File seek() Method: The method seek() sets the file’s current position at the offset. The whence arguments 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 mean 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 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 behaviour.

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 field’s end.

Return Value:

This method does not return any value.

Example: fo= open("fooi.txt","rw+")

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

line=fo.readlines()

print("Read Line: %s" % (line))

fo.seek(0,0)

line = fo.readline()

print("Read Line: %s" % (line))

fo.close()

File tell() Method :

The method tell() returns the current position of the file read/write pointer within the file.

Syntax Following is the syntax for tell() method-

fileObject.tell()

Return Value : This method returns the current position of the file read/write pointer within the file.

Example: fo= open("fooi.txt","r+")

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

line=fo.readline()

print("Read Line: %s" % (line))

pos=fo.tell()

print("current position: ", pos)

fo.close()

File truncate() Method: The method truncate() truncates the file’s size. If the optional size argument is present, the file is truncated to (at most) that size.

The size defaults to the current position. The current file position is not changed. Note that if a specified size exceeds the file current size, the result is platform-department.

Note: This method will not work in case the file is opened in read-only mode.

Syntax : FileObject.truncate([size])

Parameters :

Size – If this is optional argument present, the file is truncated to (at most) that size.

Return Value: This method does not return value.

Example: The following example shows the usage of truncate() method.

File write() Method :

The method write() writes a string str to the file .There is no return value. Due to buffering, the string may not actually show up in the file until the flush() or close() method is called

Syntax: Following is the syntax for write() method.

fileObject.write(str)

Parameters:

Str - This is the String to be written in the file.

Return Value

This method does not return value.

File writelines() Method:

The method writelines() writes a sequence of strings to the file. The sequence can be any iterable object producing strings, typically a list of strings. There is no return value.

Syntax:

Following is the syntax for writelines() method.

fileObject.writelines(sequence)

Parameters:

Sequence- This is the sequence of the strings

Return Value – This method does not return any value.

Example:

fo=open("fooi.txt","r+")

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

seq = ["This is 4th line\n", "This is the 5th line"]

fo.seek(0,2)

line = fo.writelines(seq)

fo.seek(0,0)

for index in range(5):

line=next(fo)

print("Line No %d - %s" % (index,line))

fo.close()

os.access() Method: os.access(path,mode)

Parameters:

* path - This is the path which would be tested for existence or any access.
* mode - This should be F\_OK to test the existence of path, or it can be the inclusive OR of one or more of R\_OK, W\_OK, and X\_OK to test permissions.

o os.F\_OK: Value to pass as the mode parameter of access() to test the existence of path.

o os.R\_OK: Value to include in the mode parameter of access() to test the readability of path.

o os.W\_OK: Value to include in the mode parameter of access() to test the writability of path.

o os.X\_OK: Value to include in the mode parameter of access() to determine if path can be executed

Return Value: This method returns True if access is allowed, False if not.

import os, sys

ret = os.access("/tmp/foo.txt", os.F\_OK)

print ("F\_OK - return value %s"% ret)

ret = os.access("/tmp/foo.txt", os.R\_OK)

print ("R\_OK - return value %s"% ret)

ret = os.access("/tmp/foo.txt", os.W\_OK)

print ("W\_OK - return value %s"% ret)

ret = os.access("/tmp/foo.txt", os.X\_OK)

print ("X\_OK - return value %s"% ret)

os.getcwd() Method :The method getcwd() returns current working directory of a process.

Syntax: Following is the syntax for getcwd() method-

os.ggetcwd(path)

Return Value This method returns the current working directory of a process.

Example The following example shows the usage of getcwd() method-

import os, sys

os.chdir("/var/www/html" )

print ("Current working dir : %s" % os.getcwd())

fd = os.open( "/tmp", os.O\_RDONLY )

os.fchdir(fd)

print ("Current working dir : %s" % os.getcwd())

os.close( fd )