File input / output

In this section you will learn how to read and write ASCII data (pure text data).

Content:

- Open
- Close
- Read
- Write
- Read CSV file
- Write CSV file

Before we begin it should be clear what a file is. A file is like a sheet of paper with a header, data in the middle and the line at the end. The header contains the file name, its size and type, and more. The data is what ever you are working with. The end of file line contains in most cases a specual character telling the system 'here is the end'.

Depending on the system you are working on the lines end with different special characters which are not displayed (most editors provide the function 'show invisible characters').

```
Windows:        CR+LF which is equal to \r\n
Linux and Mac:  LF which is euqal \n
```

Open a file for reading

To open the file with the Python `open` function is the first step. It returns a file object that allows us to use finctions for reading and writing.

Reference:

```
open(file, mode='r', newline=None)
```

A file can be opend for reading or writing ASCII or binary data by setting the mode in the function call.

Modes:

```
'r'       open for reading (default, text mode)
'w'       open for writing, overwriting an existing file
(text mode)
'b'       binary
't'       text (default)
'rb'      read binary
'wb'      write binary
'a'       appending at the end of the file
'+'       reading and writing
```

Note: For text files, the system depending line endings (\n on Unix, \r\n on Windows) will be converted to \n when reading, and when writing it will be converted back to the system specific line ending. Nice, you don't have to care about it.

Now, we want to open the file World_Nations_population_land_area.txt which is stored in the material/data folder in read only mode. Because it is a text file (ASCII) we can use the short mode 'r' instead of 'rt'.

```
f = open('../data/World_Nations_population_land_area.txt', 'r')

print(f)
```

In [ ]:
```
f = open('../data/World_Nations_population_land_area.txt', 'r')

print(f)
```

```
<_io.TextIOWrapper name='../data/World_Nations_population_land_area.txt'
mode='r' encoding='UTF-8'>
```

Close a file

After the work is done you can close the file object with the `close` function.

```
f.close()
```

You can control whether a file is closed or not with the `closed` method.

```
f.closed
```

In [ ]:
```
f.close()
f.closed
```

Out[ ]:
```
True
```

If you try to close an already closed file nothing will happen (return value None).

In [ ]:
```
f.close()
```

Exercise

What happens when you try to read from an closed file? This is only about the message you get, the read function will be introduced in the next section.

Enter

```
f.read()
```

In [ ]:
```
f = open('../data/World_Nations_population_land_area.txt', 'r')
foo = f.read()
f.close()
```

In [ ]:
```
type(foo)
```

Out[ ]:
```
str
```

Reading functions

The Python standard library provides a few functions to read data from a file.

1. read()
2. readline()
3. readlines()

1. read()

The function `read` can be used to read the content of the file object. It reads the complete file content as a string at once.

The following code prints the complete content of the input file

```
f = open('../data/World_Nations_population_land_area.txt',
'r')

data = f.read()
print(data)
```

Output:

```
# Country       Population (2020)       Land Area (Km²)
Density(P/Km²)
1    Afghanistan    38928346    652860    60
2    Albania    2877797    27400    105
3    Algeria    43851044    2381740    18
4    Andorra    77265    470    164
5    Angola    32866272    1246700    26
6    Antigua and Barbuda    97929    440    223
7    Argentina    45195774    2736690    17
8    Armenia    2963243    28470    104
9    Australia    25499884    7682300    3
10    Austria    9006398    82409    109
...
195     Zimbabwe        14862924        386850  38```
```

In [ ]:
```
f = open('../data/World_Nations_population_land_area.txt', 'r')

data = f.read()
print(data)
```

If you iterate over the `read()` output it would iterate over every character in the file.

For example, retrieve the size of the content list and print the first 120 characters in content:

```
f = open('../data/World_Nations_population_land_area.txt', 'r')

content = f.read()

print(len(content))

print(content[:120])
```

Output:

```
7521
# Country       Population (2020)       Land Area (Km²)
Density(P/Km²)
1       Afghanistan     38928346        652860  60
2       Albania 2877797 27400   105
```

In [ ]:
```
f = open('../data/World_Nations_population_land_area.txt', 'r')

content = f.read()

print(len(content))

print(content[:120])
f.close()
```

```
6110
# Country       Population(2020)        Land Area (Km²) Density(P/Km²)
1       Afghanistan     38928346        652860  60
2       Albania 2877797 27400   105
3
```

A better way to program is to use the `with` method because at the end the file will be closed automatically.

```
with open('../data/World_Nations_population_land_area.txt', 'r')
as f:
    print(f.read())
```

