Brief Python

Python Course for Programmers

Learn Python Language for Data Science

CHAPTER 5: TEXT PROCESSING (STRING AND FILE)

DR. ERIC CHOU

IEEE SENIOR MEMBER

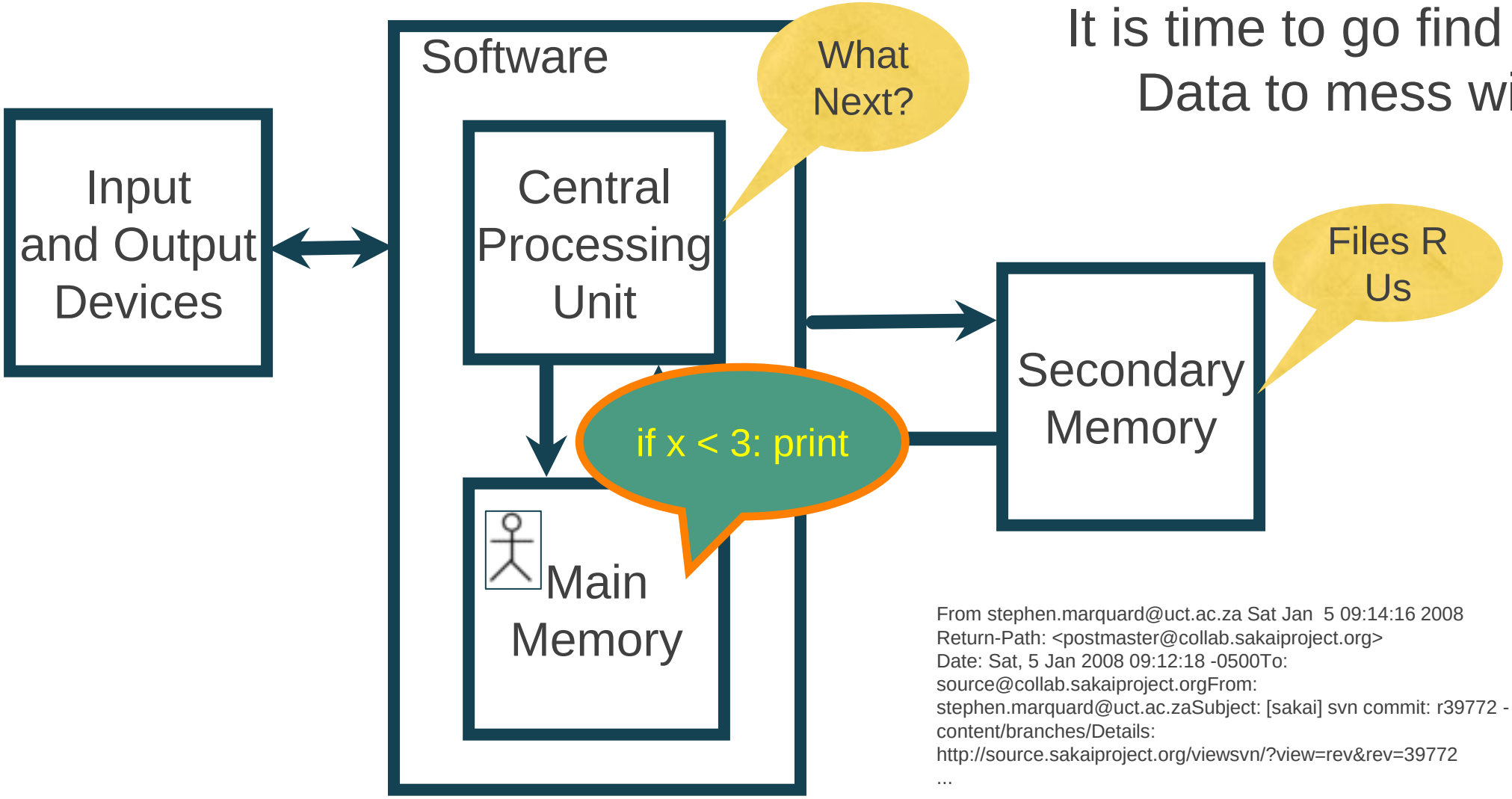
Objectives

- File Processing (Chapter 4 String and File)
- Tokenization (Chapter 5 Tokenization)
- Define EBNF and SD (Chapter 5)
- Regular Expression (Chapter 6)

File Handler

LECTURE 1

It is time to go find some Data to mess with!



