Opening a file

```
# MyData.txt
# This is a Test file.
# It is stored in hard disk.
# Files will have their names.
# They are accessed using their names.

fp = open("MyData.txt") #default is read mode, returns a file object

print(type(fp)) # this object belongs to TextIOWrapper class
```

```
<class '_io.TextIOWrapper'>
```

```
print(fp) # iterable object
```

```
<_io.TextIOWrapper name='MyData.txt' mode='r' encoding='cp1252'>
```

```
for i in fp:
    print(i)
```

```
Hello Students
```

```
fp.close() # Closing the file which was openned
```

open a file not stored in the current working directory

```
fp = open("C:\Users\mania\OneDrive\Desktop\abc.txt") #absolute path - 
path from the root directory
print(fp.read())
fp.close()
```

```
  Cell In[15], line 1
    fp = open("C:\Users\mania\OneDrive\Desktop\abc.txt") #absolute
path - path from the root directory
               ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes
in position 2-3: truncated \UXXXXXXXX escape
```

```
fp = open("C:\\Users\\mania\\OneDrive\\Desktop\\abc.txt") #absolute
path - path from the root directory
print(fp.read())
fp.close()
```

```
This file is stored on desktop.
```

Reading a file

read(): read the whole file, returns a string

```
fp = open("MyData.txt","r")
str1 = fp.read()
print(str1)
print(type(str1))
fp.close()
```

```
Hello Students
<class 'str'>
```

```
fp = open("MyData.txt","r")
str1 = fp.read(6)  #read(n) : n number of characters need to read
print(str1)
str1 = fp.read(10)
print(str1)
fp.close()
```

```
Hello
Students
```

readline() : reads a line (read characters till \n is encountered)

```
fp = open("MyData.txt","r")
line1 = fp.readline()
print(line1, end="")
# print(type(line1))
line2 = fp.readline()
print(line2,end="")
line3 = fp.readline()
print(line3,end="")
print(fp.read())
fp.close()
```

```
Hello Students
```

readlines() : reads the whole file and returns the list of all the lines.

```
fp = open("MyData.txt","r")
li = fp.readlines()
print(li)
fp.close()
```

```
['Hello Students']
```

```
fp = open("MyData.txt","r")
li = fp.readlines()
for i in li:
```

```
    print(i, end="")
fp.close()
```

```
Hello Students
```

Properties of file object

```
fp.name
```

```
'MyData.txt'
```

```
fp.closed
```

```
True
```

```
fp.mode
```

```
'r'
```

Error handling using with

```
fp = open("MyData.txt","r")
print(fp.read())
print(10/0)
fp.close()
```

```
Hello Students

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[41], line 3
      1 fp = open("MyData.txt","r")
      2 print(fp.read())
----> 3 print(10/0)
      4 fp.close()

ZeroDivisionError: division by zero
```

```
fp.closed  #as before closing the file, the ZeroDivisionError has occured
```

```
False
```

```
with open("MyData.txt","r") as fp:
    print(fp.read())
    print(10/0)
    fp.close()
```

```
Hello Students
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[45], line 3
      1 with open("MyData.txt","r") as fp:
      2     print(fp.read())
----> 3     print(10/0)
      4     fp.close()

ZeroDivisionError: division by zero
```

```
fp.closed #The same error has occured here, but the file has been closed automaticaly.
```

```
True
```

writing to a file

write() : write a string to the file

```
fp = open("file1.txt","w") # create and opens a file1.txt for writing
s1 = "Hello Students"
fp.write(s1)
fp.close()

fp = open("MyData.txt","w") # opens a MyData.txt for writing
s1 = "Hello Students"
fp.write(s1)                # overwrites with s1 string
fp.close()
```

writelines(list/tuple/set)

```
li = ['This is a Test file.\n', 'It is stored in hard disk.\n', 'Files will have their names.\n', 'They are accessed using their names.']
fp = open("file2.txt","w")
fp.writelines(li)
fp.close()
```

File opening modes (r, w, a, r+, w+, a+, x) : Text Files

Read ("r"):

Opens a file for reading. The file must exist, otherwise, an error is raised.

```
with open('file1.txt', 'r') as file:
    content = file.read()
```

Write ("w"):

Opens a file for writing.

If the file doesn't exist, it is created. If it does exist, its content is truncated and overwritten by the new content.

```
with open('file1.txt', 'w') as file:
    file.write('Hello, world!')
```

Append ("a"):

Opens a file for appending.

If the file doesn't exist, it is created. If it exists, new data is written at the end of the file.

```
with open('file1.txt', 'a') as file:
    file.write('\nAppend this line.')
```

Read and Write ("r+"):

Opens a file for both reading and writing.

The file must exist.

```
with open('file1.txt', 'r+') as file:
    content = file.read()
    file.write('Adding more content.')
```

Write and Read ("w+"):

Opens a file for both writing and reading.

If the file doesn't exist, it is created. If it exists, its content is truncated.

```
with open('file1.txt', 'w+') as file:
    file.write('Start fresh.')
    file.seek(0)
    content = file.read()
```

Append and Read ("a+"):

Opens a file for both appending and reading.

If the file doesn't exist, it is created.

```
with open('file1.txt', 'a+') as file:
    file.write('\nAdd this at the end.')
```

```
    file.seek(0)
    content = file.read()
```

Exclusive ("x"):

creates a new file and opens it for writing.