Output:

```
# Country       Population (2020)       Land Area (Km²)
Density(P/Km²)
# Country       Population(2020)        Land Area (Km²)
Density(P/Km²)
1       Afghanistan     38928346        652860  60
2       Albania 2877797 27400   105
3       Algeria 43851044        2381740 18
4       Andorra 77265   470     164
5       Angola  32866272        1246700 26
6       Antigua and Barbuda     97929   440     223
7       Argentina       45195774        2736690 17
8       Armenia 2963243 28470   104
9       Australia       25499884        7682300 3
10      Austria 9006398 82409   109
...
195     Zimbabwe        14862924        386850  38
```

In [ ]:
```
with open('../data/World_Nations_population_land_area.txt', 'r') as fh:
    print(fh.read())
fh.closed
```

```
# Country       Population(2020)        Land Area (Km²) Density(P/Km²)
1       Afghanistan     38928346        652860  60
2       Albania 2877797 27400   105
3       Algeria 43851044        2381740 18
4       Andorra 77265   470     164
5       Angola  32866272        1246700 26
6       Antigua and Barbuda     97929   440     223
7       Argentina       45195774        2736690 17
8       Armenia 2963243 28470   104
9       Australia       25499884        7682300 3
10      Austria 9006398 82409   109
11      Azerbaijan      10139177        82658   123
12      Bahamas 393244  10010   39
13      Bahrain 1701575 760     2239
14      Bangladesh      164689383       130170  1265
15      Barbados        287375  430     668
16      Belarus 9449323 202910  47
17      Belgium 11589623        30280   383
18      Belize  397628  22810   17
19      Benin   12123200        112760  108
20      Bhutan  771608  38117   20
21      Bolivia 11673021        1083300 11
22      Bosnia and Herzegovina  3280819 51000   64
23      Botswana        2351627 566730  4
24      Brazil  212559417       8358140 25
25      Brunei  437479  5270    83
26      Bulgaria        6948445 108560  64
27      Burkina Faso    20903273        273600  76
28      Burundi 11890784        25680   463
29      Côte d'Ivoire   26378274        318000  83
30      Cabo Verde      555987  4030    138
31      Cambodia        16718965        176520  95
32      Cameroon        26545863        472710  56
33      Canada  37742154        9093510 4
34      Central African Republic        4829767 622980  8
35      Chad    16425864        1259200 13
36      Chile   19116201        743532  26
37      China   1439323776      9388211 153
38      Colombia        50882891        1109500 46
39      Comoros 869601  1861    467
40      Congo   5518087 341500  16
41      Costa Rica      5094118 51060   100
42      Croatia 4105267 55960   73
43      Cuba    11326616        106440  106
44      Cyprus  1207359 9240    131
45      Czechia 10708981        77240   139
46      Democratic Republic of the Congo        89561403        2267050 4
0
47      Denmark 5792202 42430   137
48      Djibouti        988000  23180   43
49      Dominica        71986   750     96
50      Dominican Republic      10847910        48320   225
51      Ecuador 17643054        248360  71
52      Egypt   102334404       995450  103
53      El Salvador     6486205 20720   313
54      Equatorial Guinea       1402985 28050   50
55      Eritrea 3546421 101000  35
56      Estonia 1326535 42390   31
57      Eswatini        1160164 17200   67
58      Ethiopia        114963588       1000000 115
```

```
59      Fiji    896445  18270   49
60      Finland 5540720 303890  18
61      France  65273511        547557  119
62      Gabon   2225734 257670  9
63      Gambia  2416668 10120   239
64      Georgia 3989167 69490   57
65      Germany 83783942        348560  240
66      Ghana   31072940        227540  137
67      Greece  10423054        128900  81
68      Grenada 112523  340     331
69      Guatemala       17915568        107160  167
70      Guinea  13132795        245720  53
71      Guinea-Bissau   1968001 28120   70
72      Guyana  786552  196850  4
73      Haiti   11402528        27560   414
74      Holy See        801     0       2003
75      Honduras        9904607 111890  89
76      Hungary 9660351 90530   107
77      Iceland 341243  100250  3
78      India   1380004385      2973190 464
79      Indonesia       273523615       1811570 151
80      Iran    83992949        1628550 52
81      Iraq    40222493        434320  93
82      Ireland 4937786 68890   72
83      Israel  8655535 21640   400
84      Italy   60461826        294140  206
85      Jamaica 2961167 10830   273
86      Japan   126476461       364555  347
87      Jordan  10203134        88780   115
88      Kazakhstan      18776707        2699700 7
89      Kenya   53771296        569140  94
90      Kiribati        119449  810     147
91      Kuwait  4270571 17820   240
92      Kyrgyzstan      6524195 191800  34
93      Laos    7275560 230800  32
