Files

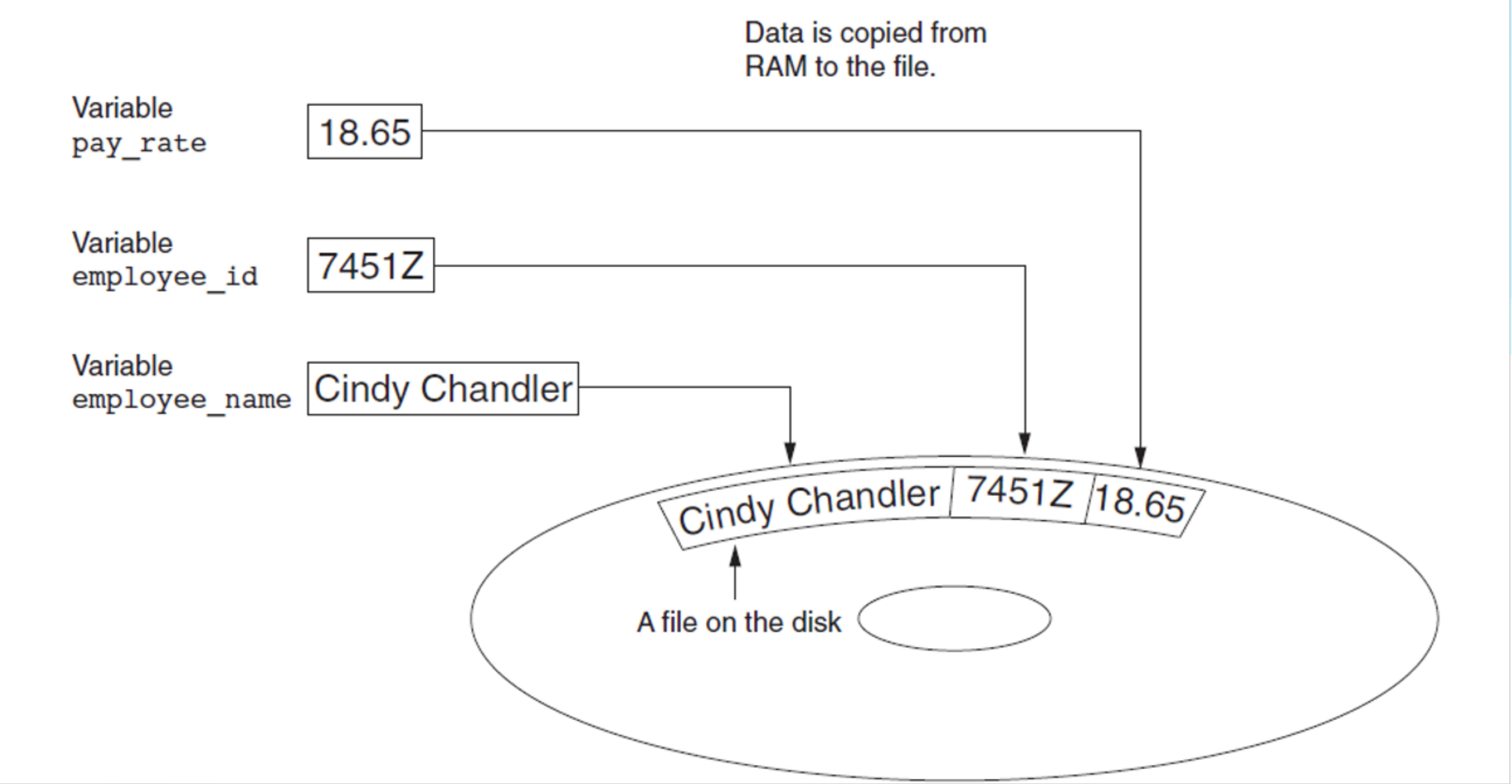
Topics

- Introduction to File Input and Output
- Using Loops to Process Files
- Processing Records

Introduction to File Input and Output

- For program to retain data between the times it is run, you must save the data
- Data is saved to a file, typically on computer disk
- Saved data can be retrieved and used at a later time
- “**Writing data to**”: saving data on a file
- **Output file**: a file that data is written to

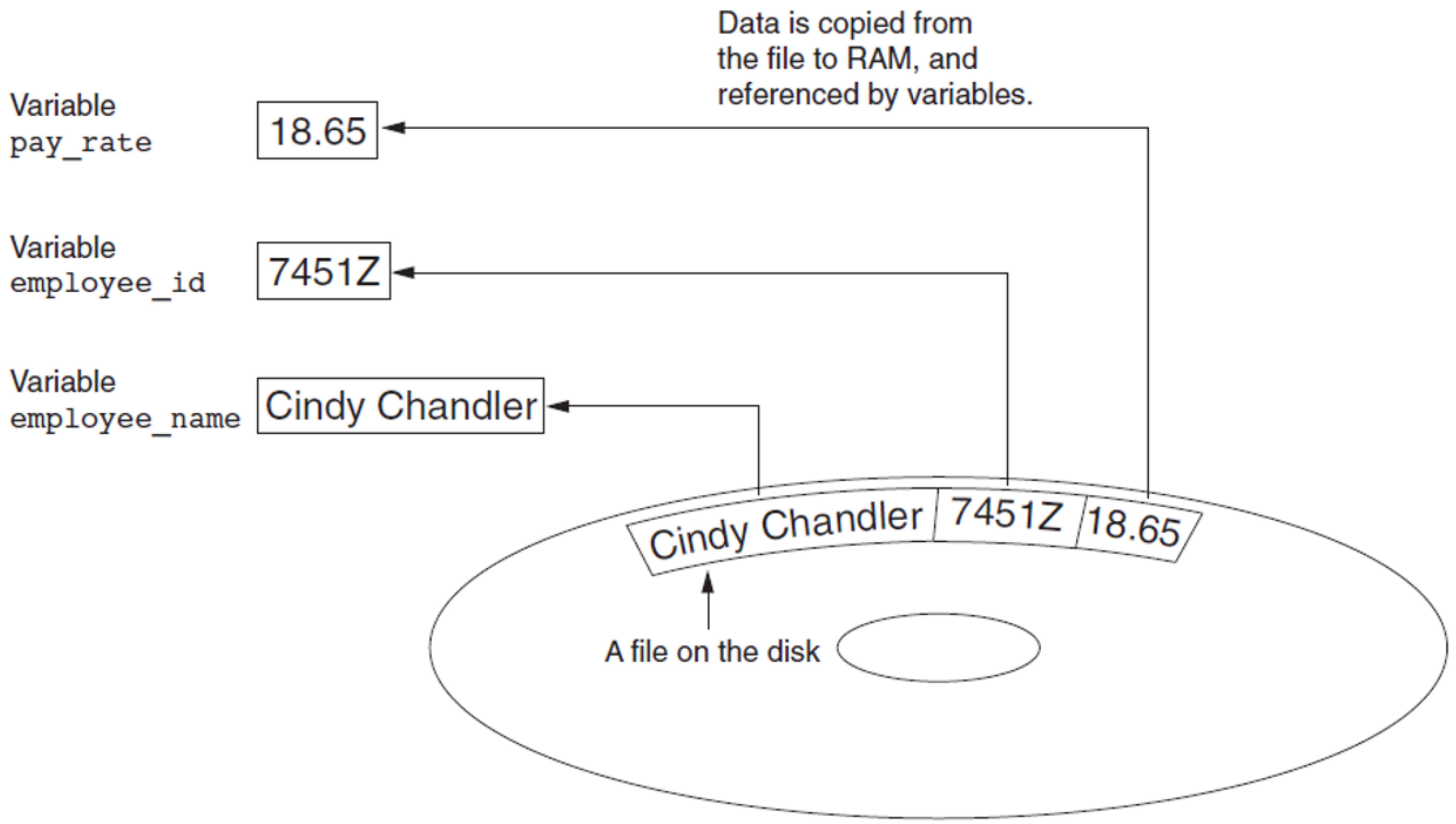
Figure 6-1 Writing data to a file



Introduction to File Input and Output (cont’d.)

- “**Reading data from**”: process of retrieving data from a file
- **Input file**: a file from which data is read
- Three steps when a program uses a file
  - Open the file
  - Process the file
  - Close the file

Figure 6-2 Reading data from a file



Types of Files and File Access Methods

- In general, two types of files
  - Text file: contains data that has been encoded as text
  - Binary file: contains data that has not been converted to text

Types of Files and File Access Methods (cont’d.)

- Two ways to access data stored in file
  - Sequential access: file read sequentially from beginning to end, can’t skip ahead

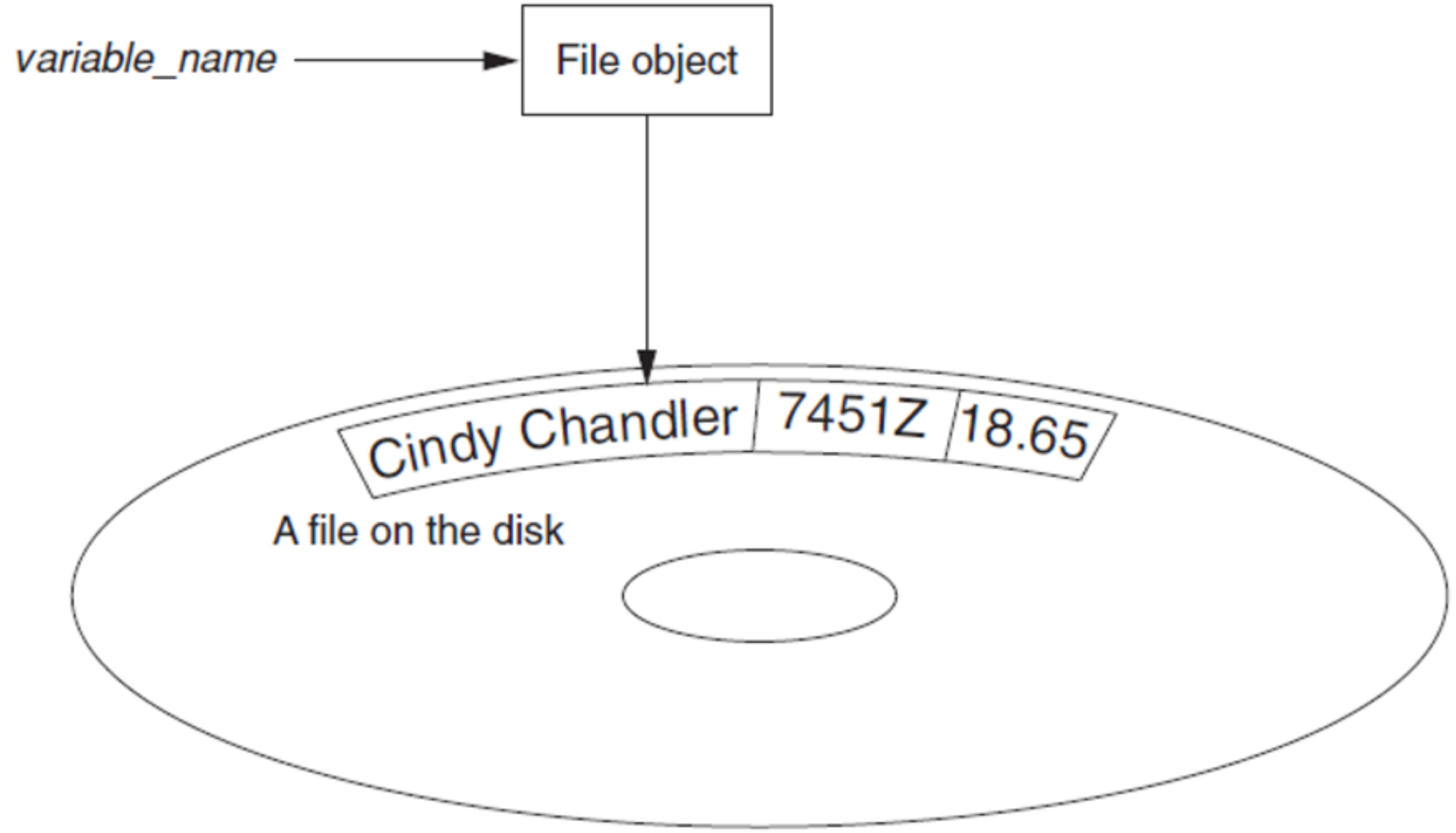
  - Direct access: can jump directly to any piece of data in the file