From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Date: Sat, 5 Jan 2008 09:12:18 -0500To: source@collab.sakaiproject.orgFrom: stephen.marquard@uct.ac.zaSubject: [sakai] svn commit: r39772 - content/branches/Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772
...

File Processing

- A text file can be thought of as a sequence of lines

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Date: Sat, 5 Jan 2008 09:12:18 -0500
To: source@collab.sakaiproject.org
From: stephen.marquard@uct.ac.za
Subject: [sakai] svn commit: r39772 - content/branches/

Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772
```

http://www.py4e.com/code/mbox-short.txt

Opening a File

- Before we can read the contents of the file, we must tell Python which file we are going to work with and what we will be doing with the file
- This is done with the open() function
- open() returns a “file handle” - a variable used to perform operations on the file
- Similar to “File -> Open” in a Word Processor

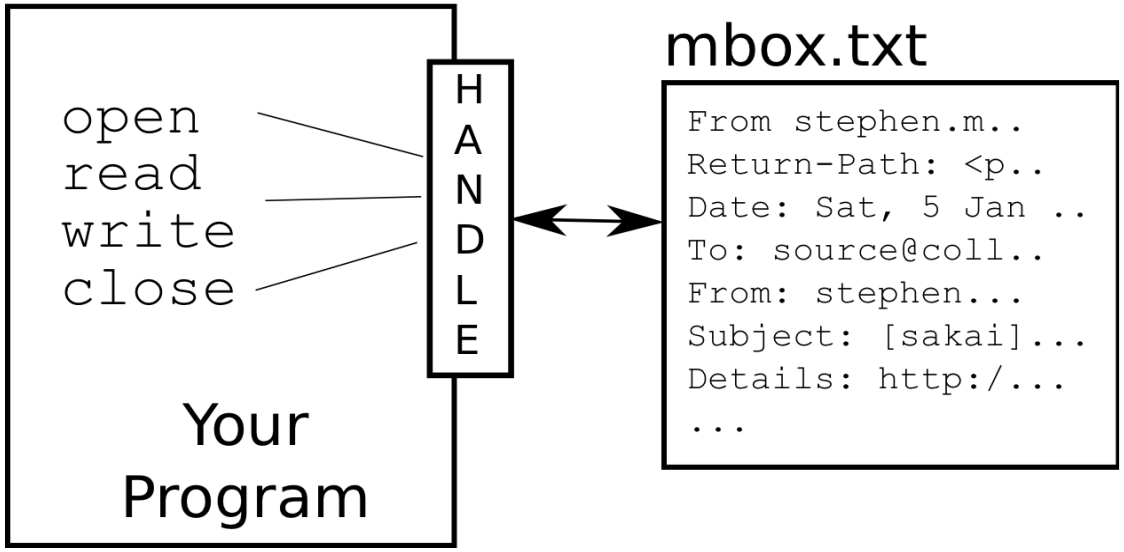
Using open()

```
fhand = open('mbox.txt', 'r')
```

- handle = open(filename, mode)
- returns a handle use to manipulate the file
- filename is a string
- mode is optional and should be 'r' if we are planning to read the file and 'w' if we are going to write to the file

What is a Handle?