94      Latvia  1886198 62200   30
95      Lebanon 6825445 10230   667
96      Lesotho 2142249 30360   71
97      Liberia 5057681 96320   53
98      Libya   6871292 1759540 4
99      Liechtenstein   38128   160     238
100     Lithuania       2722289 62674   43
101     Luxembourg      625978  2590    242
102     Madagascar      27691018        581795  48
103     Malawi  19129952        94280   203
104     Malaysia        32365999        328550  99
105     Maldives        540544  300     1802
106     Mali    20250833        1220190 17
107     Malta   441543  320     1380
108     Marshall Islands        59190   180     329
109     Mauritania      4649658 1030700 5
110     Mauritius       1271768 2030    626
111     Mexico  128932753       1943950 66
112     Micronesia      548914  700     784
113     Moldova 4033963 32850   123
114     Monaco  39242   1       26337
115     Mongolia        3278290 1553560 2
116     Montenegro      628066  13450   47
117     Morocco 36910560        446300  83
118     Mozambique      31255435        786380  40
```

```
119     Myanmar 54409800        653290  83
120     Namibia 2540905 823290  3
121     Nauru   10824   20      541
122     Nepal   29136808        143350  203
123     Netherlands     17134872        33720   508
124     New Zealand     4822233 263310  18
125     Nicaragua       6624554 120340  55
126     Niger   24206644        1266700 19
127     Nigeria 206139589       910770  226
128     North Korea     25778816        120410  214
129     North Macedonia 2083374 25220   83
130     Norway  5421241 365268  15
131     Oman    5106626 309500  16
132     Pakistan        220892340       770880  287
133     Palau   18094   460     39
134     Palestine State 5101414 6020    847
135     Panama  4314767 74340   58
136     Papua New Guinea        8947024 452860  20
137     Paraguay        7132538 397300  18
138     Peru    32971854        1280000 26
139     Philippines     109581078       298170  368
140     Poland  37846611        306230  124
141     Portugal        10196709        91590   111
142     Qatar   2881053 11610   248
143     Romania 19237691        230170  84
144     Russia  145934462       16376870        9
145     Rwanda  12952218        24670   525
146     Saint Kitts and Nevis   53199   260     205
147     Saint Lucia     183627  610     301
148     Saint Vincent and the Grenadines        110940  390     284
149     Samoa   198414  2830    70
150     San Marino      33931   60      566
151     Sao Tome and Principe   219159  960     228
152     Saudi Arabia    34813871        2149690 16
153     Senegal 16743927        192530  87
154     Serbia  8737371 87460   100
155     Seychelles      98347   460     214
156     Sierra Leone    7976983 72180   111
157     Singapore       5850342 700     8358
158     Slovakia        5459642 48088   114
159     Slovenia        2078938 20140   103
160     Solomon Islands 686884  27990   25
161     Somalia 15893222        627340  25
162     South Africa    59308690        1213090 49
163     South Korea     51269185        97230   527
164     South Sudan     11193725        610952  18
165     Spain   46754778        498800  94
166     Sri Lanka       21413249        62710   341
167     Sudan   43849260        1765048 25
168     Suriname        586632  156000  4
169     Sweden  10099265        410340  25
170     Switzerland     8654622 39516   219
171     Syria   17500658        183630  95
172     Tajikistan      9537645 139960  68
173     Tanzania        59734218        885800  67
174     Thailand        69799978        510890  137
175     Timor-Leste     1318445 14870   89
176     Togo    8278724 54390   152
177     Tonga   105695  720     147
178     Trinidad and Tobago     1399488 5130    273
```

```
179     Tunisia 11818619        155360  76
180     Turkey  84339067        769630  110
181     Turkmenistan    6031200 469930  13
182     Tuvalu  11792   30      393
183     Uganda  45741007        199810  229
184     Ukraine 43733762        579320  75
185     United Arab Emirates    9890402 83600   118
186     United Kingdom  67886011        241930  281
187     United States of America        331002651       9147420 36
188     Uruguay 3473730 175020  20
189     Uzbekistan      33469203        425400  79
190     Vanuatu 307145  12190   25
191     Venezuela       28435940        882050  32
192     Vietnam 97338579        310070  314
193     Yemen   29825964        527970  56
194     Zambia  18383955        743390  25
195     Zimbabwe        14862924        386850  38
```

Out[ ]: True

In [ ]:
```
with open('../data/World_Nations_population_land_area.txt', 'r') as f:
    print(f.read())
```

2. readline()

The function `readline` reads the file content line by line. For demonstration purposes we use here the method without `with` and at the end, we have to `close` the file manually.