Filenames and File Objects

- Filename extensions: short sequences of characters that appear at the end of a filename preceded by a period
  - Extension indicates type of data stored in the file
- File object: object associated with a specific file
  - Provides a way for a program to work with the file: file object referenced by a variable

Filenames and File Objects (cont'd.)

Figure 6-4 A variable name references a file object that is associated with a file



Opening a File

- open function: used to open a file
- Creates a file object and associates it with a file on the disk
- General format:

```
    file_object = open(filename, mode)
```

- Mode: string specifying how the file will be opened

    Example: reading only ('r'), writing ('w'), and appending ('a')

Some of the Python file modes

```
customer_file = open('customers.txt', 'r')

sales_file = open('sales.txt', 'w')
```

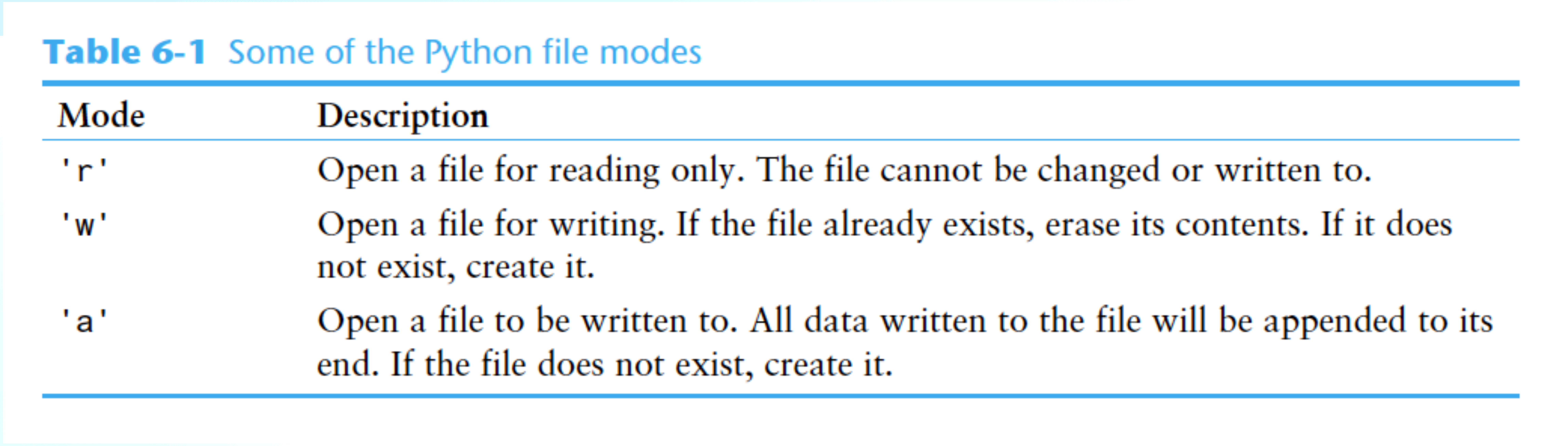


Table 6-1 Some of the Python file modes

Mode	Description
`'r'`	Open a file for reading only. The file cannot be changed or written to.
`'w'`	Open a file for writing. If the file already exists, erase its contents. If it does not exist, create it.
`'a'`	Open a file to be written to. All data written to the file will be appended to its end. If the file does not exist, create it.

Writing Data to a File

- **Method**: a function that belongs to an object
  - Performs operations using that object
- File object's write method used to write data to the file
  - Format: `file_variable.write(string)`
- File should be closed using file object close method
  - Format: `file_variable.close()`

file_write2.py

Chapter 06

```python
def main():
    # Open a file named
    outfile = open('test1.txt', 'w')

    # Write lines by write() method
    outfile.write('Line 1')
    outfile.write('Line 2')
    outfile.write('Line 3')

    # Write lines by writelines() method
    list1 = ['Line 4', 'Line 5', 'Line 6']
    outfile.writelines(list1)

    # Close the file.
    outfile.close()

main()
```

```
Line 1Line 2Line 3Line 4Line 5Line 6
```

file_write3.py

```python
def main():
    # Open a file named
    outfile = open('test2.txt', 'w')

    # Write lines by write() method
    outfile.write('Line 1\n')
    outfile.write('Line 2\n')
    outfile.write('Line 3\n')

    # Write lines by writelines() method
    list1 = ['Line 4\n', 'Line 5\n', 'Line 6\n']
    outfile.writelines(list1)

    # Close the file.
    outfile.close()

main()
```

```
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6

```

Specifying the Location of a File

- If open function receives a filename that does not contain a path, assumes that file is in same directory as program
- If program is running and file is created, it is created in the same directory as the program
  - Can specify alternative path and file name in the open function argument
    - Prefix the path string literal with the letter `r`

Specifying the Location of a File (cont’d.)

```
test_file = open(‘test.txt’, ’w’)

test_file = open(r’c:\temp\test.txt’, ‘w’)
```

- The r prefix specifies that the string is a raw string.

Reading Data From a File

- **read method**: file object method that reads entire file contents into memory
- Only works if file has been opened for reading
- Contents returned as a string

```
# Open a file in read mode
infile = open('file.txt', 'r')

# Read the file's conents.
file_contents = infile.read()
```

file_read2.py

```python
def main():
    # Open a file named test2.txt
    infile = open('test2.txt', 'r')

    # Read the file's contents.
    file_contents = infile.read()

    # Close the file.
    infile.close()

    # Print the data that was read intomemory.
    print(file_contents)

main()
```

```
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
```

Reading Data From a File (cont'd.)

- **readline method**: file object method that reads a line from the file
- Line returned as a string, including '\n'

```
# Open a file in read mode
infile = open('file.txt', 'r')

# Read lines from the file
line1 = infile.readline()
line2 = infile.readline()
line3 = infile.readline()
```

line_read2.py

```python
def main():
    # Open a file named test2.txt.
    infile = open('test2.txt', 'r')

    # Read three lines from the file
    line1 = infile.readline()
    line2 = infile.readline()
    line3 = infile.readline()

    # Close the file.
    infile.close()

    # Print the data that was read into
    # memory.
    print(line1)
    print(line2)
    print(line3)

main()
```

```
Line 1

Line 2

Line 3

```

Concatenating a Newline to and Stripping it From a String

- **In most cases, data items written to a file are values referenced by variables**
  - Usually necessary to concatenate a `'\n'` to data before writing it
    - Carried out using the + operator in the argument of the write method

```
outfile = open('file.txt', 'w')
name = input('Input name: ')
outfile.write(name + '\n')
```

write_names.py

```python
# This program gets three names from the user
# and writes them to a file.

def main():
    # Get three names.
    print('Enter the names of three friends.')
    name1 = input('Friend #1: ')
    name2 = input('Friend #2: ')
    name3 = input('Friend #3: ')

    # Open a file named friends.txt.
    myfile = open('friends.txt', 'w')

    # Write the names to the file.
    myfile.write(name1 + '\n')
    myfile.write(name2 + '\n')
    myfile.write(name3 + '\n')

    # Close the file.
    myfile.close()
    print('The names were written to friends.txt.')

# Call the main function.
main()
```

```
Enter the names of three friends.
Friend #1: Joe
Friend #2: Rose
Friend #3: Geri
The names were written to friends.txt.
```

Concatenating a Newline to and Stripping it From a String (cont'd.)

- **In many cases need to remove `'\n'` from string after it is read from a file**
  - `rstrip` method: string method that strips specific characters from end of the string

```
infile = open('file.txt', 'r')
line1 = infile.readline()
line1 = line1.rstrip('\n')
```

strip_newline2.py

```python
# This program reads the contents of the
# philosophers.txt file one line at a time.
def main():
    # Open a file named test2.txt.
    infile = open('test2.txt', 'r')

    # Read three lines from the file
    line1 = infile.readline()
    line2 = infile.readline()
    line3 = infile.readline()

    # Strip the \n from each string.
    line1 = line1.rstrip('\n')
    line2 = line2.rstrip('\n')
    line3 = line3.rstrip('\n')

    # Close the file.
    infile.close()

    # Print the data that was read into
    # memory.
    print(line1)
    print(line2)
    print(line3)

# Call the main function.
main()

```

```
Line 1
Line 2
Line 3
```

Appending Data to an Existing File

- When open file with 'w' mode, if the file already exists it is **overwritten**
- To **append data** to a file use the 'a' mode
    - If file exists, it is not erased, and if it does not exist it is created
    - Data is written to the file at the end of the current contents

```
outfile = open('file.txt', 'a')
Outfile.write('Somchai\n')
```

Writing and Reading Numeric Data

- Numbers must be converted to strings before they are written to a file
- **`str` function**: converts value to string

```
outfile = open('file.txt', 'w')
outfile.write(str(101) + '\n')
outfile.write(str(202) + '\n')
```

write_numbers.py

```python
# This program demonstrates how numbers
# must be converted to strings before they
# are written to a text file.

def main():
    # Open a file for writing.
    outfile = open('numbers.txt', 'w')

    # Get three numbers from the user.
    num1 = int(input('Enter a number: '))
    num2 = int(input('Enter another number: '))
    num3 = int(input('Enter another number: '))

    # Write the numbers to the file.
    outfile.write(str(num1) + '\n')
    outfile.write(str(num2) + '\n')
    outfile.write(str(num3) + '\n')

    # Close the file.
    outfile.close()
    print('Data written to numbers.txt')

# Call the main function.
main()

```

```
Enter a number: 22
Enter another number: 14
Enter another number: -99
Data written to numbers.txt
```

Writing and Reading Numeric Data (cont'd.)

