Python File Operations

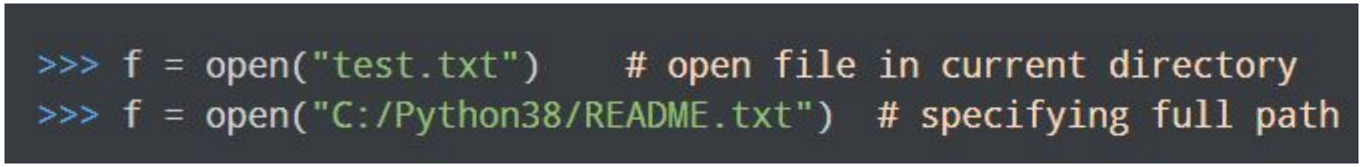
Files

- Files are named locations on disk to store related information.
- They are used to permanently store data in a **non-volatile** memory (e.g. **hard disk**).
- Since Random Access Memory (RAM) is volatile (which loses its data when the computer is turned off), we use files for future use of the data by permanently storing them.
- When we want to read from or write to a file, we need to open it first. When we are done, it needs to be closed so that the resources that are tied with the file are

Basic File Operation

- opening a file
- reading a file
- writing a file
- closing a file
- Various file methods

Opening Files in Python



```
>>> f = open("test.txt")    # open file in current directory
>>> f = open("C:/Python38/README.txt")  # specifying full path
```

>>>**Import os**
>>>**os.getcwd()**

Mode	Description
r	Opens a file for reading. (default)
w	Opens a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
x	Opens a file for exclusive creation. If the file already exists, the operation fails.
a	Opens a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.
t	Opens in text mode. (default)
b	Opens in binary mode.
+	Opens a file for updating (reading and writing)

Closing Files in Python

```python
f = open("test.txt")      # equivalent to 'r' or 'rt'
f = open("test.txt",'w')  # write in text mode
f = open("img.bmp",'r+b') # read and write in binary mode
```

```python
f = open("test.txt", encoding = 'utf-8')
# perform file operations
f.close()
```

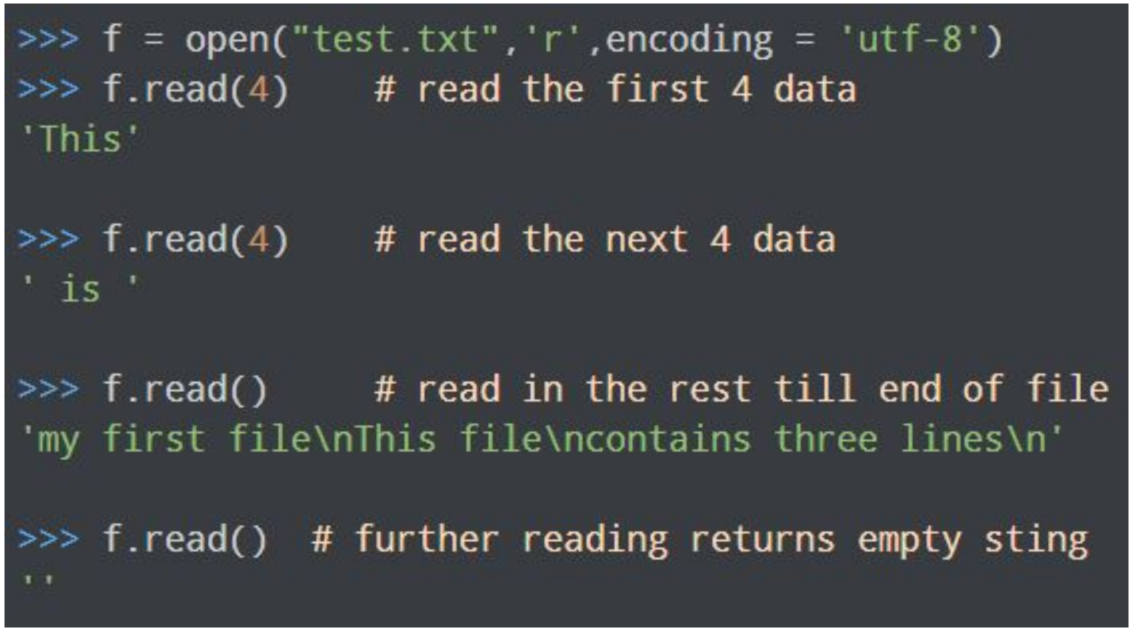
This method is not entirely safe.

```python
try:
   f = open("test.txt", encoding = 'utf-8')
   # perform file operations
finally:
   f.close()
```

```python
with open("test.txt", encoding = 'utf-8') as f:
   # perform file operations
```

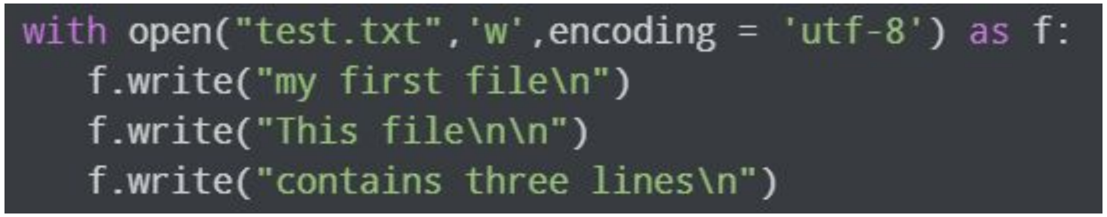
This method is best and entirely safe.