```
>>> fhand = open('mbox.txt')
>>> print(fhand)
<_io.TextIOWrapper name='mbox.txt' mode='r' encoding='UTF-8'>
```



When Files are Missing

```
>>> fhand = open('stuff.txt')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or
directory: 'stuff.txt'
```

The newline Character

- We use a special character called the "newline" to indicate when a line ends
- We represent it as \n in strings
- Newline is still one character - not two

```
>>> stuff = 'Hello\nWorld!'
>>> stuff
'Hello\nWorld!'
>>> print(stuff)
Hello
World!
>>> stuff = 'X\nY'
>>> print(stuff)
X
Y
>>> len(stuff)
3
```

File Processing

A text file can be thought of as a sequence of lines

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Date: Sat, 5 Jan 2008 09:12:18 -0500
To: source@collab.sakaiproject.org
From: stephen.marquard@uct.ac.za
Subject: [sakai] svn commit: r39772 - content/branches/

Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772
```

File Processing

A text file has newlines at the end of each line

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008\n
Return-Path: <postmaster@collab.sakaiproject.org>\n
Date: Sat, 5 Jan 2008 09:12:18 -0500\n
To: source@collab.sakaiproject.org\n
From: stephen.marquard@uct.ac.za\n
Subject: [sakai] svn commit: r39772 - content/branches/\n
\n
Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772\n
```

Reading Files in Python

ACTIVITY

File Handle as a Sequence

- A file handle open for read can be treated as a sequence of strings where each line in the file is a string in the sequence
- We can use the for statement to iterate through a sequence
- Remember - a sequence is an ordered set

```
xfile = open('mbox.txt')
for cheese in xfile:
    print(cheese)
```

Counting Lines in a File

- Open a file read-only
- Use a for loop to read each line
- Count the lines and print out the number of lines

```
fhand = open('mbox.txt')
count = 0
for line in fhand:
    count = count + 1
print('Line Count:', count)
```

```
$ python open.py
Line Count: 132045
```

Reading the *Whole* File

- We can read the whole file (newlines and all) into a single string

```
>>> fhand = open('mbox-short.txt')
>>> inp = fhand.read()
>>> print(len(inp))
94626
>>> print(inp[:20])
From stephen.marquar
```

Searching Through a File

- We can put an if statement in our for loop to only print lines that meet some criteria

```
fhand = open('mbox-short.txt')
for line in fhand:
    if line.startswith('From:') :
        print(line)
```

What are all these blank lines doing here?

```
From: stephen.marquard@uct.ac.za

From: louis@media.berkeley.edu

From: zqian@umich.edu

From: rjlowe@iupui.edu
...
```

What are all these blank lines doing here?

- Each line from the file has a newline at the end
- The print statement adds a newline to each line

```
From: stephen.marquard@uct.ac.za\n
\n
From: louis@media.berkeley.edu\n
\n
From: zqian@umich.edu\n
\n
From: rjlowe@iupui.edu\n
\n
...
```

Searching Through a File (fixed)

- We can strip the whitespace from the right-hand side of the string using rstrip() from the string library
- The newline is considered "white space" and is stripped

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if line.startswith('From:') :
        print(line)
```

From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
....

Skipping with continue

- We can conveniently skip a line by using the continue statement

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if not line.startswith('From:') :
        continue
    print(line)
```

Using in to Select Lines

- We can look for a string anywhere in a line as our selection criteria

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if not '@uct.ac.za' in line :
        continue
    print(line)
```

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
X-Authentication-Warning: set sender to stephen.marquard@uct.ac.za using -f
From: stephen.marquard@uct.ac.za
Author: stephen.marquard@uct.ac.za
From david.horwitz@uct.ac.za Fri Jan  4 07:02:32 2008
X-Authentication-Warning: set sender to david.horwitz@uct.ac.za using -f...
```

Prompt for File Name

```
fname = input('Enter the file name:  ')
fhand = open(fname)
count = 0
for line in fhand:
    if line.startswith('Subject:') :
        count = count + 1
print('There were', count, 'subject lines in', fname)
```

Enter the file name: mbox.txt
There were 1797 subject lines in mbox.txt

Enter the file name: mbox-short.txt
There were 27 subject lines in mbox-short.txt

Bad File Names