- **Number are read from a text file as strings**
    - Must be converted to numeric type in order to perform mathematical operations
    - Use int and float functions to convert string to numeric value

```
infile = open('file.txt', 'r')
line1 = infile.readline()
line1 = int(line1)
line2 = int(infile.readline())
```

read_numbers.py

```python
# This program demonstrates how numbers that are
# read from a file must be converted from strings
# before they are used in a math operation.

def main():
    # Open a file for reading.
    infile = open('numbers.txt', 'r')

    # Read three numbers from the file.
    num1 = int(infile.readline())
    num2 = int(infile.readline())
    num3 = int(infile.readline())

    # Close the file.
    infile.close()

    # Add the three numbers.
    total = num1 + num2 + num3

    # Display the numbers and their total.
    print('The numbers are:', num1, num2, num3)
    print('Their total is:', total)

# Call the main function.
main()

```

```
The numbers are: 22 14 -99
Their total is: -63
```

Using Loops to Process Files

- Files typically used to hold large amounts of data
  - Loop typically involved in reading from and writing to a file

write_sales.py

```python
# This program prompts the user for sales amounts
# and writes those amounts to the sales.txt file.

def main():
    # Get the number of days.
    num_days = int(input('For how many days do ' + \
                         'you have sales? '))

    # Open a new file named sales.txt.
    sales_file = open('sales.txt', 'w')

    # Get the amount of sales for each day and write
    # it to the file.
    for count in range(1, num_days + 1):
        # Get the sales for a day.
        sales = float(input('Enter the sales for day #' + \
                            str(count) + ': '))

        # Write the sales amount to the file.
        sales_file.write(str(sales) + '\n')

    # Close the file.
    sales_file.close()
    print('Data written to sales.txt.')

# Call the main function.
main()

```

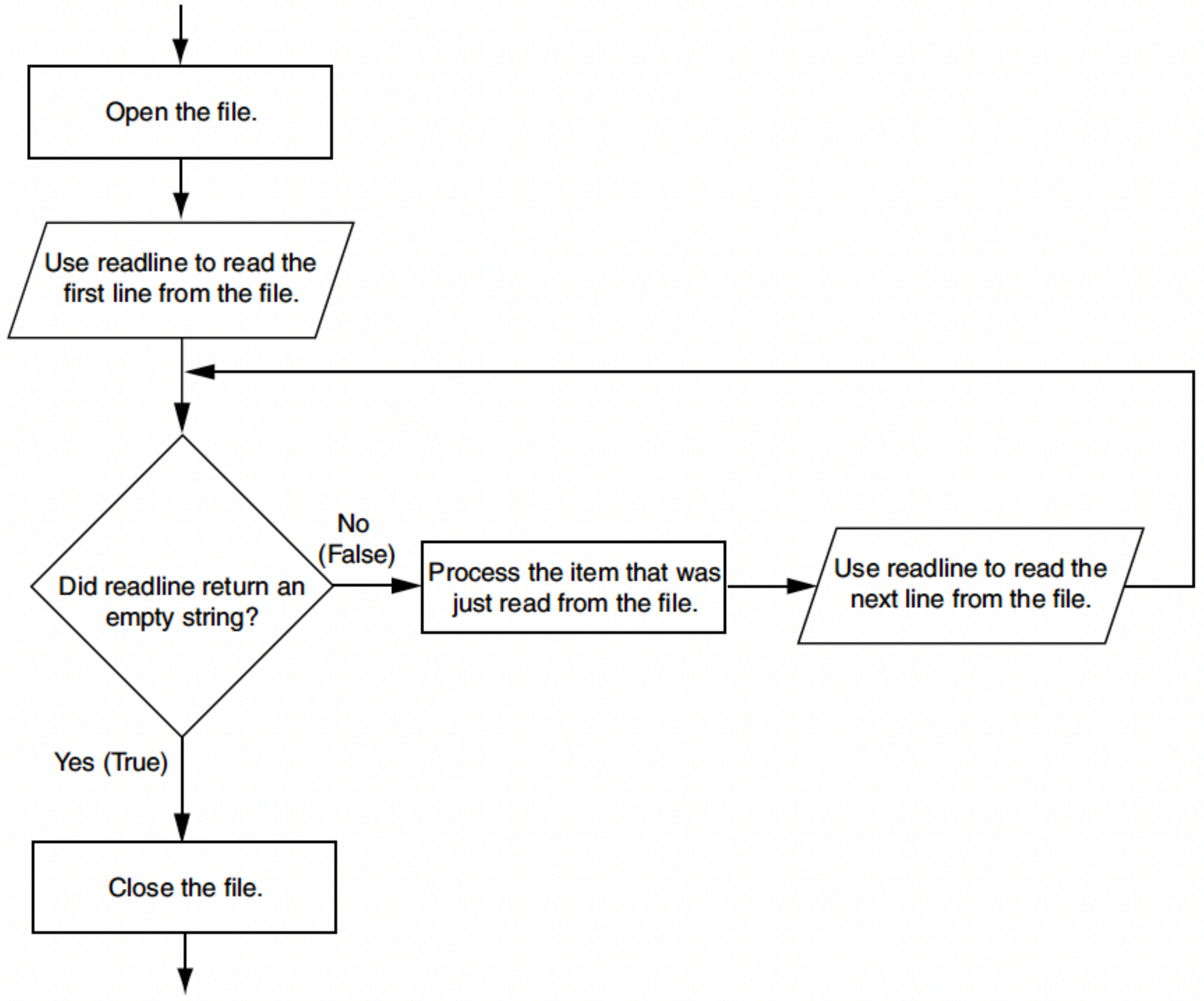
```
For how many days do you have sales? 5
Enter the sales for day #1: 1000.0
Enter the sales for day #2: 2000.0
Enter the sales for day #3: 3000.0
Enter the sales for day #4: 4000.0
Enter the sales for day #5: 5000.0
Data written to sales.txt.
```

Using Loops to Process Files (cont'd.)

- Often the number of items stored in file is unknown
  - The `readline` method uses an empty string as a sentinel when end of file is reached
  - Can write a while loop with the condition

```
    while line != ''
```

Figure 6-17 General logic for detecting the end of a file


Open the file.
Use readline to read the first line from the file.
Did readline return an empty string?
No (False)
Process the item that was just read from the file.
Use readline to read the next line from the file.
Yes (True)
Close the file.

read_sales.py

```python
# This program reads all of the values in
# the sales.txt file.

def main():
    # Open the sales.txt file for reading.
    sales_file = open('sales.txt', 'r')

    # Read the first line from the file, but
    # don't convert to a number yet. We still
    # need to test for an empty string.
    line = sales_file.readline()

    # As long as an empty string is not returned
    # from readline, continue processing.
    while line != '':
        # Convert line to a float.
        amount = float(line)

        # Format and display the amount.
        print(format(amount, '.2f'))

        # Read the next line.
        line = sales_file.readline()

    # Close the file.
    sales_file.close()

# Call the main function.
main()

```

```
1000.00
2000.00
3000.00
4000.00
5000.00
```

Using Python’s for Loop to Read Lines

- Python allows the programmer to write a for loop that automatically reads lines in a file and stops when end of file is reached
- Format:

```
for line in file_object:
    statements
```

- The loop iterates once over each line in the file

read_sales2.py

```python
# This program uses the for loop to read
# all of the values in the sales.txt file.

def main():
    # Open the sales.txt file for reading.
    sales_file = open('sales.txt', 'r')

    # Read all the lines from the file.
    for line in sales_file:
        # Convert line to a float.
        amount = float(line)
        # Format and display the amount.
        print(format(amount, '.2f'))

    # Close the file.
    sales_file.close()

# Call the main function.
main()

```

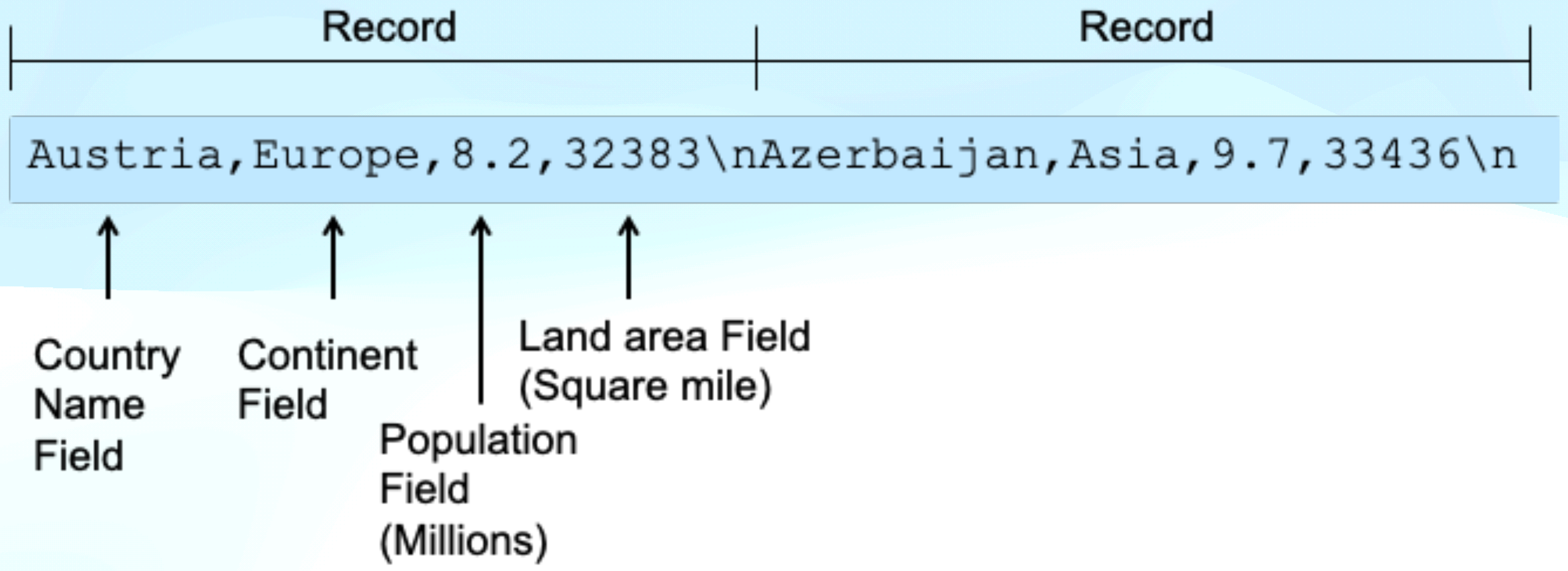
```
1000.00
2000.00
3000.00
4000.00
5000.00
```

CSV Files

- Another type of text file, called a **CSV-formatted file**, has several items of data on each line with the items separated by commas.
- CSV stands for Comma-Separated Values.

