Date
July 19, 2021

WEEK 9

File Handling

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 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 freed.

Hence, in Python, a file operation takes place in the following order:

1. Open a file
2. Read or write (perform operation)
3. Close the file

OPENING FILES IN PYTHON

Python has a built in open() function to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

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

We can specify the mode while opening a file. In mode, we specify whether we want to read [r], write [w], or append [a] to a file. We can also specify, if we want to open a file in text mode or binary mode.

The default is reading in text mode. In this mode, we get strings when reading from the file.
On the other hand, binary mode returns bytes and this is mode to be used when dealing with non-text files like images or executable files.

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.

CODE:
```
f = open("test.txt")        # equivalent to 'r', read mode
f = open("test.txt", 'w')   # to write in text mode
```

CLOSING FILES IN PYTHON

→ When we are done with performing operations on the file, we need to properly close the file.

→ Closing a file will free-up the resources that were tied with the file. It is done using the close() method available in python.

→ Python has a garbage collector to clean up unreferenced objects but we must not rely on it to close the file.

CODE:
```
f = open("test.txt", 'r')
# perform file operations
f.close()
```

→ This method is not entirely safe. If an exception occurs when we are performing some operation with the file, the code exits without closing the file.

A safer way is to use a try...finally block.

```
try:
    f = open("test.txt", 'r')
    # perform file operations
finally:
    f.close()
```

→ This way we are guaranteeing that the file is properly closed even if an exception is raised that causes program flow to stop.

WRITING TO FILES IN PYTHON

→ In order to write into a file Python, we need to open it in write [w] mode.

→ We need to be careful with the w mode, as it will overwrite into the file if it already exists. Due to this, all the previous data is erased.

→ Writing a string is done using the write() method. This method returns the number of characters written to the file.

CODE:

```
f = open("test.txt", 'w')
f.write("my file")
f.write("\n")
f.write("I am Gagneet Kaur.")
```

OUTPUT

```
myfile
I am Gagneet Kaur.
```

→ This program will create a new file named test.txt in the current directory if it does not exist. If it exists, it is overwritten.

→ We must include the newline characters ourselves to distinguish the different lines.

READING FILES IN PYTHON

→ To read a file in Python, we must open the file in reading [r] mode.

→ There are various methods available for this purpose. We can use the read(size) methode to read the size number of data. If the size parameter is not defined / specified, it reads and returns up to the end of file.

→ We can read the test.txt file we wrote previously.

CODE:

```
f = open ("test.txt", 'r')
f.read(4)          # read the first 4 data
f.read(4)          # read the next 4 data
f.read()           # read in the till the end of file
f.close()
```

OUTPUT:

```
'myfi'
'le\nI'
' am Gagneet Kaur'
```

→ We can see that the read() method returns a newline as '\n'. Once the end of the file is reached, we get an empty string on further reading.

→ We can change our current file cursor (position) using the seek() method. Similarly, the tell() method returns our current position (in no. of bytes).

Date
July 21, 2021

classmate
Date
Page

File Handling

BIG TEXT FILE HANDLING

→ We use the readline() method to read individual lines of a file. This method reads a file till the newline, including the newline character.

CODE:
```
f=open("test.txt",'r')
f.readline()
f.readline()
```

Output:
```
myfile
I am Gagneet Kaur
```

Let's See Some Operations We can Do/Perform on Big Files

→ Finding a number from a phone directory file that is very large.

CODE:
```
f = open("directory.txt",'r')
flag = 0
s = '0'
while (s!=''):          # this null character/empty string shows that it is the end of the file.
    s = f.readline()
if(s!=''):n = int(s)
```

```
if (n = 9024876051):
    print ('The no. was found')
    flag = 1
    break
if (flag == 0):
    print ("The no. was not found")
```

CAESAR CIPHER

CODE:

```
"This program considers an input file and encrypts it
by using caesar cipher. by that we mean we shift
the letters by 3 units, for example a becomes d,
b becomes e, and so on"

import string

def create_caesar_dictionary():
    l = string.ascii_lowercase
    l = list(l)
    d = {}
    for i in range(len(l)):
        d[l[i]] = l[(i+3)%26]
    return d

f = open("sherlock.txt", 'r')          # opens already existing file
g = open("encrypted_sherlock.txt", 'w')  # creates new file

d = create_caesar_dictionary()
# creates a dictionary
```

```
c = f.read(1)                  # reads one character from file f
while (c != '')                # end of file condition
    g.write(d[c])              # writes in file g
    c = f.read(1)              # reads next character in file f

f.close()
g.close()
```

GENETIC SEQUENCES

Humans have 4 letters for genetic sequence.
A particular sequence defines if a person has disease or not. Let's open a file and see.

CODE:

```
f = open("human.txt", 'r')

seq = f.read()

bp = 'AATCGA'

bp in seq
```

>> True (if the particular seq. is present)