```
fname = input('Enter the file name:  ')
try:
    fhand = open(fname)
except:
    print('File cannot be opened:', fname)
    quit()

count = 0
for line in fhand:
    if line.startswith('Subject:') :
        count = count + 1
print('There were', count, 'subject lines in', fname)
```

Enter the file name: mbox.txt
There were 1797 subject lines in mbox.txt

Enter the file name: na na boo boo
File cannot be opened: na na boo boo

Files

LECTURE 1

Python 3 Beginner's Reference Cheat Sheet

Main data types

boolean = *True / False*
integer = 10
float = 10.01
string = "123abc"
list = [value1, value2, ...]
dictionary = { key1:value1, key2:value2, ...}

Numeric operators

+	addition
-	subtraction
*	multiplication
/	division
**	exponent
%	modulus
//	floor division

Comparison operators

==	equal
!=	different
>	higher
<	lower
>=	higher or equal
<=	lower or equal

Boolean operators

and	logical AND
or	logical OR
not	logical NOT

Special characters

#	coment
\n	new line
\<char>	scape char

String operations

string[i]	retrieves character at position i
string[-1]	retrieves last character
string[i:j]	retrieves characters in range i to j

List operations

list = []	defines an empty list
list[i] = x	stores x with index i
list[i]	retrieves the item with index I
list[-1]	retrieves last item
list[i:j]	retrieves items in the range i to j
del list[i]	removes the item with index i

Dictionary operations

dict = {}	defines an empty dictionary
dict[k] = x	stores x associated to key k
dict[k]	retrieves the item with key k
del dict[k]	removes the item with key k

String methods

string.upper()	converts to uppercase
string.lower()	converts to lowercase
string.count(x)	counts how many times x appears
string.find(x)	position of the x first occurrence
string.replace(x,y)	replaces x for y
string.strip(x)	returns a list of values delimited by x
string.join(L)	returns a string with L values joined by string
string.format(x)	returns a string that includes formatted x

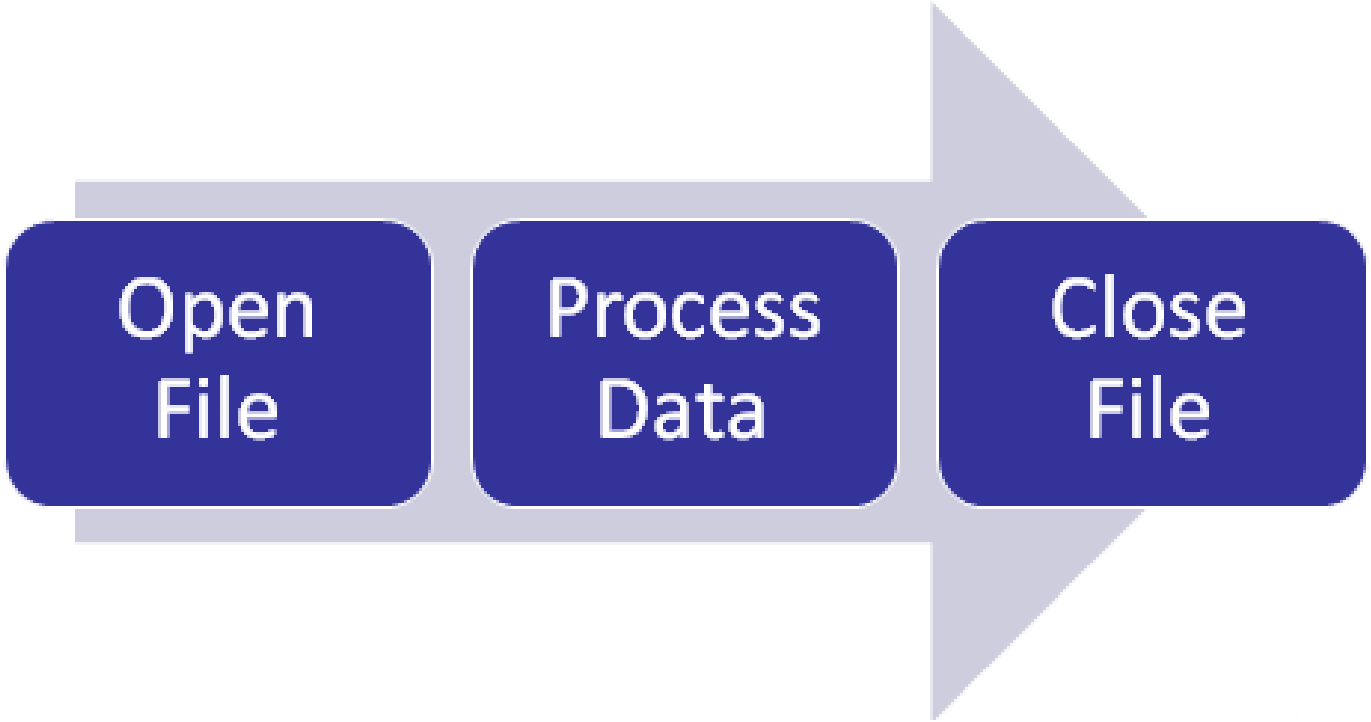
List methods

list.append(x)	adds x to the end of the list
list.extend(L)	appends L to the end of the list
list.insert(i,x)	inserts x at i position
list.remove(x)	removes the first list item whose value is x
list.pop(i)	removes the item at position i and returns its value
list.clear()	removes all items from the list
list.index(x)	returns a list of values delimited by x
list.count(x)	returns a string with list values joined by S
list.sort()	sorts list items
list.reverse()	reverses list elements
list.copy()	returns a copy of the list

Dictionary methods

dict.keys()	returns a list of keys
dict.values()	returns a list of values
dict.items()	returns a list of pairs (key,value)
dict.get(k)	returns the value associtated to the key k
dict.pop()	removes the item associated to the key and returns its value
dict.update(D)	adds keys-values (D) to dictionary
dict.clear()	removes all keys-values from the dictionary
dict.copy()	returns a copy of the dictionary

Legend: x,y stand for any kind of data values, **s** for a string, **n** for a number, **L** for a list where **i,j** are list indexes, **D** stands for a dictionary and **k** is a dictionary key.



File Access

Opening Files

- In Python the ‘open()’ function accepts a path to the file that you’d like to open along with a mode in which the file will be opened.
- The most commonly used modes are **read**, **write**, and **append**.
- This function creates a new **File** object which can then be iterated to extract or write information.

Open function

File Object Created

Path to File

Mode