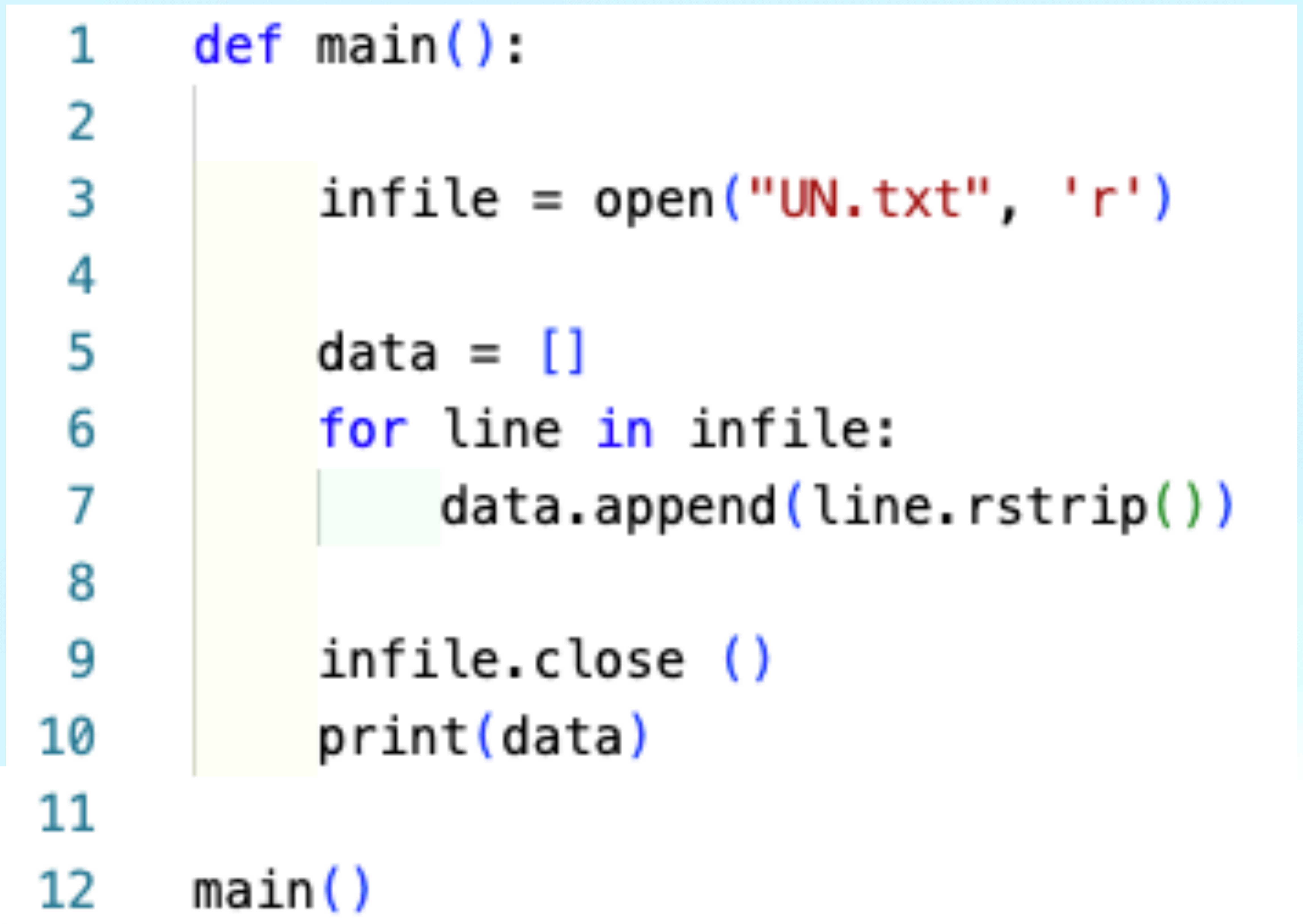
```
Austria,Europe,8.2,32383
Azerbaijan,Asia,9.7,33436
Bahamas,North America,.32,5358
Bahrain,Asia,1.3,253
```

CSV Files (cont'd.)



Record
Record
Austria,Europe,8.2,32383\nAzerbaijan,Asia,9.7,33436\n
Country Name Field
Continent Field
Population Field (Millions)
Land area Field (Square mile)

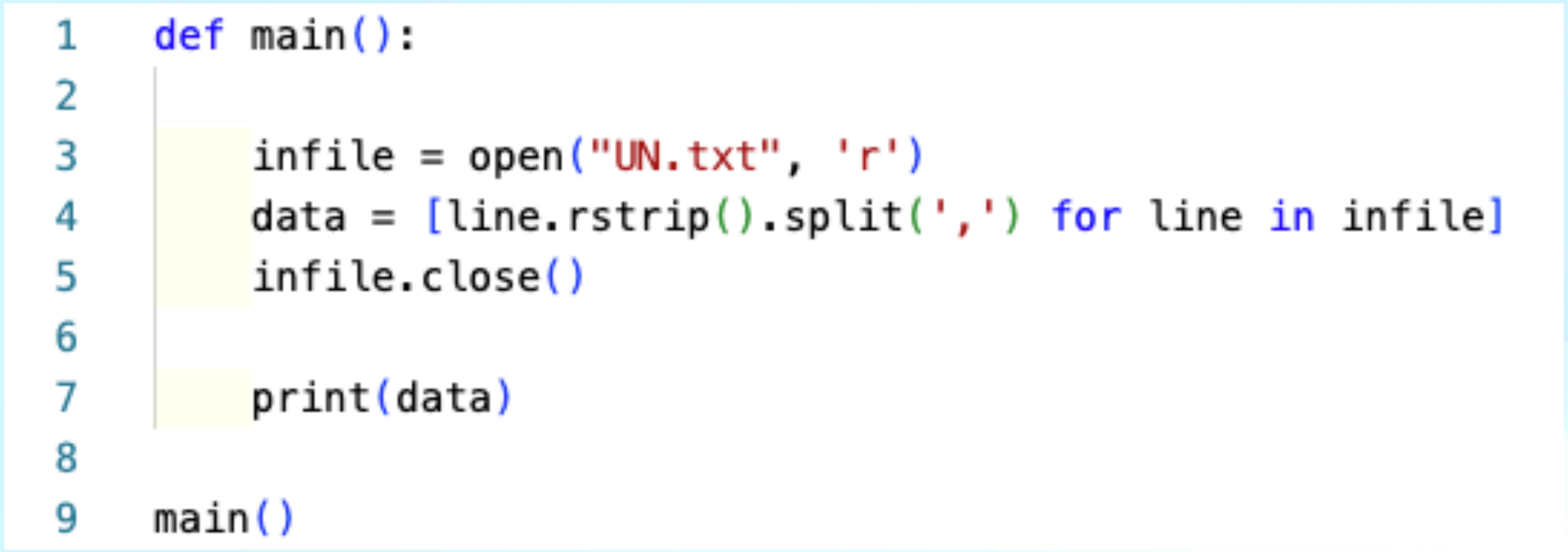
15-1.py



```python
def main():

    infile = open("UN.txt", 'r')

    data = []
    for line in infile:
        data.append(line.rstrip())

    infile.close ()
    print(data)

main()
```

```
['Afghanistan,Asia,31.8,251772', 'Albania,Europe,3.0,11100',
 'Algeria,Africa,38.3,919595', 'Andorra,Europe,.085,181',
 'Angola,Africa,19.1,481354', 'Antigua and Barbuda,North America,.091,108',
 ...]
```

