BIG DADDY'S PYTHON TOOLBOX For 3.5

...plus a LOT LOT more at: www.wikipython.com

Data on Disk

Structure, method, syntax and examples **assume the following**. **(1)** Variables "**pyFile**", "**pyDir**", "**pyPath**", "**txList**" & "**dataList**" hold data file names [i.e., **Mary.txt** or **myCsv.csv**], directory path ["**D:\\Pyfiles**"], ***pyDir+pyfile,*** and strings of text in a list, **respectively**. **(2)** File names and directory locations are hard coded by the programmer. **(3)** Read, write, append or attribute retrieval below is **done inside a "with" command structure, not shown,** except where noted. **(4)** Data written and retrieved is data type "text" unless otherwise specified. **(5)** The file object created is called "**myFile**" (6) you import modules as needed.

Create and Open a New File Object
with open(pyPath,'w+')as myfile
Append to an existing file
with open(pyPath,'a')as myfile

Write Text Strings to a File Object
```
for line in txList:
    myFile.write(line)
```

Create a New Empty File Object
```
with open(pyPath,'a') as emptyfile
    emptyfile.close  # just being
```
cautious-**with** closes file automatically

Create/Open/Write **not** using **with**
```
myFile= open(pyPath, 'w')
myFile.write("This is line 1. \n")
myFile.close()  #MUST manually
close the file!
```

Ways to Read Access a File
(1) **Loop** through it (2)**.readline**()
(3) **.read**() (4) **.readlines**() (5) .list()

Read by Looping
```
for line in myfile:
    print(line, end="")
    (process line here)
```
String == compare must include "\n" at the end for compare to work.

.readline() - process a line at a time
```
getAline="initialize the variable"
    while not(getAline==""):
        getAline=myFile.readline()
            if ...etc...:
```

.read() - into list of strings .seek(0)
txList=[myfile.read()] *exhausts pointer - now at EOF. Reads all lines into a single joined line with newline ('\n') dividers
myFile.seek(0) reset pointer to start
txtList=myfile.read() *reads individual letters of strings to list items
myFile.seek(0) reset pointer to start
To remove \n in read use .splitlines()
txtList=myFile.read().splitlines() - will put discrete lines without newline as items in txtList

.readlines() or list()
- get whole file at once
txList=myFile.readlines() or
txtList=list(myfile)
will put discrete lines with newline as items in txtList; no splitlines() support

Basic **Modes**: '**r**'-read only '**r+**'-read or write '**w**'-write only, over-writes existing file '**w+**' read or over-write '**a**'-append '**a+**'-append or read; add '**b**' to anything to invoke binary format; w,wb,w+,wb+,a,ab,a+,ab+ create new file if file does not exist

Useful File/Path/Directory Tools
Get current Directory
holdCurDir=os.getcwd()
Change directory to 'path'
os.chdir('path')
Check for current filepath
if os.path.exists(pyDir):
Check for current file
if os.path.isfile(pyPath)
Get list of directories in path
dirList=os.listdir(path)
Get list of entries in path directory
os.listdir(path)

import CSV-Comma Seperated Values

Use a standard "with open" as "myFile" structure, then create a csv.reader object or a csv.writer object.
csv.reader (csv file object [, dialect='excel'] [, optional parameter override]) - iterates the file and then....
.__next__() reads the subsequent lines (rows/records) of data
.line_num returns number of lines read
csv.writer (csv file object, dialect= 'excel', formatting parameters) -writing series of lists, each a record/table row
.writerow(row) write a row formatted by dialect (on existing file overwrite or append)
.writerows(rows) write all rows

Create csv reader obj (example)
mycsvr=csv.reader(myFile, dialect= 'excel')
Get next row of data
Row1=mycsvr.__next__()
Create csv writer obj (example)
Mycsvr=csv.writer(myFile, dialect='excel')
Write or Append csv data
mycsvr.writerow(dataList) or
Mycsvr.writerows(dataList)

import pickle - serialized objects

Uses standard "with open" structure - must be opened for **binary** operations
To **dump** (save) an object/file:
pickle.dump(object-to-pickle, save-to-file, protocol=3, fix_imports=True)
EX: pickle.dump (someList,myFile)
To **load** (retrieve) an object/file:
pickle.load(file-to-read [, fix_imports= True] [, encoding="ASCII"][, errors= "strict"])
EX: myList = pickle.load(myFile)

import sqlite3 - SQL database

Create **connection** object
sq3con = sqlite3.connect ('mysqlFile.db' [,detect_types]) **also:**
sq3con = sqlite3.connect (":memory:")
A few connection object methods:
.cursor (see below) **.commit()**
.rollback() **.close()** **.iterdump()**
Create **cursor** object
CurObj = **sq3con.cursor()**
A cursor object methods and attributes:
.fetchone() **.fetchmany(size)**
.fetchall() **.close)** **.rowcount** **.lastrowid**
.execute("sql [,parameters]")
EX: Curobj.execute("CREATE TABLE table_name (col_name data_type,...)")
.executemany("sql [,parameters]")
Notes: sql statements are case insensitive. Multiple statements are separated by semicolons (;). SQL ignores white space. Parameters are separated by commas but a comma after the last parameter causes a error.
Create database
Connection creates if it does not exist
A few SQL commands to **.execute**
CREATE TABLE
ALTER TABLE
DROP TABLE
INSERT INTO table_name VALUE(vals.)
REPLACE search_str, sub_str, rep_with
UPDATE table_name SET col_name = new_value WHERE limiting conditions
DELETE FRROM col_name WHERE
SELECT col_name FROM table WHERE
Data types (**Python**:SQL)
None:NULL **int**:INTEGER **float**:REAL
str:TEXT **bytes**:BLOB

Dictionary Read/Write (csv module) - see https://docs.python.org
csv.DictReader(csvfile, fieldnames=None, restkey=None, restval=None, dialect='excel', *args, **kwds)
csv.DictWriter(csvfile, fieldnames, restval='', extrasaction='raise', dialect='excel', *args, **kwds)

comments and suggestions appreciated: john@johnoakey.com