Open the file again:

```
f = open('../data/World_Nations_population_land_area.txt', 'r')
```

Read the first line and print it:

```
line = f.readline()

print(line)
```

Output:

```
# Country       Population (2020)       Land Area (Km²)
Density(P/Km²)
```

Read and print next line.

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

Output:

```
2       Albania         2,877,797       27,400  105
```

And so on. A loop over the needed lines would be a better way.

Exercise

Print the first 10 countries from the file to stdout.

Hint: Don't forget to close the file.

In [ ]:
```
f = open('../data/World_Nations_population_land_area.txt', 'r')

for i in range(11):
    print(f.readline())
```

```
# Country       Population(2020)        Land Area (Km²) Density(P/Km²)

1       Afghanistan     38928346        652860  60

2       Albania 2877797 27400   105

3       Algeria 43851044        2381740 18

4       Andorra 77265   470     164

5       Angola  32866272        1246700 26

6       Antigua and Barbuda     97929   440     223

7       Argentina       45195774        2736690 17

8       Armenia 2963243 28470   104

9       Australia       25499884        7682300 3

10      Austria 9006398 82409   109
```

In [ ]:
```
f = open('../data/World_Nations_population_land_area.txt', 'r')

for i in range(250):
    print(f.readline())
```

#	Country	Population(2020)	Land Area (Km²)	Density(P/Km²)
1	Afghanistan	38928346	652860	60
2	Albania	2877797	27400	105
3	Algeria	43851044	2381740	18
4	Andorra	77265	470	164
5	Angola	32866272	1246700	26
6	Antigua and Barbuda	97929	440	223
7	Argentina	45195774	2736690	17
8	Armenia	2963243	28470	104
9	Australia	25499884	7682300	3
10	Austria	9006398	82409	109
11	Azerbaijan	10139177	82658	123
12	Bahamas	393244	10010	39
13	Bahrain	1701575	760	2239
14	Bangladesh	164689383	130170	1265
15	Barbados	287375	430	668
16	Belarus	9449323	202910	47
17	Belgium	11589623	30280	383
18	Belize	397628	22810	17
19	Benin	12123200	112760	108
20	Bhutan	771608	38117	20
21	Bolivia	11673021	1083300	11
22	Bosnia and Herzegovina	3280819	51000	64
23	Botswana	2351627	566730	4
24	Brazil	212559417	8358140	25
25	Brunei	437479	5270	83
26	Bulgaria	6948445	108560	64
27	Burkina Faso	20903273	273600	76
28	Burundi	11890784	25680	463
29	Côte d'Ivoire	26378274	318000	83

```
30      Cabo Verde      555987  4030    138

31      Cambodia        16718965        176520  95

32      Cameroon        26545863        472710  56

33      Canada  37742154        9093510 4

34      Central African Republic        4829767 622980  8

35      Chad    16425864        1259200 13

36      Chile   19116201        743532  26

37      China   1439323776      9388211 153

38      Colombia        50882891        1109500 46

39      Comoros 869601  1861    467

40      Congo   5518087 341500  16

41      Costa Rica      5094118 51060   100

42      Croatia 4105267 55960   73

43      Cuba    11326616        106440  106

44      Cyprus  1207359 9240    131

45      Czechia 10708981        77240   139

46      Democratic Republic of the Congo        89561403        2267050 4
0

47      Denmark 5792202 42430   137

48      Djibouti        988000  23180   43

49      Dominica        71986   750     96

50      Dominican Republic      10847910        48320   225

51      Ecuador 17643054        248360  71

52      Egypt   102334404       995450  103

53      El Salvador     6486205 20720   313

54      Equatorial Guinea       1402985 28050   50

55      Eritrea 3546421 101000  35

56      Estonia 1326535 42390   31

57      Eswatini        1160164 17200   67

58      Ethiopia        114963588       1000000 115

59      Fiji    896445  18270   49
```

```
60      Finland 5540720 303890  18
61      France  65273511        547557  119
62      Gabon   2225734 257670  9
63      Gambia  2416668 10120   239
64      Georgia 3989167 69490   57
65      Germany 83783942        348560  240
66      Ghana   31072940        227540  137
67      Greece  10423054        128900  81
68      Grenada 112523  340     331
69      Guatemala       17915568        107160  167
70      Guinea  13132795        245720  53
71      Guinea-Bissau   1968001 28120   70
72      Guyana  786552  196850  4
73      Haiti   11402528        27560   414
74      Holy See        801     0       2003
75      Honduras        9904607 111890  89
76      Hungary 9660351 90530   107
77      Iceland 341243  100250  3
78      India   1380004385      2973190 464
79      Indonesia       273523615       1811570 151
80      Iran    83992949        1628550 52
81      Iraq    40222493        434320  93
82      Ireland 4937786 68890   72
83      Israel  8655535 21640   400
84      Italy   60461826        294140  206
85      Jamaica 2961167 10830   273
86      Japan   126476461       364555  347
87      Jordan  10203134        88780   115