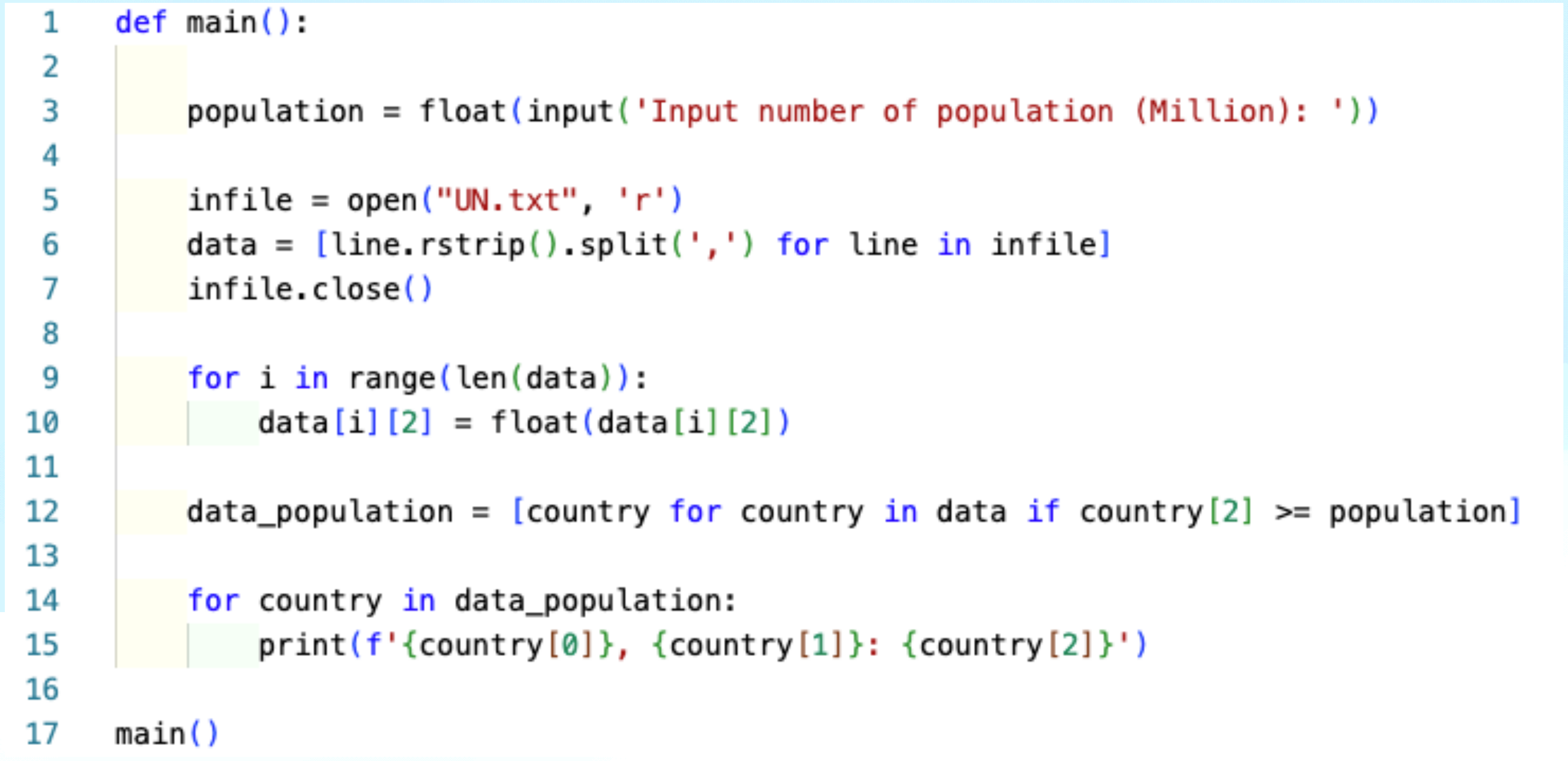
15-2.py



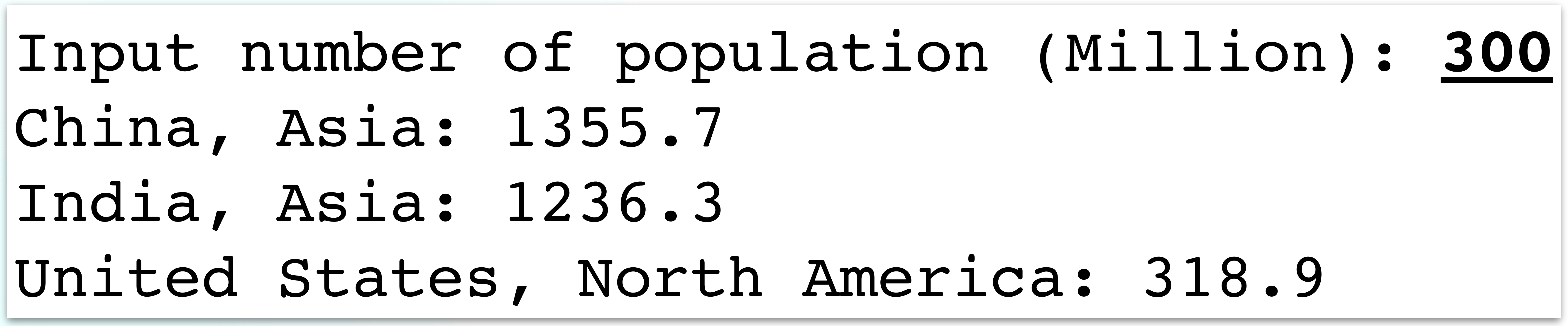
```python
def main():

    infile = open("UN.txt", 'r')
    data = [line.rstrip().split(',') for line in infile]
    infile.close()

    print(data)

main()
```

```
[['Afghanistan', 'Asia', '31.8', '251772'],
 ['Albania', 'Europe', '3.0', '11100'],
 ['Algeria', 'Africa', '38.3', '919595'],
 ['Andorra', 'Europe', '.085', '181'],
 ['Angola', 'Africa', '19.1', '481354'],
 ['Antigua and Barbuda', 'North America', '.091', '108'], ...]
```

15-3.py

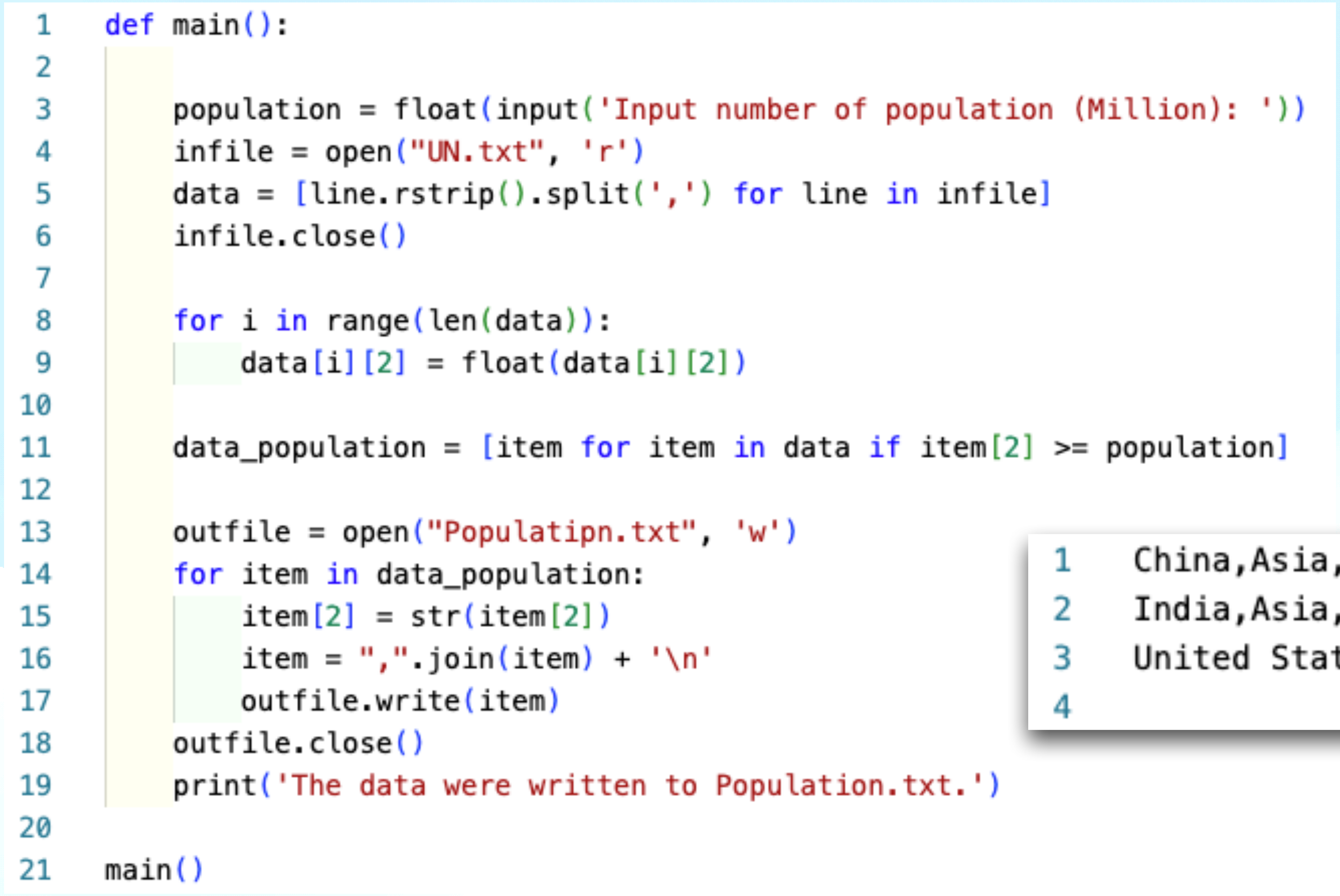


```python
def main():

    population = float(input('Input number of population (Million): '))

    infile = open("UN.txt", 'r')
    data = [line.rstrip().split(',') for line in infile]
    infile.close()

    for i in range(len(data)):
        data[i][2] = float(data[i][2])

    data_population = [country for country in data if country[2] >= population]

    for country in data_population:
        print(f'{country[0]}, {country[1]}: {country[2]}')

main()
```

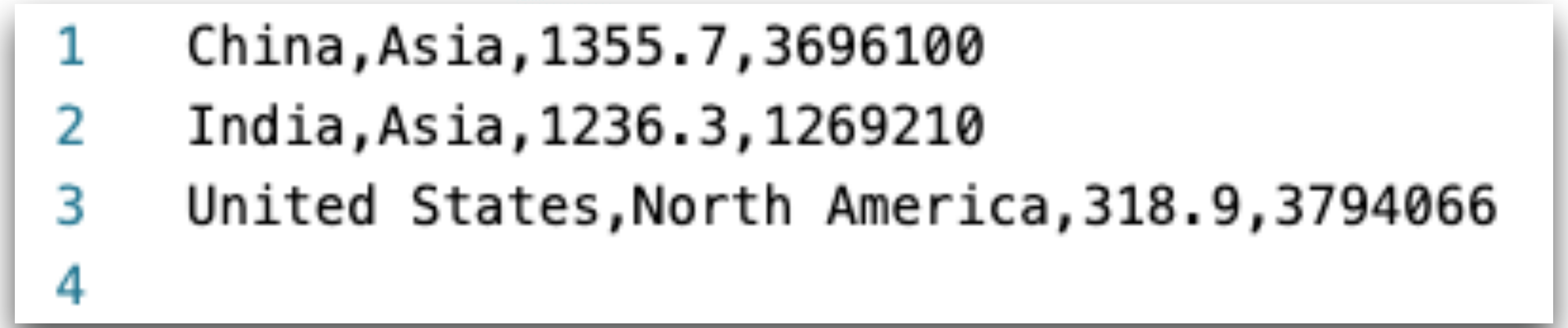


```
Input number of population (Million): 300
China, Asia: 1355.7
India, Asia: 1236.3
United States, North America: 318.9
```

