Lesson – 9: File Management

Difference b/w Text File & Binary File

A text file stores data in the form of alphabets, digits and other special symbols by storing their ASCII values and are in a human-readable format (.txt, .c, .py etc.). Whereas binary file contains a sequence or a collection of bytes which are not in a human-readable format (.png, .jpeg, .mp4 etc.)

IMPORTANT NOTES - 1:

In Python 3 - a string is a sequence of characters, i.e Unicode points; these are an abstract concept, and can't be directly stored on disk. A byte string is a sequence of, unsurprisingly, bytes - things that *can* be stored on disk. The mapping between them is an *encoding* - there are quite a lot of these (and infinitely many are possible) - and you need to know which applies in the particular case in order to do the conversion, since a different encoding may map the same bytes to a different string:

```
>>> b'\xcf\x84o\xcf\x81\xce\xbdo\xcf\x82'.decode('utf-16')

'蓏콯캁澽苏'

>>> b'\xcf\x84o\xcf\x81\xce\xbdo\xcf\x82'.decode('utf-8')
'τορνος'
```

Once you know which one to use, you can use the **.decode()** method of the byte string to get the right character string from it as above. For completeness, the **.encode()** method of a character string goes the opposite way:

```
>>> 'τορνος'.encode('utf-8')
b'\xcf\x84o\xcf\x81\xce\xbdo\xcf\x82'
```

IMPORTANT NOTES - 2:

The only thing that can be stored onto a computer's memory are bytes.

To store anything onto a computer's memory, you must first encode it, i.e. convert it to bytes or binary format. For example:

- To store music, you must first encode it using MP3, WAV, etc.
- To store an image, you must first encode it using PNG, JPEG, etc.
- To store text, you must first encode it using ASCII, UTF-8, etc.

MP3, WAV, PNG, JPEG, ASCII and UTF-8 are examples of encodings.

An encoding is a format to represent audio, images, text, etc in bytes.

In Python, a byte string is just that: **a sequence of bytes**. It isn't human-readable. Under the hood, everything must be converted to a byte string before it can be stored onto a computer's memory.

On the other hand, a character string, often just called a "string", is **a sequence of characters**. It is human-readable. A character string can't be directly stored in a computer, it has to be encoded first (converted into a byte string). There are multiple encodings through which a character string can be converted into a byte string, such as ASCII and UTF-8.

```
>>> 'My Name Is Subhash'.encode('ASCII')
```

The above Python code will encode the string 'My Name Is Subhash' using the encoding ASCII. The result of the above code will be a byte string. If you print it, Python will represent it as b'My Name Is Subhash'

Remember, however, that byte strings aren't human-readable, it's just that Python decodes them from ASCII when you print them. In Python, a byte string is represented by a b, followed by the byte string's ASCII representation.

A byte string can be decoded back into a character string, if you know the encoding that was used to encode it.

```
b'My Name Is Subhash'.decode('ASCII')
```

The above code will return the original string 'I am a string'.

Encoding and decoding are inverse operations. Everything must be encoded before it can be written to disk, and it must be decoded before it can be read by a human

Program: 1

```
import os
fp = open("test.txt", "w+" )
print( fp.name )
print( fp.mode )
print( fp.closed )
fp.write("hello how are you\n");
fp.write("hello how are you\n");
fp.write("hello how are you\n");
print(fp.tell())
print(fp.seek(0))
print(fp.tell())
print(fp.read(2))
print(fp.readlines())
print(fp.tell())
fp.seek(0)
print(fp.read(5))
print(fp.tell( ))
fp.seek(0)
print(fp.readline())
os.rename("test.txt", "ps.txt")
os.remove("ps.txt")
os.mkdir("test")
os.chdir("test")
fp_new = open("input.txt", "w")
fp_new.write("Hello, Welcome to test directory" );
fp_new.close()
print(os.getcwd())
os.mkdir("testinside")
print(os.listdir( ))
```

Output:

```
test.txt
w+
False
54
0
0
he
['llo how are you\n', 'hello how are you\n',
'hello how are you\n']
54
hello
5
hello how are you
/Users/subhash/PYTHON_LEARNING/files/tes
t
['input.txt', 'testinside']
```

Program: 2

Output:

```
fp_to_read = open("file_one.txt", "r")
l = fp_to_read.read(1)
str = l

while l != '':
    l = fp_to_read.read(1)
    str = str + l

print(str)

fp_to_read.close()
```

Program : 3

Output:

```
fp_to_read = open("file_one.txt", "r")
l = fp_to_read.read(1)
str = l

while l != '':
    l = fp_to_read.readline()
    str = str + l

print(str)

fp_to_read.close()
```

Program: 4

Output:

```
fp_to_read = open("file_one.txt", "r")
fp_to_write = open("file_two.txt", "w")

l = fp_to_read.read(1)

while l != '':
    fp_to_write.write(l)
    l = fp_to_read.read(1)

fp_to_read.close()
fp_to_write.close()

fp_to_read = open("file_two.txt", "r")
l = fp_to_read.read(1)
str = l
while l != '':
    l = fp_to_read.read(1)
    str = str + l

print(str)
fp_to_read.close( )
```

Program: 5

Output:

```
fp_to_read = open("file_one.txt", "r")
fp_to_write = open("file_two.txt", "w")

l = fp_to_read.read(1)

while l != '':
    fp_to_write.write(l)
    l = fp_to_read.readline()

fp_to_read.close()
fp_to_write.close()

fp_to_read = open("file_two.txt", "r")
l = fp_to_read.readline()
str = l
while l != '':
    l = fp_to_read.readline()
    str = str + l

print(str)

fp_to_read.close()
```

