* **I/O System**
  + Expectations from an I/O System:
    - It should allow us to communicate with multiple sources and destinations.
      * ex. Sources: Keyboard, File, Network
      * ex. Destinations: Screen, File, Network
    - It should allow us to input/output varied entities.
      * ex. Numbers, Strings, Lists, Tuples, Sets, Dictionaries, etc.
    - It should allow us to communicate in multiple ways.
      * ex. Sequential access, Random access
    - It should allow us to deal with underlying file system.
      * ex. Create, Modify, Rename, Delete files and directories.
* **Types of data used for I/O**
  + **Text and Binary**
* **File Types**
  + Text files: All program files.
  + Binary Files: All image, music, video, executable files
* **FILE I/O**
  + Sequence of operations in file I/O
    - Open a file
    - Read/Write data from/to it.
    - Close the file.
* **Functions and syntax to open and close files**
  + logical-file-object = open(‘Physical file name’, ’file opening mode’)
    - ex. f = open(‘Mydata.txt’,’r’)
    - Here through logical-file-object **f**, the program will make connection with physical file ‘mydata.txt’ in the current directory.
* **File Opening Modes:**
  + ‘r’ – for reading purpose in text mode
  + ‘w’ – for writing purpose in text mode
  + ‘a’ – for appending the data in the existing file in text mode.
  + ‘r+’ – for reading and writing purposes in text mode.
  + ‘w+’ – for writing and reading purposes in text mode.
  + ‘a+’ – for appending and reading the data in text mode.
  + ‘rb’ – for reading purpose in binary mode
  + ‘wb’ – for writing purpose in binary mode
  + ‘ab’ – for appending the data in the existing file in binary mode.
  + ‘rb+’ – for reading and writing purposes in binary mode.
  + ‘wb+’ – for writing and reading purposes in binary mode.
  + ‘ab+’ – for appending and reading the data in binary mode.
  + Opening a file brings its contents to a buffer in memory.
  + The default file-opening-mode argument is ‘r’.
  + While opening a file for writing, if the file already exists, it is overwritten.
* **Closing a file**
  + Syntax: logical-file-object-name.close()
    - ex: f.close()
  + It will close the file by vacating the buffer. Before closing the file, the data will be transferred to the physical file from the buffer.
  + Once file is closed, read/write operators on it are not feasible.
* **Read/ Write operations**
  + Two functions for writing data to a file:
    - logical-file-object.write(strings)
      * ex: f.write(“Hi, I am at Pandit Deendayal Energy University.”)
    - To write objects other than strings, we need to convert them to strings before writing.
      * tpl = ( ‘Darshit’, ‘Ragi’, ‘Aashna’)
      * lst = [1972,1974, 2006]
      * d = { ‘name’: ‘Darshit’, ‘DoB’: ’18/02/1972’ }
      * f.write(str(tpl))
      * f.write(str(lst))
      * f.write(str(d))
  + Functions for reading data from a file:
    - variable-name = logical-file-object.read()
      * data = f.read( ) 🡨 Reads entire file contents and returns as string.
    - variable-name = logical-file-object.read(no. of bytes)
      * data = f.read(n) 🡨 Reads n characters, and returns as string.
    - variable-name = logical-file-object.readline()
      * data = f.readline() 🡨 Reads a line, and returns as string.
    - variable-name = logical-file-object.readlines()
      * data = f.readlines() 🡨 Reads all the lines from a file and form a list of lines.
    - If end-of-file is reached, f.read( ) returns an empty string.
* **with** Keyword
  + We need to close a file once its usage is over, as it will free up system resources.
  + If we don’t close a file, when the file object is destroyed, file will be closed for us by Python’s garbage collector program.
  + If we use with keyword while opening a file, the file gets closed as soon as its usage is over.

with open(‘a.dat’,’r’) as f:

data = f.read()

* + **with** ensures that the file is closed even if an exception occurs while processing it.
* **Various Examples of File Handling**
* **Create the file on the hard disk and store data within that file.**

# This program creates a file on the hard disk and writes the content within a file.

flname = input("Enter a File Name to write the content.")

f = open(flname,'a+')

ch = input("Enter text that you want to write in a file. At the end enter ~ character.")

while ch != '~':

f.write(ch)

ch = input()

f.write(‘\n’)

f.close()

print(flname, "has been succssefully created on the hard disk.")