15-4.py

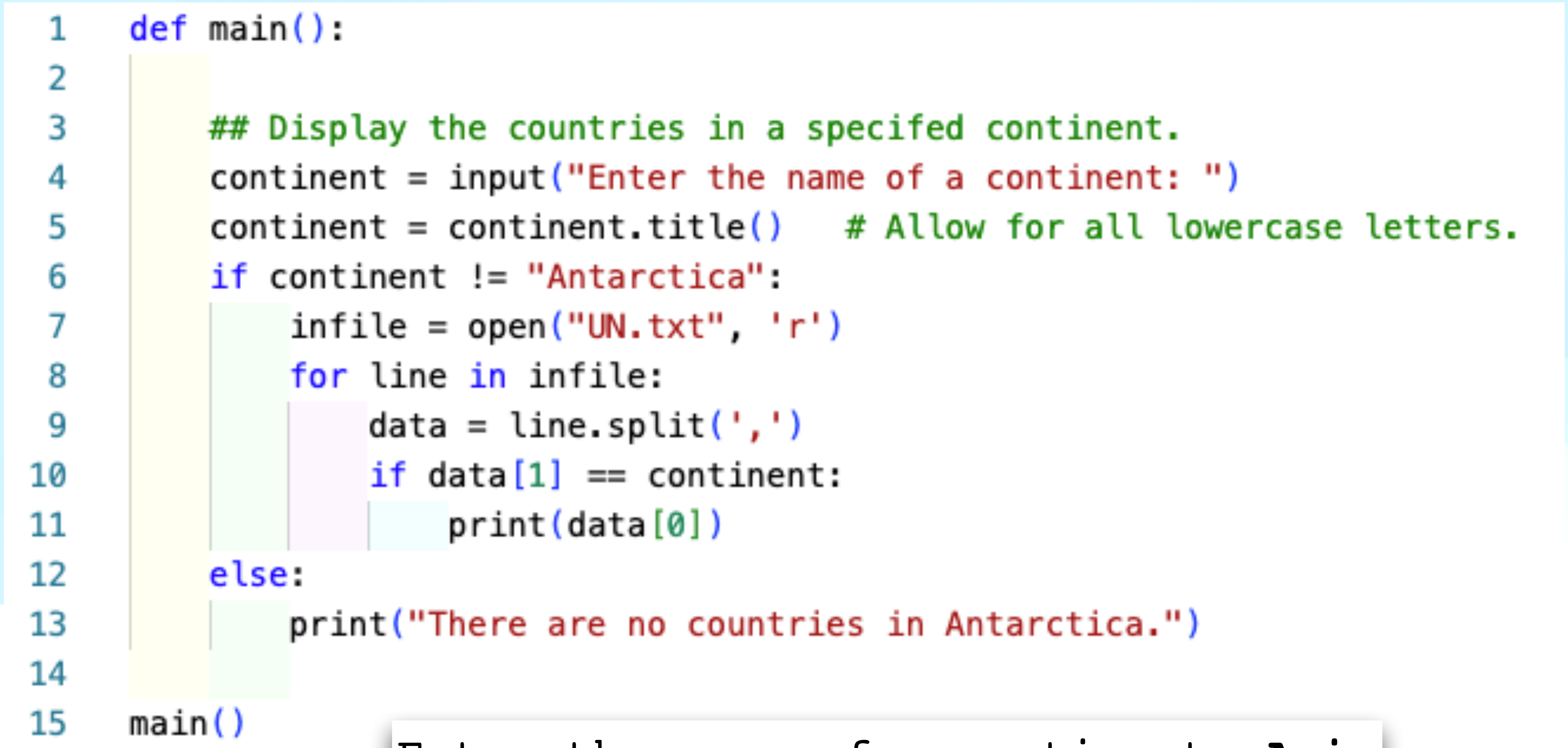


```python
def main():

    population = float(input('Input number of population (Million): '))
    infile = open("UN.txt", 'r')
    data = [line.rstrip().split(',') for line in infile]
    infile.close()

    for i in range(len(data)):
        data[i][2] = float(data[i][2])

    data_population = [item for item in data if item[2] >= population]

    outfile = open("Populatipn.txt", 'w')
    for item in data_population:
        item[2] = str(item[2])
        item = ",".join(item) + '\n'
        outfile.write(item)
    outfile.close()
    print('The data were written to Population.txt.')

main()
```

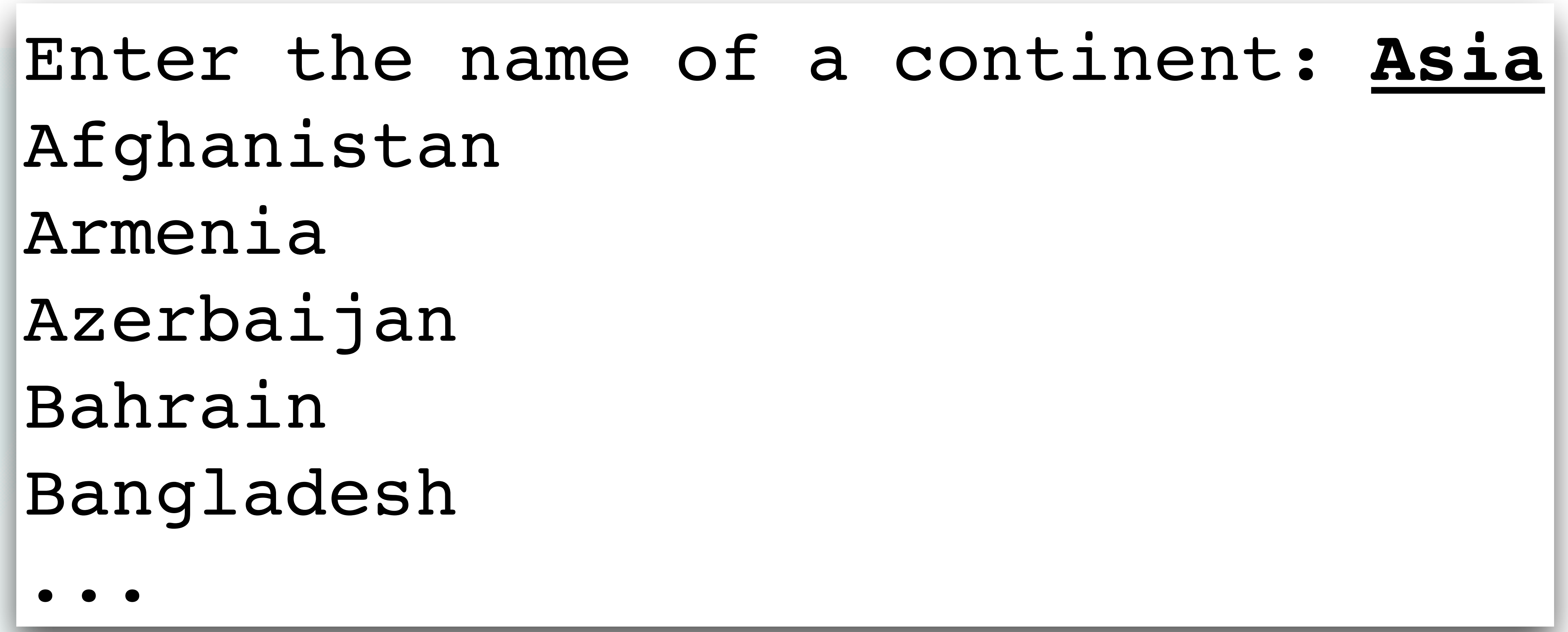


```
China,Asia,1355.7,3696100
India,Asia,1236.3,1269210
United States,North America,318.9,3794066

```

15-5.py



```python
def main():

    ## Display the countries in a specifed continent.
    continent = input("Enter the name of a continent: ")
    continent = continent.title()   # Allow for all lowercase letters.
    if continent != "Antarctica":
        infile = open("UN.txt", 'r')
        for line in infile:
            data = line.split(',')
            if data[1] == continent:
                print(data[0])
    else:
        print("There are no countries in Antarctica.")

main()
```



```
Enter the name of a continent: Asia
Afghanistan
Armenia
Azerbaijan
Bahrain
Bangladesh
...
```

CSV module

- The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel.

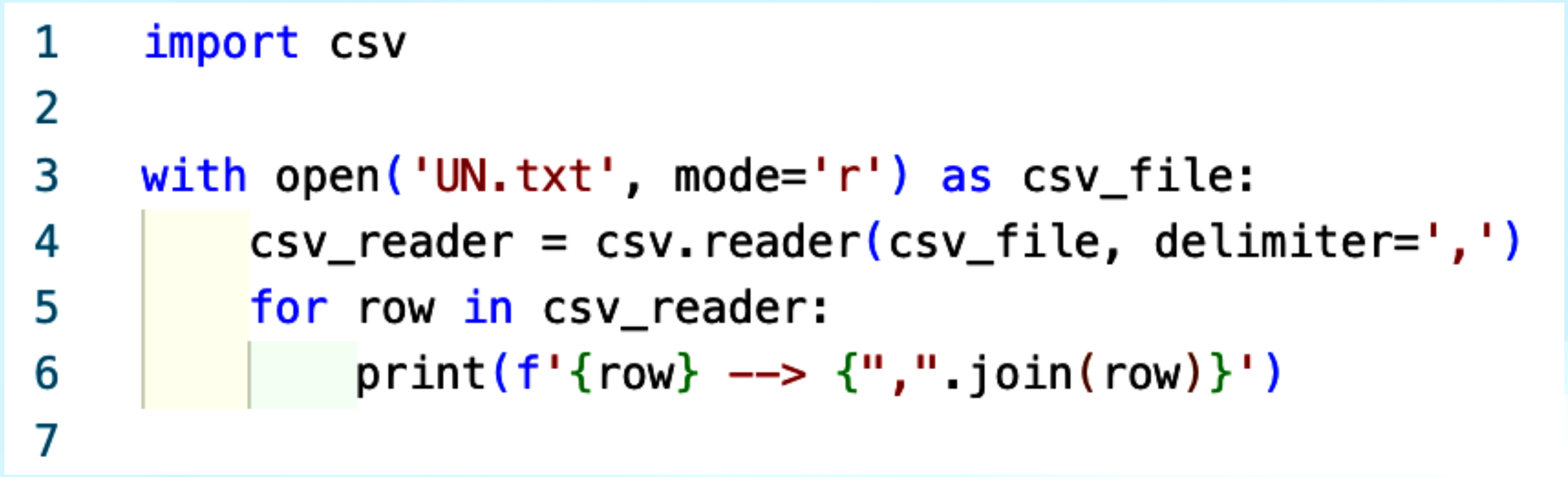
- The csv module’s reader and writer objects read and write sequences.

https://docs.python.org/3/library/csv.html

CSV module (cont'd.)

- `csv.reader(csvfile, dialect='excel', **fmtparams)`
  - Return a reader object which will iterate over lines in the given csvfile. csvfile can be any object which supports the iterator protocol and returns a string each time its `__next__()` method is called — file objects and list objects are both suitable.
  - Read more: https://docs.python.org/3/library/csv.html

15-6.py



```python
import csv

with open('UN.txt', mode='r') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        print(f'{row} --> {",".join(row)}')
```

```
['Afghanistan', 'Asia', '31.8', '251772'] --> Afghanistan,Asia,31.8,251772
['Albania', 'Europe', '3.0', '11100'] --> Albania,Europe,3.0,11100
['Algeria', 'Africa', '38.3', '919595'] --> Algeria,Africa,38.3,919595
['Andorra', 'Europe', '.085', '181'] --> Andorra,Europe,.085,181
['Angola', 'Africa', '19.1', '481354'] --> Angola,Africa,19.1,481354
...
```

The with Statement Approach

- Context managers allow you to allocate and release resources precisely when you want to. The most widely used example of context managers is

  the `with` statement. Suppose you have two related operations which you’d like to execute as a pair, with a block of code in between. Context managers allow you to do specifically that. For example:

```
with open('some_file', 'w') as opened_file:
    opened_file.write('Hola!')
```

- The above code opens the file, writes some data to it and then closes it. If an error occurs while writing the data to the file, it tries to close it.

