1 File 'IO' Operations:

page 131

In [1]:
```
fy = open("D:/_All/1.Data sets/13.emp12.csv",'r')
st = fy.read(100)       # Here, it will read 100 bytes
print(st)
len(st)
```

```
1231,Sheethal,9874512510,6/6/1985,Software developer,25000,Hyderbad,7,No
1232,Sonu,7458213640,5/5/19
```

Out[1]:
```
100
```

In [2]:
```
st = fy.readline()
print(st)
```

```
95,Software tester,28000,Madhapur,8,No
```

In [3]:
```
st = fy.readlines()
print(st)
```

```
['1233,Sandeep,9550025123,3/6/1990,Software architect,100000,Gachibowli,7.5,No\n', '1234,Manoj,9254762153,1/7/1990,s
ystem adminstrator,65000,Hitech city,8,No\n', '1235,Prasanna,9949171267,6/10/1995,software analyst,15000,Jubliee hil
ls,12,Yes\n', '1236,Priyanka,7095724262,6/8/1995,tester,20000,Banjara Hills,9,No\n', '1237,Shravanthi,9584672135,2/
9/1995,Cloud analyst,28000,Dilsuknagar,7,No\n', '1238,Yamini,8657215420,2/6/1995,HR Manager,19000,Dilsuknagar,8.5,No
\n', '1239,Chaitanya,9125467283,7/4/1994,Architect,25000,Paradise,11,Yes\n', '1240,Leela Krishna,8542136571,2/3/199
4,Software Tester,26000,Secunderabad,12,Yes\n', '1241,Mahender,9676521477,4/5/1985,Developer,60000,ECIL,7,No\n', '12
42,Yelishetty Shankar,9959243211,4/6/1979,Manager,100000,LBNagar,13,Yes\n', '1243,Swapna,9949343211,2/7/1982,Associa
te consultant,23000,LBNagar,8,No\n', '1244,Nikhil,8019200557,4/8/1994,financial consultant,25000,Tarnaka,8,No\n', '1
245,Sai Kumar,9885662939,4/9/1986,HR Manager,30000,Suchitra,9,No\n']
```

2 WRITE( ): Erase Old Data & Dump the New Data:

In [4]:
```
# Here, i want to write some data
# D:/Dataset1/i45.csv
# and another thing, Whenever the file is open -->
# we can't Write. So, we have to close the destination file in drive & write.
# Here, i don't have the file 'i45', Now here it will create the file & Write the file


wr =open("D:/_All/1.Data sets/i45.csv",'w')
wr.write("This is first stmt \n") # Now here, What content you want to Write
wr.write("This is second stmt \n")
wr.close()   # now the file has been created
```

APPEND: ADD the Data for the Existing:

In [5]:
```
awr = open("D:/_All/1.Data sets/i45.csv",'a')
awr.write("This is Third stmt \n")
awr.close()
```

In [6]:
```
# Here i'm Re-Writing, So, while using Write() --> Erase Old Data & Dump the New Data.
wr =open("D:/_All/1.Data sets/i45.csv",'w')
wr.write("This is 4 fourth stmt \n")
wr.write("This is 5-- fifth stmt \n")
wr.close()
```

3 READ DATA IN FILE - LINE BY LINE: Using for loop:

In [7]:
```
with open("D:/_All/1.Data sets/13.emp12.csv",'r') as fp:  # "D:/Dataset1/13.emp12.csv" --> as 'fp'
    for line in fp:     # Record by Record by using an a for loop:
        print(line)
```

```
1231,Sheethal,9874512510,6/6/1985,Software developer,25000,Hyderbad,7,No

1232,Sonu,7458213640,5/5/1995,Software tester,28000,Madhapur,8,No

1233,Sandeep,9550025123,3/6/1990,Software architect,100000,Gachibowli,7.5,No

1234,Manoj,9254762153,1/7/1990,system adminstrator,65000,Hitech city,8,No

1235,Prasanna,9949171267,6/10/1995,software analyst,15000,Jubliee hills,12,Yes

1236,Priyanka,7095724262,6/8/1995,tester,20000,Banjara Hills,9,No

1237,Shravanthi,9584672135,2/9/1995,Cloud analyst,28000,Dilsuknagar,7,No

1238,Yamini,8657215420,2/6/1995,HR Manager,19000,Dilsuknagar,8.5,No

1239,Chaitanya,9125467283,7/4/1994,Architect,25000,Paradise,11,Yes

1240,Leela Krishna,8542136571,2/3/1994,Software Tester,26000,Secunderabad,12,Yes

1241,Mahender,9676521477,4/5/1985,Developer,60000,ECIL,7,No

1242,Yelishetty Shankar,9959243211,4/6/1979,Manager,100000,LBNagar,13,Yes

1243,Swapna,9949343211,2/7/1982,Associate consultant,23000,LBNagar,8,No

1244,Nikhil,8019200557,4/8/1994,financial consultant,25000,Tarnaka,8,No

1245,Sai Kumar,9885662939,4/9/1986,HR Manager,30000,Suchitra,9,No
```