If the file already exists, the open() function will raise a FileExistsError.

```
with open('file1.txt', 'x') as file:
    file.write('This is a new file created exclusively.')
```

```
---------------------------------------------------------------------------
-----
FileExistsError                           Traceback (most recent call
last)
Cell In[78], line 1
----> 1 with open('file1.txt', 'x') as file:
      2     file.write('This is a new file created exclusively.')

File ~\anaconda3\Lib\site-packages\IPython\core\
interactiveshell.py:324, in _modified_open(file, *args, **kwargs)
    317 if file in {0, 1, 2}:
    318     raise ValueError(
    319         f"IPython won't let you open fd={file} by default "
    320         "as it is likely to crash IPython. If you know what
you are doing, "
    321         "you can use builtins' open."
    322     )
--> 324 return io_open(file, *args, **kwargs)

FileExistsError: [Errno 17] File exists: 'file1.txt'
```

```
with open('file3.txt', 'x') as file:
    file.write('This is a new file created exclusively.')
```

File openning modes (rb, wb, ab, rb+, wb+, ab+) : Binary Files

```
fp = open("python-logo.jpg","rb")
data = fp.read()
fp1 = open("python-logo-copy.jpg","wb")
fp1.write(data)
fp.close()
fp1.close()
```

Random Access of file

tell()

tell() is used to obtain the current position of the file pointer within the file.

```
fp = open("file1.txt","r")
print("Before reading pointer is at:", fp.tell()) #file pointer always starts from 0.
fp.read(2)
print("After reading 2 characters pointer is at: ",fp.tell())
fp.read(3)
print("After reading 5 characters pointer is at: ",fp.tell())
fp.read()
print("after going to the end of the file, pointer is at:", fp.tell())
fp.close()
```

```
Before reading pointer is at: 0
After reading 2 characters pointer is at:  2
After reading 5 characters pointer is at:  5
after going to the end of the file, pointer is at: 34
```

seek(offset, startpoint_for_offset)

offset : how many positions the pointer will move

startpoint_for_offset : specifies the point from where pointer will move specified as offset

0 -> from the beginning of the file (default)

1 -> from the current position of file pointer

2 -> from the end of the file

```
fp = open("file2.txt","r")
print("Before reading: ",fp.tell())
fp.read(6)
print("After reading 6 characters, pointer is at:", fp.tell())
fp.seek(0,0) # 0 characters from beginning of the file i.e. at 0
print("After seeking pointer to beginning of the file:", fp.tell())
fp.close()
```

```
Before reading:  0
After reading 6 characters, pointer is at: 6
After seeking pointer to beginning of the file: 0
```

```
fp = open("file2.txt","r")
print("Before reading: ",fp.tell())
fp.seek(0,2) # 0 characters from the end of the file i.e. at last
print("After seeking pointer to the of the file:", fp.tell())
fp.close()
```

```
Before reading:  0
After seeking pointer to the of the file: 116
```

```
fp = open("file2.txt","r")
print("Before reading: ",fp.tell())
fp.read(1)
fp.seek(3,1) # 3 characters from the current position of the file
pointer
print("After seeking 3 characters, the file pointer is at:",
fp.tell())
fp.close()
```

```
Before reading:  0
```

```
---------------------------------------------------------------------
-----
UnsupportedOperation                      Traceback (most recent call
last)
Cell In[91], line 4
      2 print("Before reading: ",fp.tell())
      3 fp.read(1)
----> 4 fp.seek(3,1) # 3 characters from the current position of the
file pointer
      5 print("After seeking 3 characters, the file pointer is at:",
fp.tell())
      6 fp.close()

UnsupportedOperation: can't do nonzero cur-relative seeks
```

```
fp = open("file2.txt","r")
print("Before reading: ",fp.tell())
fp.read(1)
fp.seek(0,1) # 0 characters from the current position of the file
pointer
print("After seeking 0 characters, the file pointer is at:",
fp.tell())
fp.close()
```

```
Before reading:  0
After seeking 0 characters, the file pointer is at: 1
```

Note :

To do non-zero end relative and current relative seeks, file should be opened in binary mode.

```
fp = open("file2.txt","rb")
print("Before reading: ",fp.tell())
fp.read(1)
fp.seek(3,1) # 3 characters from the current position of the file
pointer
print("After seeking 3 characters, the file pointer is at:",
fp.tell())
fp.close()
```

```
Before reading:  0
After seeking 3 characters, the file pointer is at: 4
```

```
fp = open("file2.txt","rb")
print("Before reading: ",fp.tell())
print(type(fp.read(1))) # returns bytes
fp.seek(3,1) # 3 characters from the current position of the file
pointer
print("After seeking 3 characters, the file pointer is at:",
fp.tell())
fp.close()
```

```
Before reading:  0
<class 'bytes'>
After seeking 3 characters, the file pointer is at: 4
```

```
fp = open("file2.txt","rb")
print("Before reading: ",fp.tell())
print(fp.read(4).decode()) # returns bytes
fp.seek(3,1) # 3 characters from the current position of the file
pointer
print("After seeking 3 characters, the file pointer is at:",
fp.tell())
print(fp.read().decode())
fp.close()
```

```
Before reading:  0
This
After seeking 3 characters, the file pointer is at: 7
 a Test file.
It is stored in hard disk.
Files will have their names.
They are accessed using their names.
```

```
fp = open("file2.txt","rb")
print(type(fp))
fp.close()
```

```
<class '_io.BufferedReader'>
```