```
f = open('c:\\temp\\data.txt",'r')
```

Python File **Open** Operation

```
f = open("test.txt")
f = open("test.txt", “r”)
f = open("test.txt", mode = 'r’, encoding = 'utf-8')
```

file name

file access mode

file text encoding

File Access Mode

LECTURE 1

File Access Mode

read	• r • r+ (read/write) – contents preserved
write	• w • w+ (read/write) - contents deleted
append	• a • a+ (read/write) – contents preserved
binary	• b • Opens file in Binary mode. Addition to r,w, or a
universal	• U • Addition to r,w,or a. Applies universal newline translator.

File Access Mode

The second parameter of the open function corresponds to a mode which is typically read (‘**r**’), write (‘**w**’), or append (‘**a**’).

- **Read Mode:** A value of **‘r’** indicates that you’d like to open the file for read only operations,
- **Write Mode:** A value of **‘w’** indicates you’d like to open the file for write operations.
  - Note: In the event that you open a file that already exists for write operations this will overwrite any data currently in the file so you must be careful with write mode.
- **Append Mode:** (**‘a’**) will open a file for write operations, but instead of overwriting any existing data it will append data to the end of the file.

File Access Mode

- Below you will find a list of all the available file modes. As I mentioned in a previous slide the most commonly used are read, write, and append.
- Read/Write Capability: However, you can also add the "+" to each of the modes to enable read/write capability. The contents of a file can be preserved or deleted depending upon the combination that you use.
- For example, w+ will open a file for read/write but the contents of the file are **deleted** while r+ **preserves** the contents of the file.
- **Binary Mode:** Adding a 'b' to r, w, or a will open a file in Binary mode.
- **Universal Mode:** Finally, the universal or 'U' character applies a universal newline translator.

File Read (char, token, line, block)

LECTURE 1

read()
read function with a specific number of bytes.

- **file.read(n)** - This method reads n number of characters from the file, or if n is blank it reads the entire file.
- **file.readline()** - This method reads an entire line from the text file.
- **file.readlines()** - This method reads an entire file into a list of line strings from the text file.
- **file.read()** – This method reads the entire file from a text file.
- Return 0 (False) when end of file.

File Access Pattern

File Access Pattern Algorithm:

