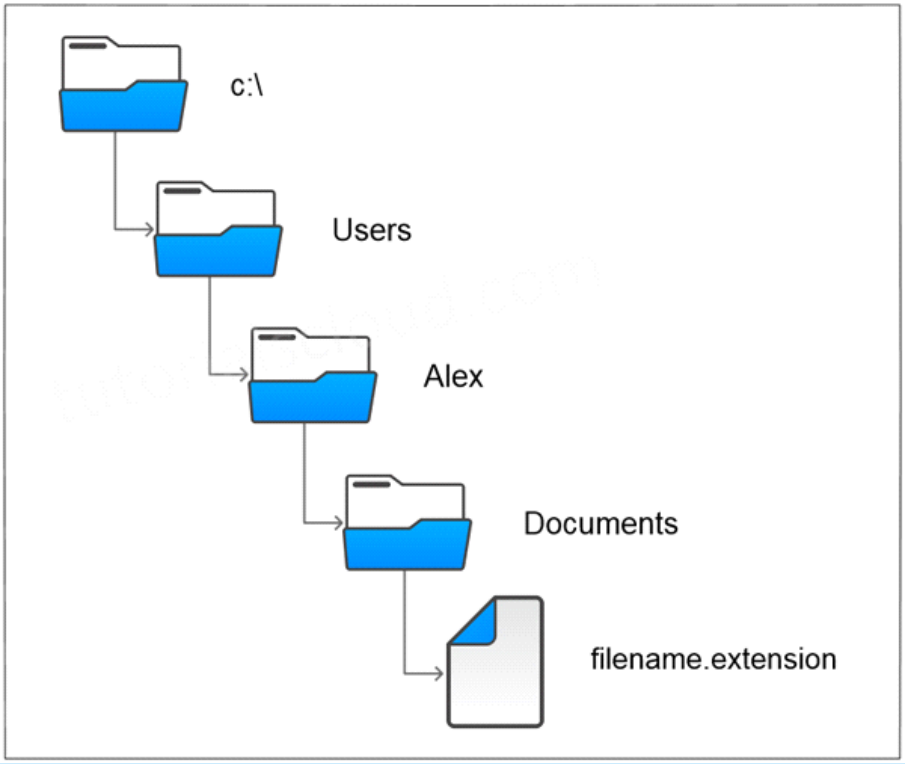
FILE HANDLING

05 November 2020 21:28

All programs need the input to process and output to display data. And everything needs a file as name storage compartments on computers that are managed by OS. Though variables provide us a way to store data while the program runs, if we want out data to persist even after the termination of the program, we have to save it to a file.

FILE AND ITS PATH

There are always two parts of a file in the computer system, the filename and its extension. Also, the files have two key properties - its name and the location or path, which specifies the location where the file exists. The filename has two parts, and they are separated by a dot (.) or period.
Figure - File and its path:



A built-in open method is used to create a Python file-object, which provides a connection to the file that is residing on the programmer's machine. After calling the function open, programmers can transfer strings of data to and from the external file that is residing in the machine.

FILE OPENING IN PYTHON

open() function is used to open a file in Python. It's mainly required two arguments, first the file name and then file opening mode.

Syntax:

file_object = open(filename [,mode] [,buffering])

In the above syntax, the parameters used are:

- filename: It is the name of the file.
- mode: It tells the program in which mode the file has to be open.
- buffering: Here, if the value is set to zero (0), no buffering will occur while accessing a file; if the value is set to top one (1), line buffering will be performed while accessing a file.

MODES OF OPENING FILE IN PYTHON

The file can be opened in the following modes:

Mode	Description
r	Opens a file for reading only. (It's a default mode.)
w	Opens a file for writing. (If a file doesn't exist already, then it creates a new file. Otherwise, it's truncate a file.)
x	Opens a file for exclusive creation. (Operation fails if a file does not exist in the location.)
a	Opens a file for appending at the end of the file without truncating it. (Creates a new file if it does not exist in the location.)
t	Opens a file in text mode. (It's a default mode.)
b	Opens a file in binary mode.
+	Opens a file for updating (reading and writing.)

PYTHON FILE OPENING MODES

FILE OBJECT ATTRIBUTES

If an attempt to open a file fails, then open returns a false value. Otherwise, it returns a file object that provides various information related to that file.

Example:

```
# file opening example in Python
# METHOD FIRST FOR OPENING  A FILE if  u want to read or wright then first is best
fo = open("sample.txt", "wb")
    print ("File Name: ", fo.name)
    print ("Mode of Opening: ", fo.mode)
    print ("Is Closed: ", fo.closed)
    print ("Softspace flag : ", fo.softspace) # will cover next video
```

Output:

```
File Name: sample.txt
Mode of Opening: wb
Is Closed: False
Softspace flag: 0
```

FILE READING IN PYTHON

For reading and writing text data, different text-encoding schemes are used, such as ASCII (American Standard Code for Information Interchange), UTF-8 (Unicode

Transformation Format), UTF-16.
Once a file is opened using an open() method, then it can be read by a method called read().

Example:

```
# read the entire file as one string
# METHOD FIRST FOR OPENING  A FILE
with open('filename.txt') as f:
data = f.read()

# Iterate over the lines of the File
with open('filename.txt') as f:
for line in f :
   print(line, end=' ')

# process the lines
```

FILE WRITING IN PYTHON

Similarly, for writing data to files, we have to use open() with 'wt' mode, clearing and overwriting the previous content. Also, we have to use the write() function to write into a file.

Example:

```
# Write text data to a file
with open('filename.txt' , 'wt') as f:
    f.write ('hi there, this is a first line of file.\n')
    f.write ('and another line.\n')
```

Output:

```
hi there, this is a first line of file.
and another line.
```

By default, in Python - using the system default text, encoding files are read/written. Though Python can understand several hundred text-encodings but the most common encoding techniques used are ASCII, Latin-1, UTF-8, UTF-16, etc. The use of 'with' statement in the example establishes a context in which the file will be used. As the control leaves the 'with' block, the file gets closed automatically.

WRITING A FILE THAT DOES NOT EXIST

The problem can be easily solved by using another mode - technique, i.e., the 'x' mode to open a file instead of 'w' mode.
Let's see two examples to differentiate between them.

Example:

```
with open('filename' , 'wt') as f:
    f.write ('Hello, This is sample content.\n')
# This will create an error that the file 'filename' doesn't exist.

with open ('filename.txt' , 'xt') as f:
    f.write ('Hello, This is sample content.\n')
```

In binary mode, we should use 'xb' instead of 'xt'.

CLOSING A FILE IN PYTHON

In Python, it is not system critical to close all your files after using them, because the file will auto close after Python code finishes execution. You can close a file by using the close() method.

Syntax:

```
file_object.close();
```

Example:

```
fo.close()
```

I HOPE YOU WILL ENJOY THIS VIDEO,

IF YOU SHARE THIS VIDEO WITH YOU FRIENDS

THIS ENCOURAGE ME A LOT TO UPLOAD

A VIDEO VERY EFFICIENTLY & QUICKLY.

PLEASE SUBSCRIBE MY CHANNEL METAEDUCATORS

SHARE + LIKE +COMMENT