Python Input/Output

Input

`Python` is able to accept input from the keyboard or from files.

1. Keyboard input

a. raw_input():

this takes user input from the keyboard as a single string variable

Key arguments

- prompt: any text to prompt user for keyboard input

2. File input

a. open():

This function opens a file directly. The arguments are the file name and mode (read, write, etc.). Once a file is open the user has a number of options to access the contents of that file:

for loop

This approach replicates how we interacted with files in `Unix`. Each time through the for loop, the index variable takes on the value of a line of the opened file.

```
# open the file we want to work with
InFile = open("wages.csv",'r')

# a counter for the line number; this can be useful for a number of reasons
LineNumber=0

for line in InFile:
  # skip the header line of a file
  if LineNumber>0:
    # remove the end of line character '\n'
    line=line.strip()
    # print the line
    print line

    # split the fields into a list using the comma delimiter
    fieldsList=line.split(',')

  # increment the line number counter
  LineNumber=LineNumber + 1

# close the file after the loop is finished
InFile.close()
```

readLine

Another option is the `.readline()` function, which will read the next line of the file each time it is called. This can be placed in a loop and work just like the previous "for loop" example or if you were only interested in the first line or first few lines of a file you could use this function to read only the lines you care about.

```
# open the file we want to work with
InFile = open("wages.csv",'r')

firstLine=InFile.readline()

secondLine=InFile.readline()

InFile.close()
```

readLines

This function reads all lines of a file into a list. It actually uses `.readline()` to do so.

```
# open the file we want to work with
InFile = open("wages.csv",'r')

fileContents=InFile.readlines()

# close the file after the loop is finished
InFile.close()
```

b. numpy.loadtxt():

This function reads tabular data into an array. Because the data is read into an array, the data type has to be the same for all entries.

Key arguments

- fname: what file to read from
- dtype: the data type in the file; default is float
- comments: characters used to indicate comment lines that shouldn't be read from the file
- delimiter: what character separates the columns in the file

c. pandas.read_csv():

this function reads tabular content from a file into a Dataframe structure in the pandas package. This data structure allows for multiple data types.

Key arguments

- filepath_or_buffer: what file to read from
- sep: delimiter to use
- header: row of file to use as the header
- names: a list of header names to use if the file lacks header names
- index_col: column to use as row labels

Output

`Python` is also capable of outputting information to the screen or file.

1. Screen output

a. print():

This is a function for writing to the command line

- in IPython, users can print an object simply by typing the variable name

2. File output

a. open():

Similar to how we used `open()` to read from files, we can use it to write to files. A file can be opened in "write" mode and any contents will be overwritten or in "append" mode and any contents will be appended. This is analogous to `>` and `>>` in `Unix`. The `.write()` function is then used to send information to the open file for writing.

```
# open the file we want to write information to
# the 'w' opens the file for writing; 'a' indicates append mode
OutFile = open("summary.txt",'a')

text="More information for our file"

OutFile.write(text+"\n")
# notice that we need to explicitly add a newline character
#to give anything we write to the file its own line

# close the file after the loop is finished
OutFile.close()
```

b. numpy.savetxt():

This function writes a numpy array to a text file.

Key arguments

- fname: the filename to write to
- X: the array to be written to the file
- delimeter: character to separte columns in the text file

c. pandas.to_csv():

This function writes a pandas Dataframe to a delimited text file.

Key arguments

- path_or_buf: filename to be used
- sep: delimiter for text file; defaults to space delimited

- na_rep: how to represent missing data in the text file
- header: write out column names
- index: write out row names
- mode: write ('w') or append ('a')

File directory navigation

On OSX and Linux systems, IPython and Rodeo allow for direct interaction with the file system using `Unix` commands, including `pwd`, `ls`, and `cd`. A library called `OS` exists to allow for interaction with the operating system on any platform , including Windows.

```
import os

os.listdir('.') # analogous to ls

os.getcwd() # analogous to pwd

os.chdir("path/to/directory") # analogous to cd
```