88      Kazakhstan      18776707        2699700 7
89      Kenya   53771296        569140  94
```

```
90      Kiribati        119449  810     147
91      Kuwait  4270571 17820   240
92      Kyrgyzstan      6524195 191800  34
93      Laos    7275560 230800  32
94      Latvia  1886198 62200   30
95      Lebanon 6825445 10230   667
96      Lesotho 2142249 30360   71
97      Liberia 5057681 96320   53
98      Libya   6871292 1759540 4
99      Liechtenstein   38128   160     238
100     Lithuania       2722289 62674   43
101     Luxembourg      625978  2590    242
102     Madagascar      27691018        581795  48
103     Malawi  19129952        94280   203
104     Malaysia        32365999        328550  99
105     Maldives        540544  300     1802
106     Mali    20250833        1220190 17
107     Malta   441543  320     1380
108     Marshall Islands        59190   180     329
109     Mauritania      4649658 1030700 5
110     Mauritius       1271768 2030    626
111     Mexico  128932753       1943950 66
112     Micronesia      548914  700     784
113     Moldova 4033963 32850   123
114     Monaco  39242   1       26337
115     Mongolia        3278290 1553560 2
116     Montenegro      628066  13450   47
117     Morocco 36910560        446300  83
118     Mozambique      31255435        786380  40
119     Myanmar 54409800        653290  83
```

```
120     Namibia 2540905 823290  3
121     Nauru   10824   20      541
122     Nepal   29136808        143350  203
123     Netherlands     17134872        33720   508
124     New Zealand     4822233 263310  18
125     Nicaragua       6624554 120340  55
126     Niger   24206644        1266700 19
127     Nigeria 206139589       910770  226
128     North Korea     25778816        120410  214
129     North Macedonia 2083374 25220   83
130     Norway  5421241 365268  15
131     Oman    5106626 309500  16
132     Pakistan        220892340       770880  287
133     Palau   18094   460     39
134     Palestine State 5101414 6020    847
135     Panama  4314767 74340   58
136     Papua New Guinea        8947024 452860  20
137     Paraguay        7132538 397300  18
138     Peru    32971854        1280000 26
139     Philippines     109581078       298170  368
140     Poland  37846611        306230  124
141     Portugal        10196709        91590   111
142     Qatar   2881053 11610   248
143     Romania 19237691        230170  84
144     Russia  145934462       16376870        9
145     Rwanda  12952218        24670   525
146     Saint Kitts and Nevis   53199   260     205
147     Saint Lucia     183627  610     301
148     Saint Vincent and the Grenadines        110940  390     284
149     Samoa   198414  2830    70
```

```
150     San Marino      33931   60      566
151     Sao Tome and Principe   219159  960     228
152     Saudi Arabia    34813871        2149690 16
153     Senegal 16743927        192530  87
154     Serbia  8737371 87460   100
155     Seychelles      98347   460     214
156     Sierra Leone    7976983 72180   111
157     Singapore       5850342 700     8358
158     Slovakia        5459642 48088   114
159     Slovenia        2078938 20140   103
160     Solomon Islands 686884  27990   25
161     Somalia 15893222        627340  25
162     South Africa    59308690        1213090 49
163     South Korea     51269185        97230   527
164     South Sudan     11193725        610952  18
165     Spain   46754778        498800  94
166     Sri Lanka       21413249        62710   341
167     Sudan   43849260        1765048 25
168     Suriname        586632  156000  4
169     Sweden  10099265        410340  25
170     Switzerland     8654622 39516   219
171     Syria   17500658        183630  95
172     Tajikistan      9537645 139960  68
173     Tanzania        59734218        885800  67
174     Thailand        69799978        510890  137
175     Timor-Leste     1318445 14870   89
176     Togo    8278724 54390   152
177     Tonga   105695  720     147
178     Trinidad and Tobago     1399488 5130    273
179     Tunisia 11818619        155360  76
```

```
180     Turkey  84339067        769630  110
181     Turkmenistan    6031200 469930  13
182     Tuvalu  11792   30      393
183     Uganda  45741007        199810  229
184     Ukraine 43733762        579320  75
185     United Arab Emirates    9890402 83600   118
186     United Kingdom  67886011        241930  281
187     United States of America        331002651       9147420 36
188     Uruguay 3473730 175020  20
189     Uzbekistan      33469203        425400  79
190     Vanuatu 307145  12190   25
191     Venezuela       28435940        882050  32
192     Vietnam 97338579        310070  314
193     Yemen   29825964        527970  56
194     Zambia  18383955        743390  25
195     Zimbabwe        14862924        386850  38
```