* **Write a program to read the contents of file one character at a time. Print each character that is read.**

# This program reads the content of the file.

import os, sys

flname = input("Enter a File Name to read the content.")

if os.path.isfile(flname):

f = open(flname,'r')

else:

print(flname,"doesn't exists.")

sys.exit()

ch = f.read(1)

while ch != "":

print(ch,end = '')

ch = f.read(1)

f.close()

* **Write a program to copy the content of one file into another one.**

# This program copies the content of one file to another one.

import os, sys

flrname = input("Enter a File Name to read the content.")

if os.path.isfile(flrname):

fr = open(flrname,'r')

else:

print(flrname,"doesn't exists.")

sys.exit()

flwname = input("Enter a File Name to write the content.")

fw = open(flwname,'a+')

ch = fr.read(1)

while ch != "":

# fw.write(ch)

# print(ch,end = '',file = fw, flush = True)

print(ch,end = '',file = fw)

# print(ch,end = '')

ch = fr.read(1)

fr.close()

fw.close()

print(flrname, "is copied to",flwname, "successfully.")

* **Write a program to count number of lines, words and characters in a text file.**

# This program counts number of lines, words and characters in a file.

import os, sys

flrname = input("Enter a File Name to count no. of lines, words and characters:")

if os.path.isfile(flrname):

fr = open(flrname,'r')

else:

print(flrname,"doesn't exists.")

sys.exit()

# Intialize variables.

cl = cw = cc = 0

# read line one by one and count.

for line in fr:

words = line.split()

cl = cl + 1

cw = cw + len(words)

cc = cc + len(line)

print('No. of line(s)=', cl)

print('No. of word(s)=', cw)

print('No. of char(s)=', cc)

fr.close()

* **Binary-File-Handling**
  + If file is opened for writing in binary mode then a bytes-like object should be passed to write( ) as shown below:

f = open(‘a.dat’, ‘wb+’)

d = b’\xee\x86\xaa’ 🡨 Series of three bytes, \x indicates hexadecimal.

f.write(d)

* **Moving within a File**
  + When we are reading a file or writing a file, the next read or write operation is performed from the next character/byte as compared to the previous read/write operation.
  + Thus if we read the first character from a file using f.read(1), next call to f.read(1) will automatically read the second character in the file.
  + At times we may wish to move to desired position in a file before reading/writing. This can be done using f.seek( ) method.
  + General form of seek( ): **f.seek(offset, reference)**
  + Reference can take values 0, 1, 2 being beginning of file, current position in a file and end of file respectively.
  + For file opened in text mode, reference values 0 and 2 alone can be used. Also, using 2, we can only move to end of file.
    - f.seek(512,0) 🡨 Moves to position 512 from beginning of the file.
    - f.seek(0, 2) 🡨 moves to end-of-file
  + For file opened in binary mode, reference values 0, 1 and 2 can be used.
    - f.seek(0) 🡨 moves to beginning of the file.
    - f.seek(12,0) 🡨 moves to position 12 from beginning of the file.
    - f.seek(-15,2) 🡨 moves 15 positions to left from end-of-file.
    - f.seek(6,1) 🡨 moves 6 positions to right from current position.
    - f.seek(-10,1) 🡨 moves 10 positions to left from current position.
* **Serialization and Deserialization**
  + As you know that we need to convert other than string data types to string data type before writing (Refer **Read/ Write operations),** and if the data is in the form of set, dictionary or tuple, this task becomes more cumbersome. In such cases, use a module called **json**.
  + **Serialization: json** module converts Python data type into appropriate JSON types before writing data to a file. Likewise, it converts JSON types read from a file into Python data (**Deserialization**).
  + Serialization of a Python type to JSON data is done using a function **dump()**. It writes the serialized data to a file.
  + Deserialization of a JSON data to a Python type is done using a function **load()**. It reads the data from a file, converts the data and returns the converted data.