Reading and Writing to Files in Python



```
>>> f = open("test.txt",'r',encoding = 'utf-8')
>>> f.read(4)    # read the first 4 data
'This'

>>> f.read(4)    # read the next 4 data
' is '

>>> f.read()     # read in the rest till end of file
'my first file\nThis file\ncontains three lines\n'

>>> f.read()  # further reading returns empty sting
''
```

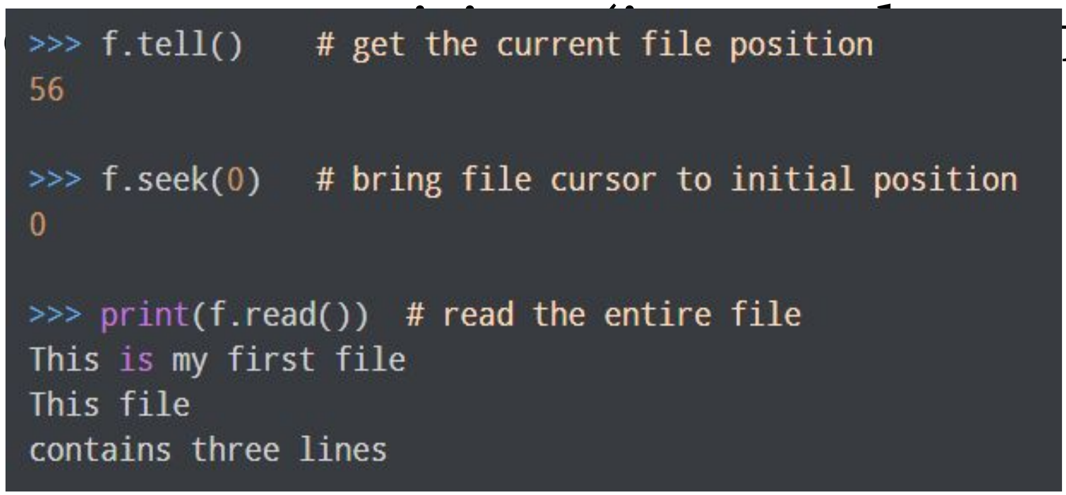
This is my first file
This file
Contains three lines



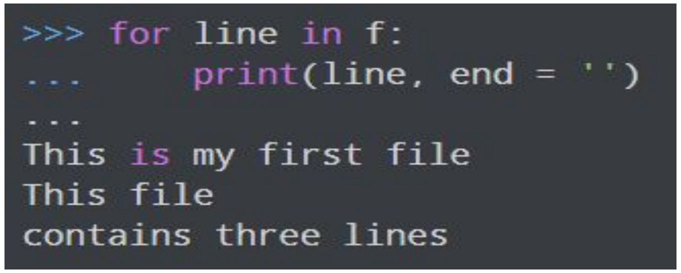
```
with open("test.txt",'w',encoding = 'utf-8') as f:
   f.write("my first file\n")
   f.write("This file\n\n")
   f.write("contains three lines\n")
```

Various file methods

- We can change our current file cursor (position) using the seek() method. Similarly, the tell() method returns our f bytes).

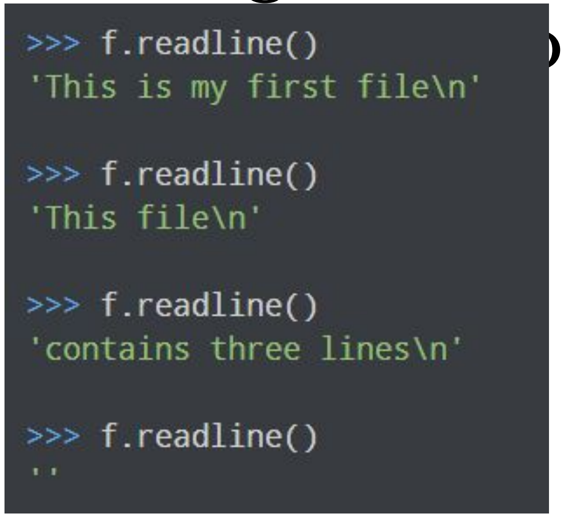


```
>>> f.tell()    # get the current file position
56

>>> f.seek(0)   # bring file cursor to initial position
0

>>> print(f.read())  # read the entire file
This is my first file
This file
contains three lines
```

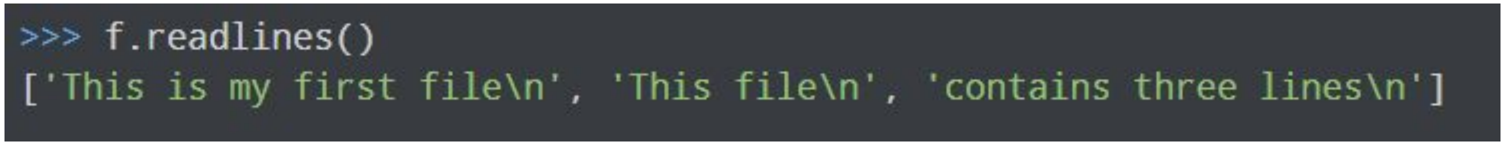


```
>>> for line in f:
...     print(line, end = '')
...
This is my first file
This file
contains three lines
```

- Alternatively, we can use the **readline()** method to read individual lines of a file. This method reads a file till the newline.
- **readlines()** method returns a list of remaining lines of the entire file. All these reading methods return empty values of file (EOF) is reached.



```
>>> f.readline()
'This is my first file\n'

>>> f.readline()
'This file\n'

>>> f.readline()
'contains three lines\n'

>>> f.readline()
''
```



```
>>> f.readlines()
['This is my first file\n', 'This file\n', 'contains three lines\n']
```

Queries???