In [ ]:
```
f.readline?
```

```
Signature: f.readline(size=-1, /)
Docstring:
Read until newline or EOF.

Returns an empty string if EOF is hit immediately.
Type:      builtin_function_or_method
```

3. readlines()

The function `readlines` reads all lines and store each line as an element in a list (of type string in this case).

```
content = open('../data/World_Nations_population_land_area.txt',
'r').readlines()

print(content)
```

Output:

```
['# Country\tPopulation(2020)\tLand Area
(Km²)\tDensity(P/Km²)\n', '1\tAfghanistan\t38928346\t652860
\t60\n', '2\tAlbania\t2877797\t27400\t105\n', ... , '194
\tZimbabwe\t14862924\t386850\t38']
```

The output shows that between the line components a tab '\t' is used as separator.

In [ ]:
```
content = open('../data/World_Nations_population_land_area.txt', 'r').rea

print(content)
```

When you know the number of a specific line of interest, you can directly read the line by indexing.

The 65th Nation is Germany and we know that the first line is a header. So we can use the index 65 (line 66) to get the information about the German population.

```
f = open('../data/World_Nations_population_land_area.txt', 'r')

print(f.readlines()[65])
```

The result from above is the same as the result from below except for the difference that above the file object still needs to be closed when it is no longer needed. myline is an object of type string.

```
myline = open('../data/World_Nations_population_land_area.txt',
'r').readlines()[65]

print(myline)
```

In [ ]:
```
f = open('../data/World_Nations_population_land_area.txt', 'r')
content = f.readlines()
print(content[45])
print(type(content))
print(len(content))
print(content[50])
```

```
45      Czechia 10708981        77240   139

<class 'list'>
196
50      Dominican Republic      10847910        48320   225
```

Exercise

1. From file World_Nations_population_land_area.txt read the 125th line and tell us which country you have found.
2. Print the countries #45 to #50

In [ ]:
```
# 1:
f = open('../data/World_Nations_population_land_area.txt', 'r')
print(f.readlines()[125])
f.close()
```

```
125     Nicaragua       6624554 120340  55
```

In [ ]:
```
# 2:
f = open('../data/World_Nations_population_land_area.txt', 'r')

content=f.readlines()
for i in range(45, 51):
     print(content[i])
f.close()
```

```
45      Czechia 10708981        77240   139

46      Democratic Republic of the Congo        89561403        2267050 4
0

47      Denmark 5792202 42430   137

48      Djibouti        988000  23180   43

49      Dominica        71986   750     96

50      Dominican Republic      10847910        48320   225
```

Open a file for writing

When writing to a file, a distinction is made between creating a new file, appending data to an existing file, or overwriting an existing file. This is done by selecting the appropriateed mode in the open statement (see above).

```
fout = open('myoutput.txt', 'w')
```

It creates a file object in text write mode linked to the file *myoutput.txt*. If *myoutput.txt* exists it will be overwritten.

Write to file

Equivalent to the `read` function Python provides a `write` function as well. Every call of `write` append the input to the file until it is closed. It is important to add blanks and newlines '\n' to get the wanted text result.

```
fout = open('myoutput.txt', 'w')

x = 10
y = 5

fout.write('This is our first write call without a newline
special character.')
fout.write('And here it comes - the second write call without a
newline special character.')
fout.write(str(x))
fout.write(str(y))

fout.write('\n---------\n')
fout.write('We should use a newline character. \n')
fout.write(str(x) + '\n')
fout.write(str(y) + '\n')

fout.write('\n---------\n')
fout.write('Or a blank character.\n')
fout.write(str(x) + ' ' + str(y) + '\n')
fout.close()
```

Exercise

1. Open a new file
2. Write some text and data to the file. Use newline and blanks to format the output.
3. Close the file
4. Open the file again, but you have to append some lines.
5. Add some text and data to the file.