```
Open_File
Read data from file to a buffer
while checking the buffer is valid:
      working on the buffer
      Read data from file to a buffer
```

Terminology:

File f: file handler

Buffer: ch, token(string), line, lines(list)

Read Functions: read(1), read(), readline(), readlines()

File Access Code in Python and Java

Python Pseudo Code:

```
f = open("filename.txt", "r")
buffer=f.read_function()
while buffer:
    processing(buffer)
    buffer=f.read_function()

# null return from the
# read_function is used to
# check the end_of_file
# condition
```

Java Pseudo Code:

```
File f = new File("filename.txt" );
Scanner in = new Scanner();
while (in.hasNext()){
    buffer=in.nextReading();
    processing(buffer);
}
```

Data File to be Read in

aa.txt

alpha\n

beta\n

gamma\n

delta\n

epsilon\n

null (0, or EOF)

Read File Character by Character: read(1)

Demo Program: file1.py

```
f = open("aa.txt", "r")
ch=f.read(1)
while ch:
    print(ch, end=" ")
    ch=f.read(1)
f.close()
```

```
Run fs4
C:\Python\Python36\python.exe
a l p h a
 b e t a
 g a m m a
 d e l t a
 e p s i l o n
```

Use space as separator so that we know the characters are read in one by one.

Read File Token by Token: readlines()

Demo Program: file2.py (One Token Per Line File)

```
f = open("aa.txt", "r")
lines = f.readlines()
for line in lines:
    line = line.strip()
    print(line, end=" ")
f.close()
```

```
Run fs2
C:\Python\Python36\python.exe
alpha beta gamma delta epsilon
```

Equivalent to .trim() in Java
Take out all of the white space characters (\n, \t, \f, space)

Read File Token by Token: read().split()

Demo Program: file3.py (Read the whole file into a string and split)

```
f = open("aa.txt", "r")
tokens=f.read().split()
for token in tokens:
    print("Token", token)
f.close()
```

Used to identify it is a token.

```
Run fs3
C:\Python\Python36\python.exe
Token alpha
Token beta
Token gamma
Token delta
Token epsilon
```

Note:

read(): read the whole file into a string.

read().split(): read the whole file into a string. Then split the string into a list of tokens (string)

readlines(): read the whole file into a list of lines.

Read File Line by Line: readline()

Demo Program: file4.py

```
f = open("aa.txt", "r")
line = f.readline()
while line:
    print(line, end="")
    line = f.readline()
f.close()
```

```
Run fs0
C:\Python\Python36\python.exe
alpha
beta
gamma
delta
epsilon
```

Read File as a whole: readlines()

Demo Program: file5.py (Read the whole file into a list of line strings.)

