Files

Presented By

M.Malarmathi

AP/IT

- A text file is a sequence of characters stored on a permanent medium like a hard drive, flash memory, or CD-ROM

File Handling

File Opening:

- Before you can read or write a file, you have to open it using Python's built-in ***open()*** function.

file object = open(“filename.ext”, ”mode”)

Mode	**Description**
'r‘	Open a file for reading. (default)
'w‘	Open a file for writing. Overwrites the file if the file exits. Otherwise it creates a new file.
‘a’	Opens a file for appending
‘t’	Open in text mode
‘b’	Open in binary mode
+	Open file for updating (reading and writing)

File Handling Continues..

Writing a file:

Syntax: fileObject.write(string);

- Here, passed parameter is the content to be written into the opened file.
- The ***write()*** method writes any string to an open file.

Example:

```
fo = open("foo.txt", "w")
fo.write( "Python is a great language.\nYeah its great!!\n");
Print("success")
fo.close()
```

File Handling Continues..

Reading a file:

Syntax: fileObject.read([count]);

- count- passed parameter is the number of bytes to be read from the opened file.
- The ***read()*** method reads a string from an open file.

Example

```
fo = open("foo.txt", "r+")
str = fo.read(10);
print "Read String is : ", str
fo.close()
```

Output:

```
Read String is : Python is
```

File Handling Continues..

Methods	Description
read([number])	- Return specified number of characters from the file.If omitted Reads whole file at once.
readline()	- Reads one line each time from the file.
readlines()	- Reads all the lines from the file in a list.

Closing a file:

Syntax: fileObject.close();

- **close()** method flushes any unwritten information and closes the file object

Formatting Strings

- "%" operator and format() method are used to formatting strings.

<u>Number formatting with format()</u>

Example:

```
print(format(123, "d"))
print(format(123.45678, "f"))
print(format(12, "b"))
```

Formatting Strings continues

Old

- **'%s %s'** % ('one', 'two')

New

- '{} {}'.format('one', 'two')

Output: one two

New style formatting

'{1} {0}'.format(‘50', ‘80')

Output: 80 50

➢ This allows for re-arranging the order of display without changing the arguments.

Formatting Strings continues

Datetime:

- New style formatting also allows objects to control their own rendering.
- This operation is not available with old-style formatting.

Setup: **from datetime import** datetime

```
"{:%Y-%m-%d %H:%M}" .format(datetime(2005, 2, 3, 4, 5))
```

Output

```
2005-02-03 04:05
```

Thank You