https://book.pythontips.com/en/latest/context_managers.html

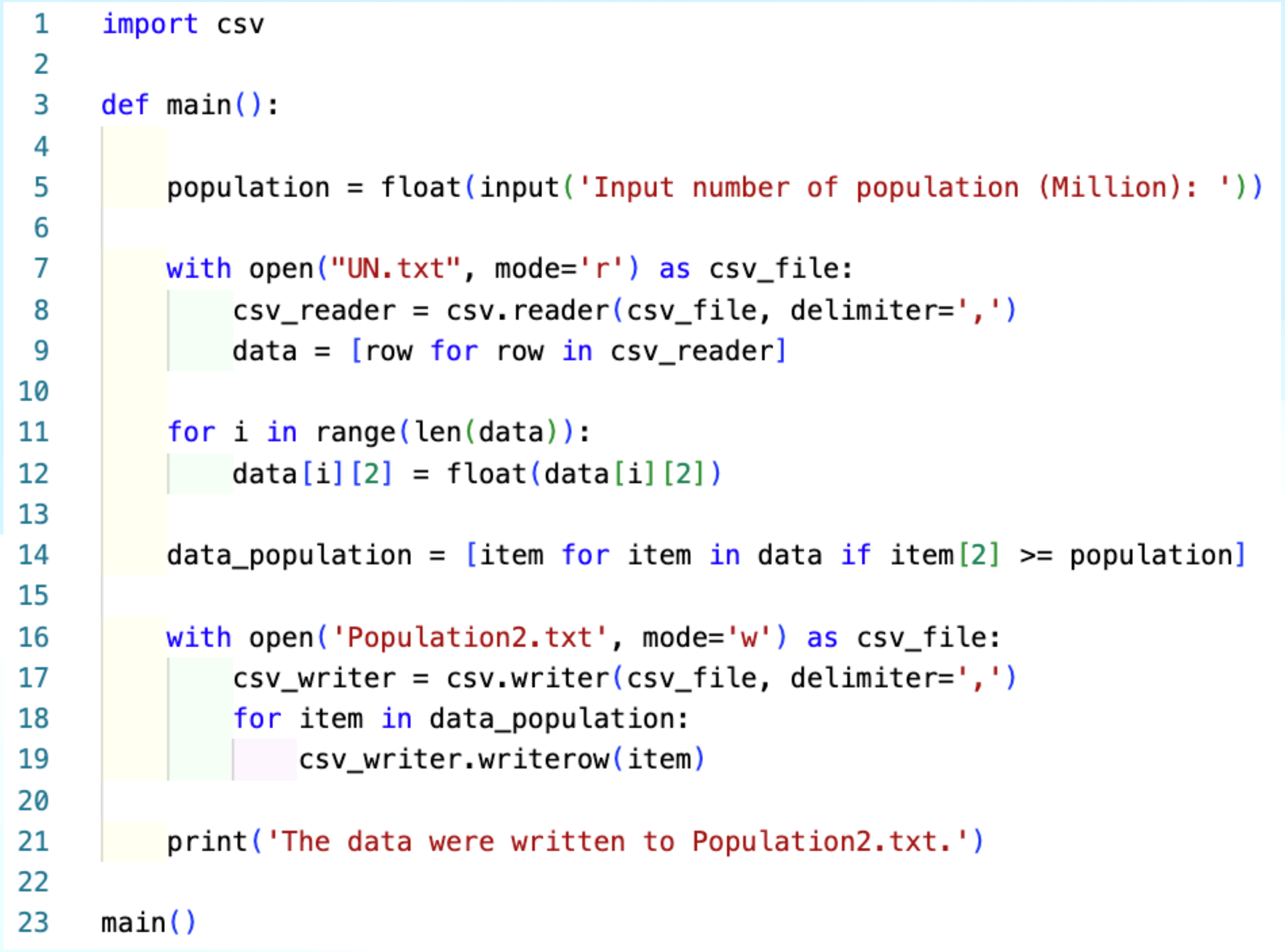
CSV module (cont’d.)

- `csv.writer(csvfile, dialect='excel', **fmtparams)`
  - Return a writer object responsible for converting the user’s data into delimited strings on the given file-like object. csvfile can be any object with a `write()` method.
  - Read more: https://docs.python.org/3/library/csv.html

writerow() and writerows() method

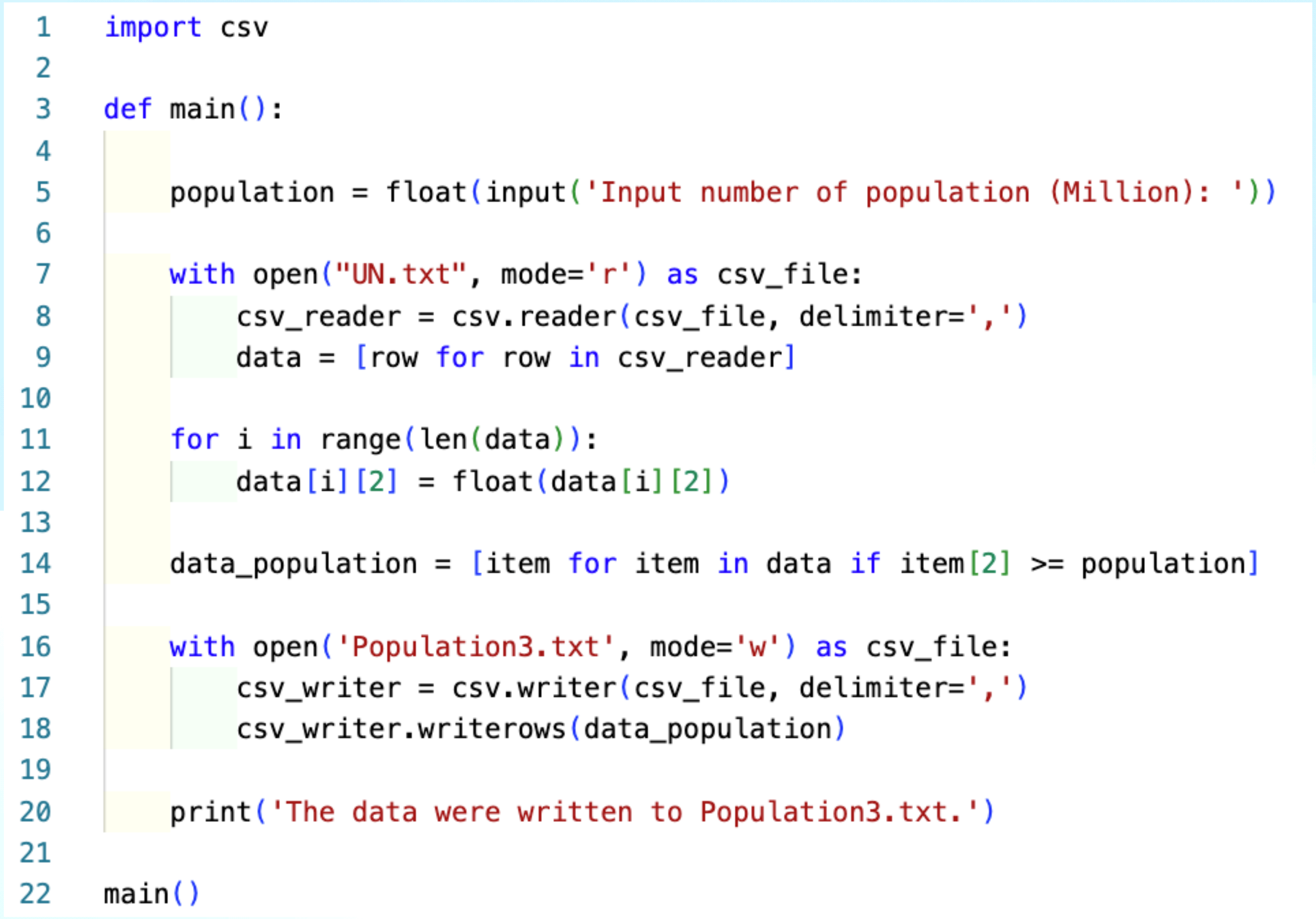
- writerow() method
  - This method writes items in an iterable (list, tuple or string), separating them by comma character.
- writerows() method
  - This method takes a list of iterables, as parameter and writes each item as a comma separated line of items in the file.

15-7.py



```python
import csv

def main():

    population = float(input('Input number of population (Million): '))

    with open("UN.txt", mode='r') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        data = [row for row in csv_reader]

    for i in range(len(data)):
        data[i][2] = float(data[i][2])

    data_population = [item for item in data if item[2] >= population]

    with open('Population2.txt', mode='w') as csv_file:
        csv_writer = csv.writer(csv_file, delimiter=',')
        for item in data_population:
            csv_writer.writerow(item)

    print('The data were written to Population2.txt.')

main()
```

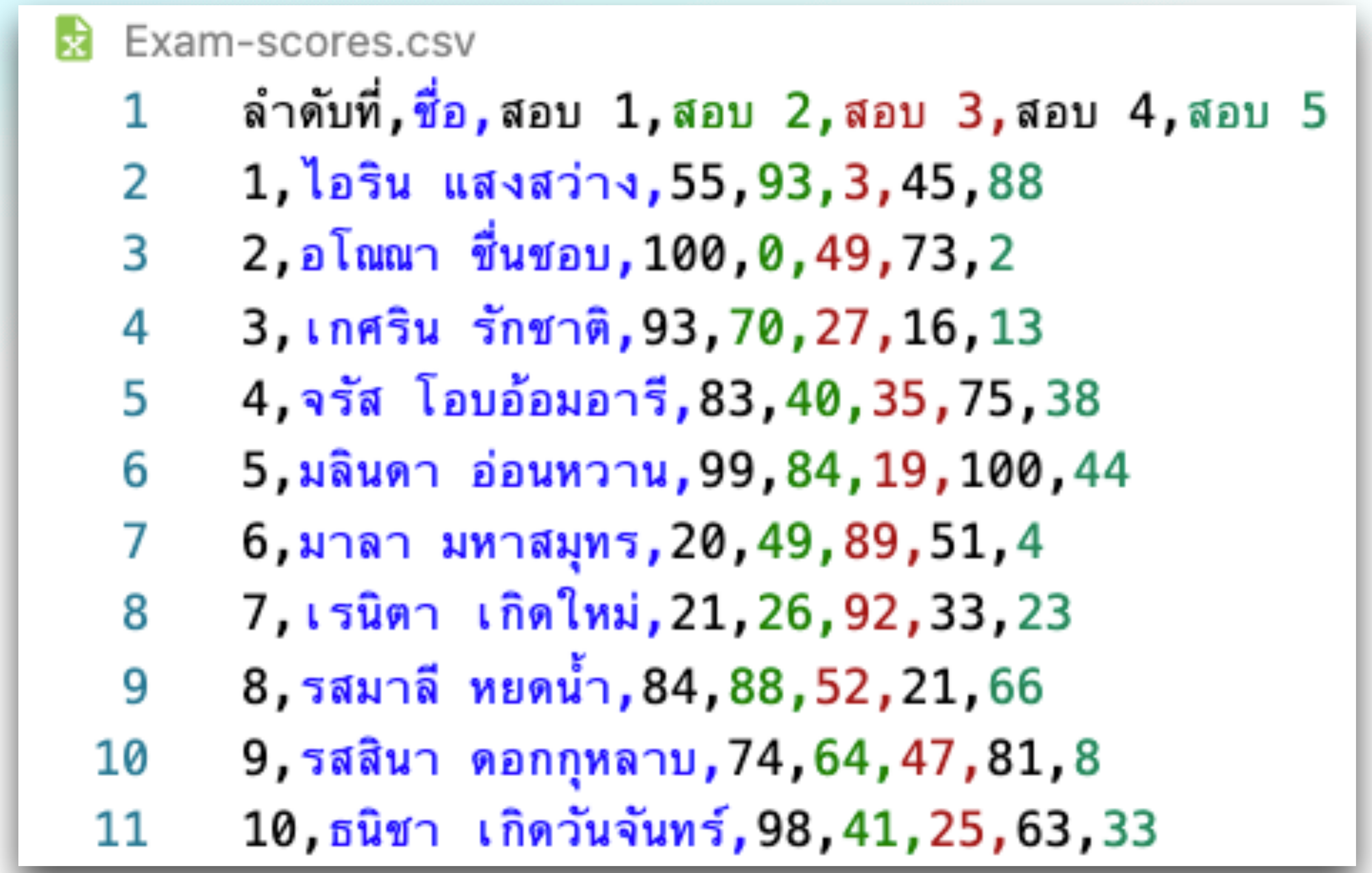
15-8.py



```python
import csv

def main():

    population = float(input('Input number of population (Million): '))

    with open("UN.txt", mode='r') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        data = [row for row in csv_reader]

    for i in range(len(data)):
        data[i][2] = float(data[i][2])

    data_population = [item for item in data if item[2] >= population]

    with open('Population3.txt', mode='w') as csv_file:
        csv_writer = csv.writer(csv_file, delimiter=',')
        csv_writer.writerows(data_population)

    print('The data were written to Population3.txt.')

main()
```