```
f = open("aa.txt", "r")
lines = f.readlines()     # lines is a list of line strings
print("Print file in list format: ")
print(lines)

all_lines = ""
for line in lines:
    all_lines += line

print("Print file as a long string: ")
print(all_lines)
f.close()
```

Output:

```
Print file in list format:
['alpha\n', 'beta\n', 'gamma\n', 'delta\n', 'epsilon\n', '\n', '\n']
Print file as a long string:
alpha
beta
gamma
delta
epsilon
```

Python String Functions

LECTURE 1

Python String Functions	Description
capitalize()	This method will convert the first character to Capitalize and following characters to Lowercase
casefold()	This method will return the given string in Lowercase.
center()	This method is used to Justify the string to Center and fill the remaining width with default white spaces
count()	This method Counts, How many times the string is occurred
encode()	This method returns the encoded version of a string object
endswith()	This method returns TRUE, if the string Ends with the specified substring
expandtabs()	This method returns a copy of the given string, where all the tab characters will be replaced with one or more spaces.
find()	It returns the **index** position of the first occurrence of a specified string. It will return -1, if the specified string is not found
format()	This method will be useful to **format** the string
format_map()	This method will be useful to format the string

Python String Functions	Description
index()	It returns the index position of the first occurrence of a specified string. It will raise ValueError, if the specified string is not found
isalnum()	This method returns TRUE, if the string contains **letters** and **numbers**
isalpha()	This method returns TRUE, if the string has at least one letter and all the letters are **Alphabetic**
isdecimal()	This method returns TRUE, if the string has at least one letter and all the letters are **Decimal**
isdigit()	This method returns TRUE, if the string has at least one letter and all the letters are **Digits**
isidentifier()	This method returns TRUE, if the string is valid identifier
islower()	This method returns TRUE, if the string has at least one letter and all the letters are in **Lowercase**
isnumeric()	This method returns TRUE, if the string has at least one letter and all the letters are **Numeric**
isprintable()	This method returns TRUE, if all the letters are Printable
isspace()	This method returns TRUE, if the string contains only white spaces
istitle()	This method returns TRUE, if the string has at least one letter and it is a Title.
isupper()	This method returns TRUE, if the string has at least one letter and all the letters are in **Uppercase**

Python String Functions	Description
join()	This method will be useful to Join (Concatenate) a list of strings
ljust()	This method is used to Justify the string to Left hand side and fill the remaining width with default white spaces
lower()	This method will convert the given string into **Lowercase** letters and return new string
lstrip()	It will remove the white spaces from Left hand side of a string
maketrans()	It returns the transaction table. We can further use this transaction in translate() method.
partition()	It partition the given string at the first occurrence of the specified separator and return a tuple with three arguments.
replace()	This method will search for specified string and replace it with new string value
rfind()	It returns the index position of the Last occurrence of a specified string. It will return -1, if the specified string is not found
rindex()	It returns the index position of the Last occurrence of a specified string. It will raise ValueError, if the specified string is not found
rjust()	This method is used to Justify the string to Right hand side and fill the remaining width with default white spaces
rpartition()	This method will partition the given string using the specified separator and return a tuple with three arguments.
rsplit()	This method will be useful to Split the string into list of strings, based on the specified delimiter. This will done from right to left
rstrip()	It will remove the white spaces from **Right** hand side of a string

Python String Functions	Description
split()	This method will be useful to Split the string into list of strings, based on the specified **delimiter**
splitlines()	It returns a list of lines in the given string by breaking the given string at line boundaries.
startswith()	This method returns TRUE, if the string Starts with the specified substring
strip()	It will **remove** the white spaces from both ends. Performs both lstrip() and rstrip()
swapcase()	This method will convert the Lowercase letters into Uppercase and Uppercase letters into Lowercase
title()	This method will convert the first character in each word to Uppercase and following characters to Lowercase
translate()	Returns a Copy of the given string in which each character has been mapped with the transaction table.
upper()	This method will convert the given string into **Uppercase** letters and return new string
zfill()	It returns a copy of the string filled with ASCII ‘0’ digits on the left hand side of the string to make a string length to specified width.

Summary

LECTURE 1

Summary

- Creation and closure of file handlers.
- Open file in different reading and writing modes
- String processing.