

# PYTHON File I / O



Session will start in few minutes

Dr Venkat Raju

# Python File I/O

---

## Persistence:

- Most of the programs are transient in the sense that they run for a short time and produce some output, but when they end, their data disappears. If you run the program again, it starts with a clean slate.
  - Other programs are **persistent**: they run for a long time (or all the time); they keep at least some of their data in permanent storage (a hard drive, for example); and if they shut down and restart, they pick up where they left off.
  - One of the simplest ways for programs to maintain their data is by **reading and writing text files**.
-

# Types of files

---

- Python supports two types of files – **text files and binary files**.
  - These two file types may look the same on the surface, but they **encode data differently**.
  - While both **binary and text files contain data stored as a series of bits** (binary values of 1s and 0s), **the bits in text files represent characters**, while the **bits in binary files represent custom data**.
  - Binary file formats may include multiple types of data in the same file, such as image, video, and audio data.
-

# Creating, opening and closing text files

- All files must be opened first before they can be read from or written to using the Python's built-in `open()`
- When a file is opened using `open()` function, it returns a file object called a file handler that provides methods for accessing the file.

**mode is parameter**

**Returns  
file\_handler user  
defined**

```
file_handler = open(filename, mode)
```

**file\_name is user  
defined**

- The mode can be "r" reading , "w" writing and "a" appending purpose.
- The mode can be "r+" reading/writing , "w+" reading/writing and "a+" reading/appending purpose.
- It is important to close the file once the processing is completed.

**`file_handler.close()`**

# Read and Write Methods

- When you use the `open()` function a file object is created.

Method	Syntax	Description
<code>read()</code>	<code>file_handler.read([size])</code>	This method is used to read the contents of a file up to a size and return it as a string.
<code>readline()</code>	<code>file_handler.readline()</code>	This method is used to read a single line in file.
<code>write()</code>	<code>file_handler.write(string)</code>	This method will write the contents of the string to the file, returning the number of characters written.
<code>tell()</code>	<code>file_handler.tell()</code>	This method returns an integer giving the file handler's current position within the file, measured in bytes from the beginning of the file.
<code>seek()</code>	<code>file_handler.seek(offset, from_what)</code>	This method is used to change the file handler's position. The position is computed from adding <code>offset</code> to a reference point. The reference point is selected by the <code>from_what</code> argument. A <code>from_what</code> value of 0 measures from the beginning of the file, 1 uses the current file position, and 2 uses the end of the file as the reference point.

Seek Operation	Meaning
f.seek(0)	<b>Move file pointer to the beginning of a File</b>
f.seek(5)	<b>Move file pointer five characters ahead from the beginning of a file.</b>
f.seek(0, 2)	<b>Move file pointer to the end of a File</b>
f.seek(5, 1)	<b>Move file pointer five characters ahead from the current position.</b>
f.seek(-5, 1)	<b>Move file pointer five characters behind from the current position.</b>
f.seek(-5, 2)	<b>Move file pointer in the reverse direction. Move it to the 5th character from the end of the file</b>

# Program for reading and writing data

```
print('----writing to file----')
obj=open("file.txt","w")
obj.write("Hello DS
Students")
obj.close()
```

```
print('----read from file----')
obj1=open("file.txt","r")
s=obj1.read()
print (s)
obj1.close()
```

```
print('----read from file 5 chars-
---')
obj2=open("file.txt","r")
s1=obj2.read(5)
print (s1)
obj2.close()
```

# File attributes

- When the Python open() function is called, it returns a file object called a file handler.

Attribute	Description
file_handler.closed	It returns a Boolean True if the file is closed or False otherwise.
file_handler.mode	It returns the access mode with which the file was opened.
file_handler.name	It returns the name of the file.

```
print('----file attributes----')
obj = open("file1.txt", "w")
print (obj.name)
print (obj.mode)
print (obj.closed)
```

```
1 print('----file attributes----')
2 obj = open("file1.txt", "w")
3 print (obj.name)
4 print (obj.mode)
5 print (obj.closed)
6
```

```
----file attributes----
file1.txt
w
False
```

# Example with seek(), tell()

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

```
position = fo.tell();
print ("Current file position : ", position)
```

```
position = fo.seek(0, 0);
str = fo.read(10);

print ("Again read String is : ", str)
fo.close()
```

```
1  fo = open("file.txt", "r+")
2  str = fo.read(10);
3  print ("Read String is : ", str)
4
5  position = fo.tell();
6  print ("Current file position : ", position)
7
8  position = fo.seek(0, 0);
9  str = fo.read(10);
10 print ("Again read String is : ", str)
11 fo.close()
12
```

```
Read String is : Hello Cond
Current file position : 10
Again read String is : Hello Cond
```

# Handling exceptions with open() and close()

- If an exception occurs while performing some operation on the file, then the code exits without closing the file.
- In order to overcome this problem, you should use a try-except-finally block to handle exceptions.

```
try:  
    f = open("file.txt", "w")  
    f.write('Welcome to Conduira  
Online!')  
    print("file writing completed")  
    f.close()  
except IOError:  
    print('ERROR in opening a file')
```

```
try:  
    f = open("file.txt", "r")  
    str = f.read()  
    print(str)  
    f.close()  
except IOError:  
    print('ERROR in opening a  
file')
```

```
1  try:  
2      f = open("file.txt", "w")  
3      f.write('Welcome to Conduira Online!')  
4      print("file writing completed")  
5      f.close()  
6  except IOError:  
7      print('ERROR in opening a file')
```

file writing completed

```
1  try:  
2      f = open("file.txt", "r")  
3      str = f.read()  
4      print(str)  
5      f.close()  
6  except IOError:  
7      print('ERROR in opening a file')
```

Welcome to Conduira Online!

# Using with statement

- The **with** statement automatically closes the file after executing its block of statements.
- In the syntax, the words **with** and **as** are keywords and the **with** keyword is followed by the **open()** function and ends with a **colon**

```
with open (file, mode) as file_handler:  
    Statement_1  
    Statement_2  
    ...  
    Statement_N
```

- The **as** keyword acts like an alias and is used to assign the returning object from the **open()** function to a new variable *file\_handler*.
- The **with** statement creates a context manager and it will automatically close the file handler object

# Program for reading and writing data

```
with open('file1.txt', 'w') as f:  
  
    data = 'python is a prg lang used for data  
    analytics'
```

```
with open('file1.txt', 'r') as f:  
  
    data = f.read()  
  
    print(data)
```

```
1 with open('file1.txt', 'w') as f:  
2     data = 'python is a prg lang used for data analytics'  
3     f.write(data)
```

```
1 with open('file1.txt', 'r') as f:  
2     data = f.read()  
3     print(data)
```

python is a prg lang used for data analytics