Program: 6

Output:

```
fp_to_read = open("file_one.txt", "r")

for line in fp_to_read:
    print(line)

fp_to_read.close()
```

Program: 7

Output:

```
fp_one = open("file_one.txt", "r")

line = ''
line = fp_one.readline()

space = 0

while line != '':
    for i in range(0,len(line)):
        if( line[i] == ' ' ):
            space = space + 1
    line = fp_one.readline()

print("Number of spaces = ", space)
```

Program: 8

Output:

```
fp_one = open("file_one.txt", "r")

line = ''
line = fp_one.readline()
spaces = 0

while line != '':
    for ch in line:
        if( ch == ' ' ):
            spaces = spaces + 1

    line = fp_one.readline()

print( "Spaces = ", spaces )
```

Assignments To Practically Try Out:

Program: 9

```
import os
fp = open("PPP.png", "r" )
print(fp.read( ))
```

Program: 10

```
import os
fp = open("PPP.png", "rb" )
print(fp.read( ))
```

Program: 11

```
import os
fp = open(“f.bin”, “wb”)
fp.write(b”Hello\nHello\nHello\n”)
fp.close()
fp = open(“f.bin”, “rb”)
print(fp.read( ))
```

Program: 12

```
import os
fp = open(“f.bin”, “wb”)
fp.write(b”Hello\nHello\nHello\n”)
fp.close()
fp = open(“f.bin”, “rb”)
print(fp.read( ))
```

Guess The Output:

1.

```
f = open("fileone.bin", "w+b")
f.write("Hello")
f.close( )
```

2.

```
f = open("fileone.bin", "w+b")
f.write(b"Hello")
f.close()
```

3.

```
f = open("fileone.bin", "w+b")
f.write("Hello".encode( ))
f.close()
```

4.

```
f = open("fileone.bin", "w+b")
f.write("Hello".encode())
f.seek(0)
str = f.read( )
print(str)
f.close( )
```

5.

```
f = open("fileone.bin", "w+b")
f.write("Hello".encode( ))
f.seek(0)
str = f.read()
print(str.decode())
f.close()
```

6.

```
>>> '蓏콩갅濾苏'.encode(‘utf-8’)
```

7.

```
>>> 'τορνος'.encode('utf-8')
```

8.

```
>>> b'\xcf\x84o\xcf\x81\xce\xbdo\xcf\x82'.decode('utf-16')
```

9.

```
>>> b'\xcf\x84o\xcf\x81\xce\xbdo\xcf\x82'.decode('utf-8')
```

Let's explore built-in support for file operations:

```
>>> import os
>>> cwd = os.getcwd()
>>> cwd
'C:\\Users\\Subhash K U\\AppData\\Local\\Programs\\Python\\Python39'
>>> os.path.abspath('equal.py')
'C:\\Users\\Subhash K U\\AppData\\Local\\Programs\\Python\\Python39\\equal.py'
>>> os.path.exists('equal.py')
True
>>> os.path.isdir('C:\\Users\\Subhash K U')
True
>>> os.path.isdir('C:\\Users\\Subhash K
U\\AppData\\Local\\Programs\\Python\\Python39\\equal.py')
False
>>> os.listdir(cwd)
['DLLs', 'Doc', 'equal.py', 'include', 'Lib', 'libs', 'LICENSE.txt', 'NEWS.txt', 'python.exe', 'python3.dll',
'python39.dll', 'pythonw.exe', 'Scripts', 'tcl', 'Tools', 'vcruntime140.dll', 'vcruntime140_1.dll']
>>>
```

Program 13: Program to recursively walk through a directory, prints the names of all the files, and calls itself recursively on all the directories:

```
import os

def walk(dirname):
   for name in os.listdir(dirname):
      path = os.path.join(dirname, name)

      if os.path.isfile(path):
         print(path)
      else:
         walk(path)

walk('C:\\Users\\Subhash K U\\OneDrive\\Desktop\\Sample')
```

Output:

Serialization and Deserialization:(Pickling and Unpickling)

```
#First - create a class and make it a module (call - obj.py )

class object:
    def __init__(self):
        self.name = input("Enter name:")
        self.age  = input("Enter age:")

    def display(self):
        print("Name = " + self.name )
        print("Age = "  + self.age )
```

Program: 13

```
#Second - Create a program to serialize objects (Pickling)
#pickling.py

import obj, pickle

fone = open("jum.bin", "wb")

choice = "yes"

while choice == "yes":
    o = obj.object()
    pickle.dump(o,fone)
    choice = input("Enter one more object? 'yes' or 'no' ?")
```

Output:

```
Enter name:subhash
Enter age:25
Enter one more object? 'yes' or 'no' ?yes
Enter name:charan
Enter age:12
Enter one more object? 'yes' or 'no' ?yes
Enter name:archita
Enter age:18
Enter one more object? 'yes' or 'no' ?no
```

Program: 14

Output:

```
#Third - Create a program to de-serialize objects (Unpickling)
#unpickling.py

import pickle, obj

fone = open("jum.bin", "rb")

while(True):
    try:
        obj = pickle.load(fone)
        obj.display()
    except EOFError:
        print("End of file reached")
        break
```

```
Name = subhash
Age = 25
Name = charan
Age = 12
Name = archita
Age = 18
End of file reached
```