University of
South Australia

Problem Solving and Programming

Week 11 – File Processing: Input and Output

Python Books

Course Textbook

Gaddis, Tony. 2012, *Starting Out with Python*, 2nd edition, Pearson Education, Inc.

Free Electronic Books

There are a number of good free on-line Python books. I recommend that you look at most and see if there is one that you enjoy reading. I find that some books just put me to sleep, while others I enjoy reading. You may enjoy quite a different style of book to me, so just because I say I like a book does not mean it is the one that is best for you to read.

- The following three books start from scratch - they don't assume you have done any prior programming:
  - The free on-line book "**How to think like a Computer Scientist: Learning with Python**", by Peter Wentworth, Jeffrey Elkner, Allen B. Downey and Chris Meyers, provides a good introduction to programming and the Python language. I recommend that you look at this book.
  - There is an on-line book "**A Byte of Python**" that is quite reasonable. See the home page for the book, or you can go directly to the on-line version for Python 3, or download a PDF copy of the book. This book is used in a number of Python courses at different universities and is another I recommend you look at.
  - Another good on-line book is "**Learning to Program**" by Alan Gauld. You can download the whole book in easy to print PDF format, and this is another book that would be good for you to look at.
- If you have done some programming before, you may like to look at the following:
  - **The Python Tutorial** - this is part of Python's documentation and is updated with each release of Python. This is not strictly an e-Book, but is book-sized.
  - **Dive into Python 3**, by Mark Pilgrim is a good book for those with some programming experience. I recommend you have a look at it. You can download a PDF copy.

UniSA

Problem Solving and Programming

- More on writing programs:
  - File Input and Output

File Input and Output

- In many problems, we will want to read information from a text file and/or write information to a text file.

- We will look at how to:
  - Open a text file.
  - Read information from a text file.
  - Write information to a text file.
  - Close a text file.

File Input and Output

- Open a text file.
  - The `open()` function returns a file object on a successful opening of the file or else results in an error situation.
    - When a failure occurs, Python generates or raises an `IOError` exception (mentioned later in the course).
  - The basic syntax of the `open()` function is:

```
file_object = open(file_name, access_mode='r')
```

    - The `file_name` is a string containing the name of the file to open.
    - The `access_mode` optional variable is also a string, consisting of a set of flags indicating which mode to open the file with.
      - This describes the way in which the file will be used.
      - The `access_mode` can be:
        - `'r'` when the file will be read (r will be assumed if it's omitted)
        - `'w'` when the file will be written to (existing file with same name will be erased)
        - `'a'` opens the file for appending

File Input and Output

- For example:
  - Create a file object and assign it a variable with `open()`.
  - You can open a file for reading or writing – for example:

```
infile = open("input.txt", "r")

outfile = open("output.txt", "w")
```

If you want to provide a full path for the file name. Remember to escape '\' characters if they are used in the operating system you are using.

File Input and Output

- Read information from a text file.
  - Use Input Methods of File Objects
    - The `read()` method is used to read the whole file into a string.
    - The `readline()` method reads one line of the open file into a string.
      - Reads until a newline character (\n) is encountered.
      - The line, including the newline character is returned as a string.
      - If `readline()` returns an empty string, the end of the file has been reached, while a blank line is represented by '\n', a string containing only a single newline.
    - The `readlines()` method does not return a string like the other two input methods. Instead, it returns a list containing all the lines of the data in the file.
      - It reads all lines and returns them as a list of strings.

File Input and Output

- Read information from a text file.
  - The `read()` method is used to read the whole file into a string.
  - For example:

```
infile = open("input.txt", "r")

string = infile.read()
print(string)

infile.close()
```

`input.txt` file:

```
this is line 1
this is line 2
this is line 3
this is line 4
this is line 5
```

Output:

```
this is line 1
this is line 2
this is line 3
this is line 4
this is line 5
```

UniSA

File Input and Output

- Read information from a text file.
  - The `readline()` method reads one line of the open file into a string.
  - For example:

```
infile = open("input.txt", "r")

string = infile.readline()
print(string)

infile.close()
```

`input.txt` file:

```
this is line 1
this is line 2
this is line 3
this is line 4
this is line 5
```

Output:

```
this is line 1
```

UniSA

File Input and Output

- Read information from a text file.
  - The `readlines()` method does not return a string like the other two input methods. Instead, it returns a list containing all the lines of the data in the file.
  - For example:

```
infile = open("input.txt", "r")

stringList = infile.readlines()
print(stringList)

infile.close()
```

`input.txt` file:

```
this is line 1
this is line 2
this is line 3
this is line 4
this is line 5
```

Output:

```
['this is line 1\n', 'this is line 2\n', 'this is line 3\n',
'this is line 4\n', 'this is line 5\n']
```

UniSA

File Input and Output

- Read information from a text file.
  - The `readlines()` method does not return a string like the other two input methods. Instead, it returns a list containing all the lines of the data in the file.
  - For example:

