Python File Handling

File handling is an important part of any software application. Python has several functions for creating, reading, updating, and deleting files.

Python File open() Method

The key function for working with files in Python is the `open()` function.

The `open()` function takes two parameters; *filename*, and *mode*.

There are four different methods (modes) for opening a file:

`"r"` - Read - Default value. Opens a file for reading, error if the file does not exist

`"a"` - Append - Opens a file for appending, creates the file if it does not exist

`"w"` - Write - Opens a file for writing, creates the file if it does not exist

`"x"` - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

`"t"` - Text - Default value. Text mode

`"b"` - Binary - Binary mode (e.g. images)

Syntax

To open a file for reading it is enough to specify the name of the file:

```
f = open("demofile.txt")
```

The code above is the same as:

```
f = open("demofile.txt", "rt")
```

Because `"r"` for read, and `"t"` for text are the default values, you do not need to specify them.

Note: Make sure the file exists, or else you will get an error.

Python File close() Method

Example

Close a file after it has been opened:

```
f = open("demofile.txt", "r")
print(f.read())
f.close()
```

Definition and Usage

The `close()` method closes an open file.

You should always close your files, in some cases, due to buffering, changes made to a file may not show until you close the file.

Syntax

```
file.close()
```

Parameter Values

No parameters

Create a New File

To create a new file in Python, use the `open()` method, with one of the following parameters:

`"x"` - Create - will create a file, returns an error if the file exist

`"a"` - Append - will create a file if the specified file does not exist

`"w"` - Write - will create a file if the specified file does not exist

Example

Create a file called "myfile.txt":

```
f = open("myfile.txt", "x")
```

Result: a new empty file is created!

Example

Create a new file if it does not exist:

```
f = open("myfile.txt", "w")
```

Python File write() Method

Example

Open the file with "a" for appending, then add some text to the file:

```
f = open("demofile2.txt", "a")
f.write("See you soon!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

Definition and Usage

The `write()` method writes a specified text to the file.

Where the specified text will be inserted depends on the file mode and stream position.

`"a"`: The text will be inserted at the current file stream position, default at the end of the file.

`"w"`: The file will be emptied before the text will be inserted at the current file stream position, default 0.

Syntax

```
file.write(byte)
```

Parameter Values

Parameter	Description
byte	The text or byte object that will be inserted.

Example

Open the file "demofile2.txt" and append content to the file:

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

Open the file "demofile3.txt" and overwrite the content:

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()

#open and read the file after the appending:
f = open("demofile3.txt", "r")
print(f.read())
```

Note: the "w" method will overwrite the entire file.

Python File read() Method

Example

Read the content of the file "demofile.txt":

```
f = open("demofile.txt", "r")
print(f.read())
```

Definition and Usage

The `read()` method returns the specified number of bytes from the file. Default is -1 which means the whole file.

Syntax

```
file.read()
```

Parameter Values

Parameter	Description
size	Optional. The number of bytes to return. Default -1, which means the whole file.

Example

Read the content of the file "demofile.txt":

```
f = open("demofile.txt", "r")
print(f.read(33))
```

Python File readline() Method

Example

Read the first line of the file "demofile.txt":

```
f = open("demofile.txt", "r")
print(f.readline())
```

Definition and Usage

The `readline()` method returns one line from the file.

You can also specified how many bytes from the line to return, by using the size parameter.

Syntax

file.readline(*size*)

Parameter Values

Parameter	Description
size	Optional. The number of bytes from the line to return. Default -1, which means the whole line.

Example

Call `readline()` twice to return both the first and the second line:

```
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

Return only the five first bytes from the first line:

```
f = open("demofile.txt", "r")
print(f.readline(5))
```

Python File seek() Method

Python file method seek() sets the file's current position at the offset. The whence argument is optional and defaults to 0, which means absolute file positioning, other values are 1 which means seek relative to the current position and 2 means seek relative to the file's end.

There is no return value. Note that if the file is opened for appending using either 'a' or 'a+', any seek() operations will be undone at the next write.

If the file is only opened for writing in append mode using 'a', this method is essentially a no-op, but it remains useful for files opened in append mode with reading enabled (mode 'a+').

If the file is opened in text mode using 't', only offsets returned by tell() are legal. Use of other offsets causes undefined behavior. Note that not all file objects are seekable.

Syntax

Following is the syntax for **seek()** method,

```
fileObject.seek(offset[, whence])
```

Parameters

- **offset** − This is the position of the read/write pointer within the file.
- **whence** − This is optional and defaults to 0 which means absolute file positioning, other values are 1 which means seek relative to the current position and 2 means seek relative to the file's end.

Return Value

This method does not return any value.

Example

The following example shows the usage of seek() method.

```
# Open a file

fo = open("foo.txt", "rw+")

print "Name of the file: ", fo.name


# Assuming file has following 5 lines

# This is 1st line

# This is 2nd line

# This is 3rd line

# This is 4th line

# This is 5th line
```

```
line = fo.readline()
print "Read Line: %s" % (line)

# Again set the pointer to the beginning
fo.seek(0, 0)
line = fo.readline()
print "Read Line: %s" % (line)

# Close opend file
fo.close()
```

Suppose, the input file '*foo.txt*' contains the following lines,

```
Python is a great language
Python is a great language
```

When we run above program, it produces following result,

```
Name of the file:  foo.txt
Read Line: Python is a great language.
Read Line: Python is a great language.
```

Various File Methods

Python has a set of methods available for the file object.

Method	Description
close()	Closes the file

detach()	Returns the separated raw stream from the buffer
fileno()	Returns a number that represents the stream, from the operating system's perspective
flush()	Flushes the internal buffer
isatty()	Returns whether the file stream is interactive or not
read()	Returns the file content
readable()	Returns whether the file stream can be read or not
readline()	Returns one line from the file
readlines()	Returns a list of lines from the file
seek()	Change the file position
seekable()	Returns whether the file allows us to change the file position
tell()	Returns the current file position
truncate()	Resizes the file to a specified size

writable()	Returns whether the file can be written to or not
write()	Writes the specified string to the file
writelines()	Writes a list of strings to the file