4 READING FOLDERS, SUB-FOLDERS, FILES USING WALK( ) AND FOR LOOP:

page 139 blue notebook

In [8]:
```
# Reading floders,Sub-folders,and files in the particular location:
import os
for folder,sub_folders,files in os.walk("E:/"):
    print("currently looking at folder:"+folder)
    print('\n')
    print("THE SUB FOLDERS ARE:\n")
    for sub_fold in sub_folders:
        print("\t Sub Folders_are: "+sub_fold)

    print("\n")
    print("THE FILES ARE:")
    for f in files:
        print("\t File: "+f)
    print('\n')
```

```
currently looking at folder:E:/


THE SUB FOLDERS ARE:

         Sub Folders_are: $RECYCLE.BIN
         Sub Folders_are: CAR PENDRIVE SONGS
         Sub Folders_are: J7MAX DOWNLOADS
         Sub Folders_are: PICS DCIM CAMERA
         Sub Folders_are: Pictures
         Sub Folders_are: PJP CAMERA J7MAX
         Sub Folders_are: PJP SCREENSHOTS J7MAX FOLDER
         Sub Folders_are: System Volume Information


THE FILES ARE:
         File: 1626522318571.jpg


currently looking at folder:E:/$RECYCLE.BIN
```

5 MODULES IN PYTHON:

In [10]:
```
# MODULES REFER TO FILE CONTAINING: PYTHON STATEMENT & DEFINITIONS:
# PYTHON CODE for example: ABC.py is called as an a MODULE:
# WE CALL MODULE TO WORKING ENVIRONMENT WITH THE HELP OF IMPORT KEYWORD:
# WE USE IMPORT KEYWORD TO IMPORT MODULE & ALLOW TO REFER IT'S OBJECTS like
# VARIABLES,VALUES,FUNCTIONS AND CLASSES:
```

In [11]:
```
import math
print(math.pi)
```

```
3.141592653589793
```

In [12]:
```
import datetime
datetime.datetime.now()  #datetime(MODULE).datetime(OBJECT)
```

Out[12]:
```
datetime.datetime(2023, 4, 23, 19, 21, 11, 130412)
```

In [13]:
```
# BY USING 'ALIAS NAME':
```

In [14]:
```
import math as m
print(m.pi)
```

```
3.141592653589793
```

In [15]:
```
# 'RANDOM NUMBER' with 'int' possibility:
```

In [16]:
```
# now again i'm re-running here means, the output will change to 13,12,10....
# Means, in-between 10 to 15, Sum random number is generated,which is in 'int' state only.

import random
print(random.randint(10,15))
```

```
13
```

In [17]:
```
#calling particular object from the MODULE;
# MATH IS HUGE MODULE LIKE pivalues,sign value, time value, math logics, umlimited logics
```

In [18]:
```
import math
import math as m
from math import *
from math import pi
print("value of pi is :"+str(pi))
```

```
value of pi is :3.141592653589793
```

In [19]:
```
# We will also get the TIME in nano seconds
```

In [20]:
```
from math import sin,cos,tan,pi,e
```

In [21]:
```
import time
print(pi)
print(sin(45))
print(cos(95))
print(time.time())
```

```
3.141592653589793
0.8509035245341184
0.7301735609948197
1682257923.001742
```

In [22]:
```
# ROUND(): round the Number:
```

In [23]:
```
x = 1.36
round(x,1)
```

Out[23]:
```
1.4
```

In [24]:
```
# ROUND WITH TWO DECIMALS
```

In [25]:
```
round(2.69875,2)
```

Out[25]: 2.7

In [26]:
```
# FLOOR: floor also comes for Round off only
```

In [27]:
```
math.floor(-1.7)
```

Out[27]: -2

In [28]:
```
# HANDLE datetime:
# we can print individually. here we are giving manually
# but if we give 'datetime.now()' means, it will print current time(system time)
```

In [29]:
```
import datetime
t = datetime.time(4,20,1)
print(t)
print('Hours:',t.hour)
print("Minutes:",t.minute)
print("seconds:",t.second)
print("MicroSeconds:",t.microsecond)
```

```
04:20:01
Hours: 4
Minutes: 20
seconds: 1
MicroSeconds: 0
```