In [ ]:
```
# 1.
fout = open('outy.txt', 'w')
```

In [ ]:
```
# 2.
pi = 3
fout.write('The value of pi' + '\n')
fout.write(str(pi) + '\n')
fout.close()
```

In [ ]:
```
# 3.
import math
fout = open('outy.txt', 'a')
fout.write('The REAL value of pi' + '\n')
fout.write(str(math.pi) + '\n')
fout.close()
```

In [ ]:
```
# 4.
```

In [ ]:
```
# 5.
```

Read binary data

There are different packages available for reading and writing binary data. The package Numpy can be used to write arrays (sequences of values) to an file.

For simplicity we skip the binary part, since it is rarely used.

Read CSV file

The CSV (comma separated values) format is very often used to store ASCII data in an file. Programns like Excel can import and export these format which makes it easy for the exchange of data. The values are separated by a delimiter (default: comma) which can be set or changed by the user.

The Python module **csv** provides classes for reading and writing this form of data.

```
import csv
```

The `reader()` class allows us to read the file content into an iterable *reader* object. A `delimiter` can be set and, if needed, you can skip initial blanks with the `skipinitialspace=True` .

To read only the first 6 lines of the CSV file we use *enumerate* to get an iterable object:

```
with open('../data/World_Nations_population_land_area.txt') as
csvfile:
    data = csv.reader(csvfile, delimiter='\t')
    for index, row in enumerate(data):
        if index <=5:
            print(row)
```

```
['# Country', 'Population(2020)', 'Land Area (Km²)',
'Density(P/Km²)']
['1', 'Afghanistan', '38928346', '652860', '60']
['2', 'Albania', '2877797', '27400', '105']
['3', 'Algeria', '43851044', '2381740', '18']
['4', 'Andorra', '77265', '470', '164']
['5', 'Angola', '32866272', '1246700', '26']
```

In [ ]:
```
import csv

with open('../data/World_Nations_population_land_area.txt') as csvfile:
    data = csv.reader(csvfile, delimiter='\t')

    for index, row in enumerate(data):
        if index <=5:
            print(row)
```

```
['# Country', 'Population(2020)', 'Land Area (Km²)', 'Density(P/Km²)']
['1', 'Afghanistan', '38928346', '652860', '60']
['2', 'Albania', '2877797', '27400', '105']
['3', 'Algeria', '43851044', '2381740', '18']
['4', 'Andorra', '77265', '470', '164']
['5', 'Angola', '32866272', '1246700', '26']
```

When the CSV file has a more unlike format because the delimiter is not one character some work has to be done.

Example input file:

```
"Number" | "Name" | "Type"
1 | "Balu" | "Bear"
2 | "Groot" | "Tree"
3 | "Chewbacca" | "Wookiee"
4 | "Duffy" | "Duck"
```

The initial blank character can be skipped by setting *skipinitialspace* to True.

```
with open('../data/species_not_real.csv') as infile:
    species = csv.reader(infile, delimiter='|',
skipinitialspace=True)
    for row in species:
        print(row)
```

Output:

```
['Number ', 'Name ', 'Type']
['1 ', 'Balu ', 'Bear']
['2 ', 'Groot ', 'Tree']
['3 ', 'Chewbacca ', 'Wookiee']
['4 ', 'Duffy ', 'Duck']
```

The trailing blank character of the first two elements of each row will remain. The `strip()` method will delete it anyway.

```
with open('../data/species_not_real.csv') as infile:
    species = csv.reader(infile, delimiter='|',
skipinitialspace=True)
    for row in species:
        row = row.strip()
        print(row)
```

The output is

```
['Number', 'Name', 'Type']
['1', 'Balu', 'Bear']
['2', 'Groot', 'Tree']
['3', 'Chewbacca', 'Wookiee']
['4', 'Duffy', 'Duck']
```

In [ ]:
```
with open('../data/species_not_real.csv') as infile:
    species = csv.reader(infile, delimiter='|', skipinitialspace=True)
    for row in species:
        print(row)
        row = list([x.strip() for x in row])
        print(row)
```

```
['Number ', 'Name ', 'Type']
['Number', 'Name', 'Type']
['1 ', 'Balu ', 'Bear']
['1', 'Balu', 'Bear']
['2 ', 'Groot ', 'Tree']
['2', 'Groot', 'Tree']
['3 ', 'Chewbacca ', 'Wookiee']
['3', 'Chewbacca', 'Wookiee']
['4 ', 'Duffy ', 'Duck']
['4', 'Duffy', 'Duck']
```