`input.txt` file:

```
this is line 1
this is line 2
this is line 3
this is line 4
this is line 5
```

```
infile = open("input.txt", "r")

stringList = infile.readlines()

for line in stringList:
    print(line, end='')    #or…  print(line[:-1])

infile.close()
```

Output:

```
this is line 1
this is line 2
this is line 3
this is line 4
this is line 5
```

UniSA

File Input and Output

- Read information from a text file – using loops to process files.
  - Reading a file with a loop and detecting the end of the file.
  - An alternative approach to reading lines is to loop until the end of file is encountered.
  - Quite often a program must read the contents of a file without knowing the number of items that are stored in the file.
    - Need a way of knowing when the end of the file has been reached.
  - The `readline` method returns an empty string (`''`) when it has attempted to read beyond the end of a file.
    - We can then write a while loop that determines when the end of a file has been reached. That is, going through a file line by line until the end of file has been reached.

File Input and Output

- Read information from a text file – using loops to process files.
  - Reading a file with a loop and detecting the end of the file. That is, processing a file line by line.
  - For example:

```
infile = open("input.txt", "r")
# read the first line from the file
line = infile.readline()
# While not end of file, keep looping. i.e. as long as
# an empty string is not returned from readline,
# continue processing.
while line != '':
    print(line)
    # Read the next line.
    line = infile.readline()
infile.close()
```

    - Where `line` represents a single line of the text file.
    - Keeps looping *until* the end of file has been reached.

File Input and Output

- Read information from a text file – using loops to process files.
  - An alternative approach to reading lines and the while loop we saw in the last few slides, is to loop over the file object.
  - Python allows you to use a `for` loop that automatically reads lines in a file without testing for any special condition that signals the end of the file. That is, processing a file line by line using a for loop.
  - The loop automatically stops when the end of the file has been reached.
  - For example:

```
infile = open("input.txt", "r")
for eachLine in infile:
    print(eachLine)
infile.close()
```

    - Where `eachLine` represents a single line of the text file.

File Input and Output

- Write information to a text file.
  - Output Methods of File Objects
    - The `write()` method takes a string and writes the data to the file.

```
outfile = open("output.txt", "w")

outfile.write("This is line 7")

outfile.close()
```

    - To write something other than a string, it needs to be converted to a string first (i.e. can use the built-in `str()` function to do so).

File Input and Output

- Write information to a text file – using a `while` loop.

```
outfile = open("output.txt", "w")

word_list = ['one', 'two', 'three', 'four', 'five', 'six']

index = 0
while index < len(word_list):
    outfile.write(word_list[index] + '\n')
    index +=1

outfile.close()
```

Output file contents:

```
one
two
three
four
five
six
```

File Input and Output

- Write information to a text file – using a `for` loop.

```
outfile = open("output.txt", "w")

word_list = ['one', 'two', 'three', 'four', 'five', 'six']

for word in word_list:
    outfile.write(word + '\n')

outfile.close()
```

Output file contents:

```
one
two
three
four
five
six
```

File Input and Output

- Write information to a text file – another example using the `str()` built-in function to convert to a string first.

```
outfile = open("output.txt", "w")

word_list = ['one', 'two', 'three', 'four', 'five', 'six']

index = 0
while index < len(word_list):
    outfile.write(str(index+1) + ' ' + word_list[index] + '\n')
    index +=1

outfile.close()
```

Output file contents:

```
1 one
2 two
3 three
4 four
5 five
6 six
```

File Input and Output

- Close a text file.
  - When you are done with a file, call the `close()` method to close it.
  - The `close()` method completes access to a file by closing it.
  - For example:

```
infile.close()
```

File Input and Output - Example

1.1 Hashmat the brave warrior

Hashmat is a brave warrior who with his group of young soldiers moves from one place to another to fight against his opponents. Before fighting he just calculates one thing, the difference between his soldier number and the opponent's soldier number. From this difference he decides whether to fight or not. Hashmat's soldier number is never greater than his opponent.

Input

The input contains two integer numbers in every line. These two numbers in each line denotes the number of soldiers in Hashmat's army and his opponent's army or vice versa.

Output

For each line of input, print the difference of number of soldiers between Hashmat's army and his opponent's army. Each output should be on separate line.

Sample Input:

10 12
10 14
100 200

Sample Output:

2
4
100

UniSA

File Input and Output - Example

1.1 Hashmat the brave warrior

```
infile = open("had.txt", 'r')

stringList = infile.readlines()

for string in stringList:
    strSplit = string.split()

    n1 = int(strSplit[0])
    n2 = int(strSplit[1])

    diff = abs(n2 - n1)
    print(diff)

infile.close()
```

`had.txt` file:

```
10 12
10 14
100 200
```

Output:

```
2
4
100
```

UniSA

End Week 11