การเปิดอ่านและเขียนไฟล์ที่เป็นข้อมูลภาษาไทย

- Sample spreadsheet: https://siam.icu/exam-scores
- Sample CSV file: https://siam.icu/exam-scores-csv



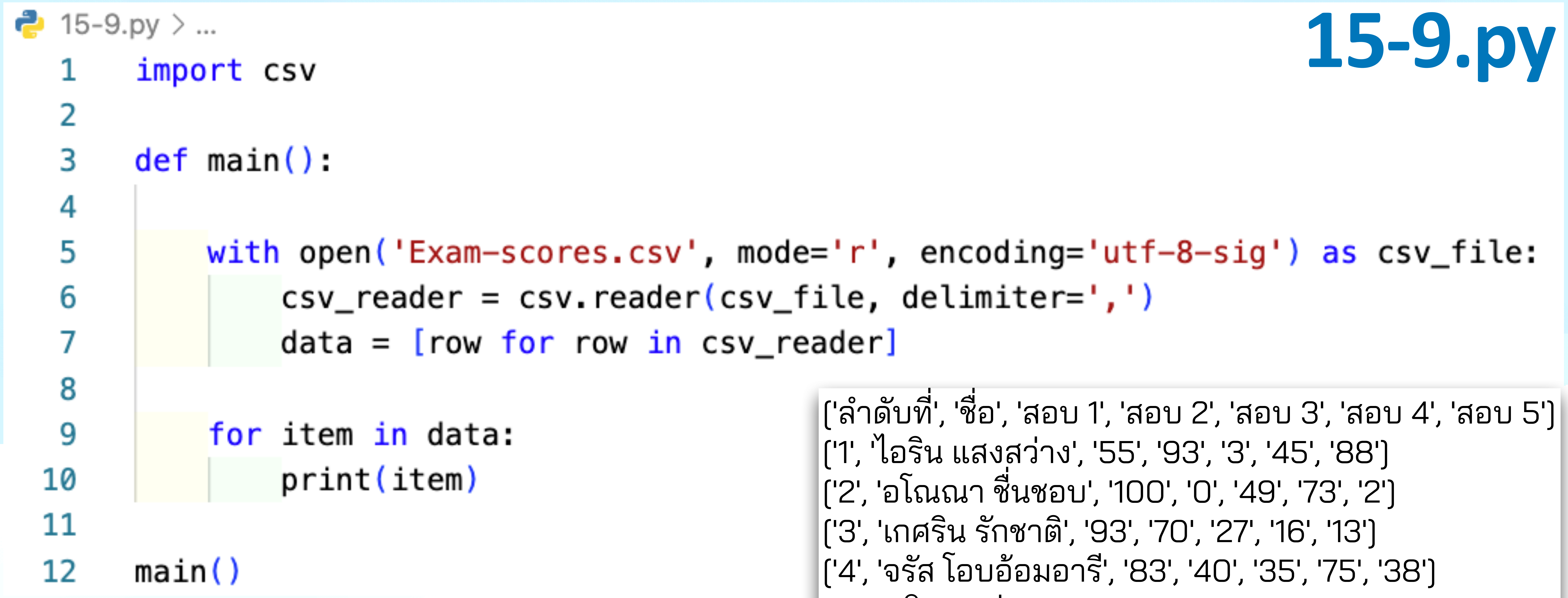
Exam-scores.csv

```
ลำดับที่,ชื่อ,สอบ 1,สอบ 2,สอบ 3,สอบ 4,สอบ 5
1,ไอริน แสงสว่าง,55,93,3,45,88
2,อโณฌา ชื่นชอบ,100,0,49,73,2
3,เกศริน รักชาติ,93,70,27,16,13
4,จรัส โอบอ้อมอารี,83,40,35,75,38
5,มลินดา อ่อนหวาน,99,84,19,100,44
6,มาลา มหาสมุทร,20,49,89,51,4
7,เรนิตา เกิดใหม่,21,26,92,33,23
8,รสมาลี หยดน้ำ,84,88,52,21,66
9,รสสินา ดอกกุหลาบ,74,64,47,81,8
10,ธนิชา เกิดวันจันทร์,98,41,25,63,33
```

การเปิดอ่านและเขียนไฟล์ที่เป็นข้อมูลภาษาไทย (ต่อ)

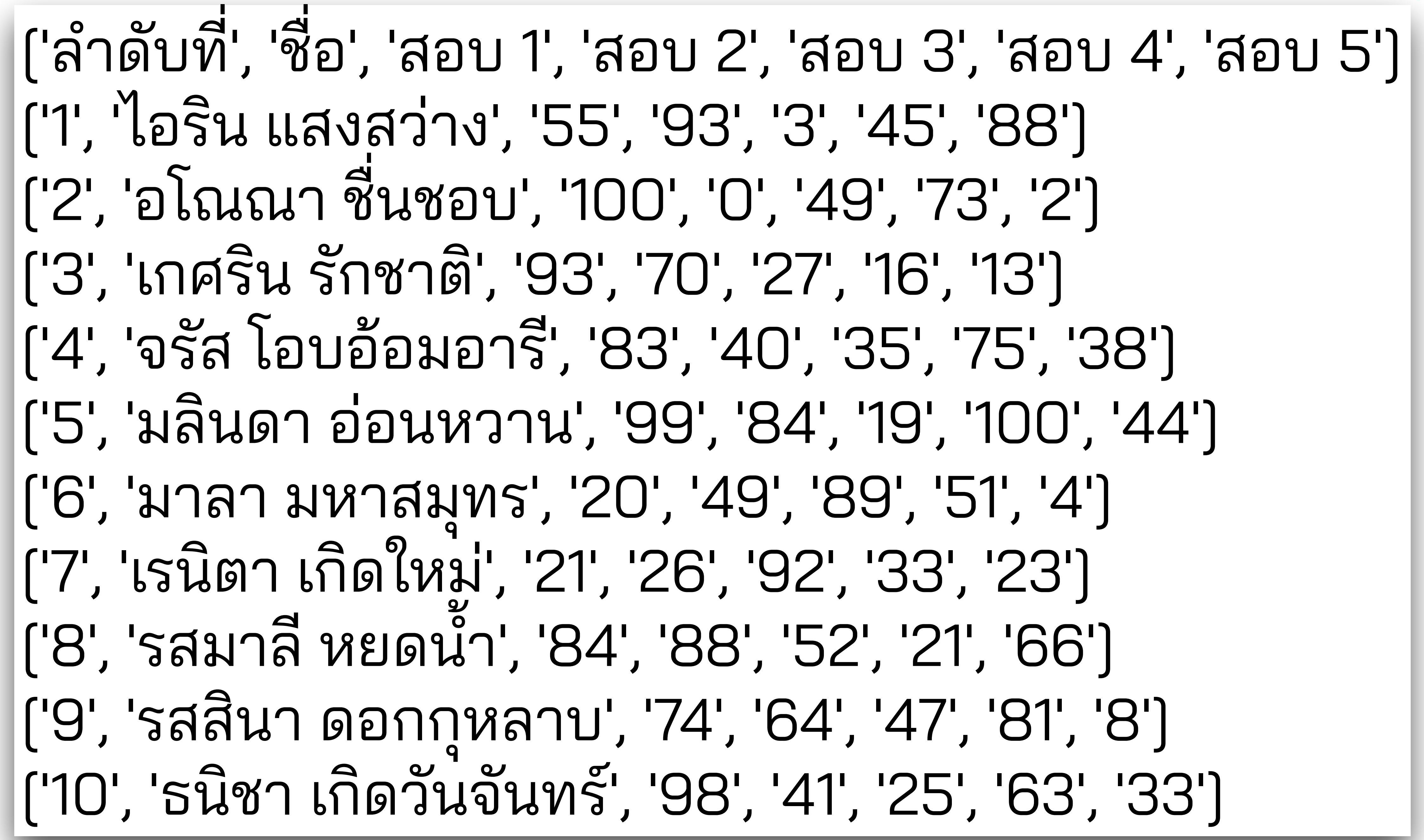
- ให้เพิ่ม encoding='utf-8-sig' ใน open() เช่น
  - `infile = open('Exam-scores.csv', 'r', encoding='utf-8-sig')`
  - `with open('Exam-scores.csv', 'r', encoding='utf-8-sig') as csv_file:`

15-9.py



15-9.py > ...

```python
import csv

def main():

    with open('Exam-scores.csv', mode='r', encoding='utf-8-sig') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        data = [row for row in csv_reader]

    for item in data:
        print(item)

main()
```



```
['ลำดับที่', 'ชื่อ', 'สอบ 1', 'สอบ 2', 'สอบ 3', 'สอบ 4', 'สอบ 5']
['1', 'ไอริน แสงสว่าง', '55', '93', '3', '45', '88']
['2', 'อโณณา ชื่นชอบ', '100', '0', '49', '73', '2']
['3', 'เกศริน รักชาติ', '93', '70', '27', '16', '13']
['4', 'จรัส โอบอ้อมอารี', '83', '40', '35', '75', '38']
['5', 'มลินดา อ่อนหวาน', '99', '84', '19', '100', '44']
['6', 'มาลา มหาสมุทร', '20', '49', '89', '51', '4']
['7', 'เรนิตา เกิดใหม่', '21', '26', '92', '33', '23']
['8', 'รสมาลี หยดน้ำ', '84', '88', '52', '21', '66']
['9', 'รสสินา ดอกกุหลาบ', '74', '64', '47', '81', '8']
['10', 'ธนิชา เกิดวันจันทร์', '98', '41', '25', '63', '33']
```