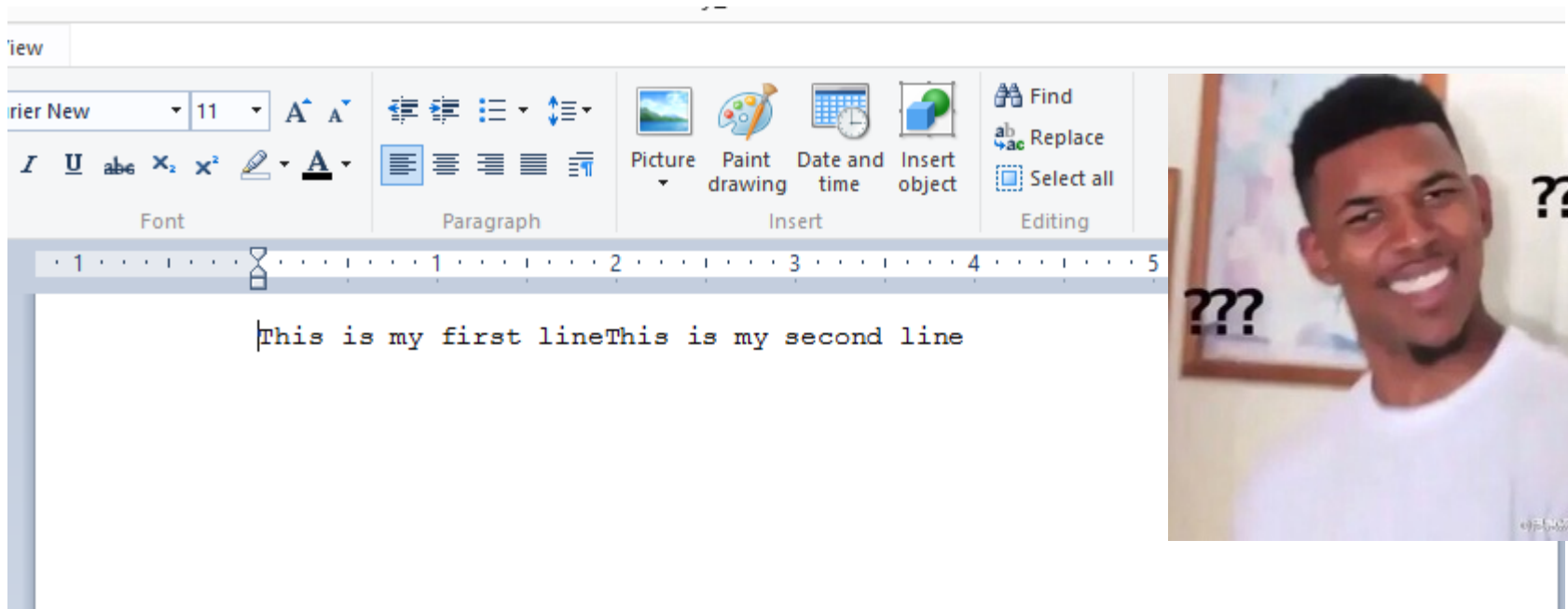
Actually Easier

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')
```

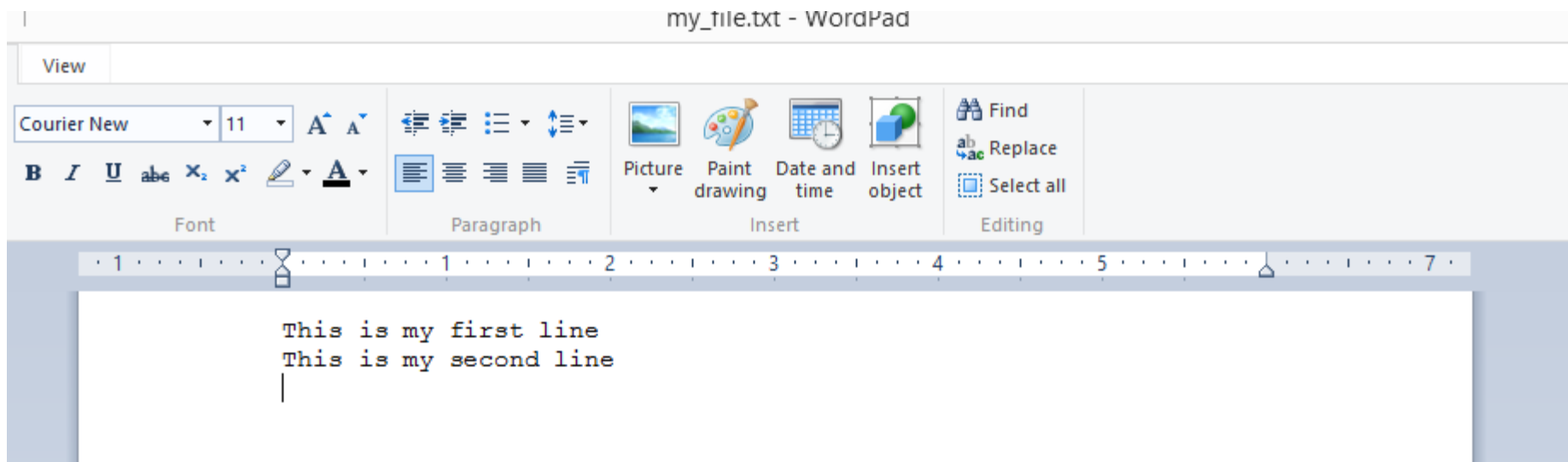
```
write_something()  
|
```



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()
```



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

- Storing in text mode is very space consuming
- E.g. storing the year '2017'
 - Text:
 - 32 30 31 37
 - Binary (Integer):
 - 07 E1

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.

Today

- You have learned how to read and write a file
- You can say that you “finished” the “core” Python Language
- The rest is extra packages, features