With the object class DictReader() the content of the CSV file can be read as a dictionary. But we already stick with the initial and trailing blank problem. To solve this issue we define a function which will remove the blanks.

```
def strip_dict(d):
    return { key.strip() : strip_dict(value)
             if isinstance(value, dict)
             else value.strip()
             for key, value in d.items() }

with open('../data/species_not_real.csv') as infile:
    species = csv.DictReader(infile, delimiter='|',
skipinitialspace=True)
    for row in species:
        row1 = strip_dict(row)
        print(row1)
```

```
{'Number': '1', 'Name': 'Balu', 'Type': 'Bear'}
{'Number': '2', 'Name': 'Groot', 'Type': 'Tree'}
{'Number': '3', 'Name': 'Chewbacca', 'Type': 'Wookiee'}
{'Number': '4', 'Name': 'Duffy', 'Type': 'Duck'}
```

In [ ]:
```
def strip_dict(d):
    return { key.strip() : strip_dict(value)
             if isinstance(value, dict)
             else value.strip()
             for key, value in d.items() }

with open('../data/species_not_real.csv') as infile:
    species = csv.DictReader(infile, delimiter='|', skipinitialspace=True
    for row in species:
        row1 = strip_dict(row)
        print(row1)
        print(row)
```

```
{'Number': '1', 'Name': 'Balu', 'Type': 'Bear'}
{'Number ': '1 ', 'Name ': 'Balu ', 'Type': 'Bear'}
{'Number': '2', 'Name': 'Groot', 'Type': 'Tree'}
{'Number ': '2 ', 'Name ': 'Groot ', 'Type': 'Tree'}
{'Number': '3', 'Name': 'Chewbacca', 'Type': 'Wookiee'}
{'Number ': '3 ', 'Name ': 'Chewbacca ', 'Type': 'Wookiee'}
{'Number': '4', 'Name': 'Duffy', 'Type': 'Duck'}
{'Number ': '4 ', 'Name ': 'Duffy ', 'Type': 'Duck'}
```

Write CSV file

Just as we expect, there is a `writer()` class for writing data to a file in CSV format.

Let's assume we want to write the following values to a file with a comma as delimiter.

```
a = [1,2,3,4,5,6]

with open('./test.csv', 'w', encoding='UTF8') as fout:
    writer = csv.writer(fout, delimiter=',')
    writer.writerow(a)
```

File content test.csv:

```
1,2,3,4,5,6
```

In [ ]:
```
a = [1,2,3,4,5,6]

with open('./test.csv', 'w', encoding='UTF8') as fout:
    writer = csv.writer(fout, delimiter=',')
    writer.writerow(a)
```

You can combine multiple lists using `writerow()` for a single list and `writerows()` for nested lists.

A good example of use is to save a header and its associated data as follows:

```
header = ['CONTINENT', 'NAME', 'ISO2', 'ISO3']

data = [['Africa', 'Algeria', 'DZ', 'DZA'],
        ['Africa', 'Angola', 'AO', 'AGO'],
        ['Africa', 'Benin', 'BJ', 'BEN'],
        ['Africa', 'Botswana', 'BW', 'BWA'],
        ['Africa', 'Burkina Faso', 'BF', 'BFA']]

with open('./countries.csv', 'w', encoding='UTF8') as f:
    writer = csv.writer(f, delimiter=',' )
    writer.writerow(header)
    writer.writerows(data)
```

File content of countries.csv:

```
CONTINENT,NAME,ISO2,ISO3
Africa,Algeria,DZ,DZA
Africa,Angola,AO,AGO
Africa,Benin,BJ,BEN
Africa,Botswana,BW,BWA
Africa,Burkina Faso,BF,BFA
```

In [ ]:
```
header = ['CONTINENT', 'NAME', 'ISO2', 'ISO3']

data = [['Africa', 'Algeria', 'DZ', 'DZA'],
        ['Africa', 'Angola', 'AO', 'AGO'],
        ['Africa', 'Benin', 'BJ', 'BEN'],
        ['Africa', 'Botswana', 'BW', 'BWA'],
        ['Africa', 'Burkina Faso', 'BF', 'BFA']]

with open('./countries.csv', 'w', encoding='UTF8') as f:
    writer = csv.writer(f, delimiter=',' )
    writer.writerow(header)
    writer.writerows(data)
```

Exercise

Input data: https://www.worldometers.info/world-population/population-by-country/

1. Create a list that contains the header information: name, population, density
2. Create a list of the 5 countries with the largest population
3. Write the CSV data with a semicolon as delimiter to the file *country_population.csv*

In [ ]:
```
# 1.
```

In [ ]:
```
# 2.
```

In [ ]:
```
# 3.
```