**#serialize/deserialize a list.**

import json

f = open('Sampledata','w+')

lst = [10, 20, 30, 40, 50, 60, 70, 80, 90]

json.dump(lst,f)

f.seek(0)

inlst = json.load(f)

print(inlst)

f.close()

**#serialize/deserialize a tuple.**

f = open('Sampledata','w+')

tpl = ('Darshit','18/02/1972', 'O+ve')

json.dump(tpl,f)

f.seek(0)

intpl = json.load(f)

print(tuple(intpl))

f.close()

**#serialize/deserialize a dictionary.**

f = open('Sampledata','w+')

dct = { 'Darshit': 52, 'Ragi' : 49 }

json.dump(dct,f)

f.seek(0)

indic = json.load(f)

print((indic))

f.close()

* + Instead of writing JSON data to a file, we can write it to a string, and read it back from a string as shown below:

import json

lst = [10, 20, 30, 40, 50, 60, 70, 80, 90]

tpl = ('Darshit','18/02/1972', 'O+ve')

dct = { 'Darshit': 52, 'Ragi' : 49 }

str1 = json.dumps(lst)

str2 = json.dumps(tpl)

str3 = json.dumps(dct)

l = json.dumps(lst)

t = tuple(json.loads(str2))

d = json.loads(str3)

print (l, t, d)

* **File and Directory Operations**
  + Python allows to perform many file and directory operations.
  + File operations include creation, deletion, renaming, copying, checking if an entry is a file, obtaining statistics of a file, etc.
  + Directory operations include creation, recursive creation, renaming, changing into, deleting, listing a directory, etc.
  + Path operations include obtaining the absolute and relative path, splitting path elements, joining path, etc.
  + ‘.’ represent current directory and ‘..’ represents parent of current directory.

import os, time, shutil

import shutil

print ("OS Name:", os.name)

print (os.getcwd())

print (os.listdir('.'))

print (os.listdir('..'))

if os.path.exists('mydir'):

print ('Mydir already exists.')

else:

os.mkdir('mydir')

os.chdir('mydir')

if os.path.exists('.\\dir1\\dir2\\dir3'):

os.chdir('.\\dir1\\dir2\\dir3')

else:

os.makedirs('.\\dir1\\dir2\\dir3')

f = open('myfile.txt','w')

f.write('Hi, learn Python file handling in simple manners.\n')

f.close()

stats = os.stat('myfile.txt')

print('Size = ',stats.st\_size)

os.rename('myfile.txt','yourfile.txt')

shutil.copyfile('yourfile.txt','ourfile.txt')

os.remove('yourfile.txt')

curpath = os.path.abspath('.')

os.path.join(curpath,'yourfile.txt')

if os.path.isfile(curpath):

print('Yourfile file does exist.')

else:

print('Yourfile file doesn\'t exist.')

file = 'ourfile.txt'

print(file)

created = os.path.getctime(file)

modified = os.path.getmtime(file)

accessed = os.path.getatime(file)

print ('Date created:' + time.ctime(created))

print ('Date modified:' + time.ctime(modified))

print ('Date accessed:' + time.ctime(accessed))

* **Write following programs considering files in mind:**
  1. Write a program to create a csv file that we can directly open in MS-Excel.
  2. Read the data stored in MS-Excel file and convert it into a dictionary. The record contains rollno, name of student, marks of three subjects. Also calculate total. Display the dictionary data on the monitor.
  3. Accept contact details from the user and create a vcard that we can directly store in our mobile.
  4. Create a specific subdirectory and copy one file from another subdirectory to this newly created subdirectory.
  5. Write a program to copy contents of one file to another. While doing so, replace all lowercase characters into uppercase characters.
  6. Write a program that merges lines alternatively from two files and writes the results to new file. If one file has less number of lines than the other, the remaining lines from the larger file should be simply copied into the target file.
  7. If an **Employee** object contains following details**:**

empcode, empname, Date of Joining, Salary

Write a program to serialize and deserialize this data.

* 1. Given a text file, write a program to create another text file deleting the words ‘a’, ‘the’, ‘an’ and replacing each one of them with a blank space.