In [30]:
```
# here Ctime means current time
# we call today as 'date one', but not the time one
# that's why it is not working on the TIME
```

In [31]:
```
today = datetime.date.today()
print(today)
print('ctime:',today.ctime())
print('tuple:',today.timetuple())
print('year:',today.year)
print('month:',today.month)
print('day:',today.day)
```

```
2023-04-23
ctime: Sun Apr 23 00:00:00 2023
tuple: time.struct_time(tm_year=2023, tm_mon=4, tm_mday=23, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=6, tm_yday=113, t
m_isdst=-1)
year: 2023
month: 4
day: 23
```

6 COLLECTION MODULES:

page 147

In [32]:
```
# Some tricky and confusion modules call as COLLECTION MODULES:
# in this collection modules, we are goin to split data: by using directly name called
# "collection module naming convention"
# 'Couter'(here 'C' is Capital)-> it will count the
# no.of recurrences, counting numbers, counting the objects:
# here Datetime, tuple also we have:
```

In [33]:
```
from collections import Counter
```

In [34]:
```
lst =[1,2,1,2,3,4,5,4,3,2,1,2,3,4,5,6,4,3,2,1]
```

In [35]:
```
Counter(lst)
```

Out[35]:
```
Counter({1: 4, 2: 5, 3: 4, 4: 4, 5: 2, 6: 1})
```

In [36]:
```
Counter('jslkjojwojmnvnclxninvciewndnlnovncbxzbsdefxc')
```

Out[36]:
```
Counter({'j': 4,
         's': 2,
         'l': 3,
         'k': 1,
         'o': 3,
         'w': 2,
         'm': 1,
         'n': 8,
         'v': 3,
         'c': 4,
         'x': 3,
         'i': 2,
         'e': 2,
         'd': 2,
         'b': 2,
         'z': 1,
         'f': 1})
```

In [37]:
```
s = "how are you, i am fine, how about you , how are you"
words = s.split()  # Here, we are spliting the Data:
Counter(words)
```

Out[37]:
```
Counter({'how': 3,
         'are': 2,
         'you,': 1,
         'i': 1,
         'am': 1,
         'fine,': 1,
         'about': 1,
         'you': 2,
         ',': 1})
```

In [38]:
```
# MOST COMMON:
```

In [39]:
```
c = Counter(words)
c.most_common(3)  # 3 common words i want to get it here
```

Out[39]:
```
[('how', 3), ('are', 2), ('you', 2)]
```

In [40]:
```
c = Counter(words)
c.most_common(2)
```

Out[40]:
```
[('how', 3), ('are', 2)]
```

7 LAMBDA EXPRESSION:

page 149

In [41]:
```
# particular LAMDA EXPRESSION started from python 3.7 version onwards:
# Normally in functions - we declare "def function_name():"
# But in LAMDA EXPRESSION -> it will bit simplify the functions:
# it will works only for "Simple Function Alternate"
# Means, if Function is an a Complicated or Big one -> we can't change that function to
# an a LAMDA EXPRESSION:
```

In [42]:
```
# square is a function name:
def square(num):
    result = num ** 2
    return result
```

In [43]:
```
square(2)
```

Out[43]: 4

In [44]:
```
def squ2(num):
    return num ** 2
```

In [45]:
```
squ2(3)
```

Out[45]: 9

In [46]:
```
# now we can write like this also: we can write 'return' beside also
```

In [47]:
```
def squ3(num):return num ** 2
```

In [48]:
```
squ3(4)
```

Out[48]: 16

In [49]:
```
# Instead of declaring the function in 'def'
# we are declaring in a 'Straight line'
```

In [50]:
```
lambda num:num**2
```

Out[50]:
```
<function __main__.<lambda>(num)>
```

In [51]:
```
# one more example
#page 151
```

In [52]:
```
my_nums = list(range(1,7))
list(map(lambda num:num ** 2,my_nums))
```

Out[52]:
```
[1, 4, 9, 16, 25, 36]
```

In [53]:
```
# FILTER WITH LAMBDA EXPRESSION:
```

In [54]:
```
nums = list(range(-5,10))
list(filter(lambda n:n%2==0,nums))
```

Out[54]:
```
[-4, -2, 0, 2, 4, 6, 8]
```

8 SCOPE OF VARIABLES:

PAGE 151

In [55]:
```
# TWO TYPES:
# 1. GLOBAL VARIABLES & 2. LOCAL VARIABLES:
# GLOBAL WORK ON ALL VARIABLES
# LOCAL CANNOT WORK ON OTHER VARIABLES, BECAUSE THESE ARE LOCAL VARIABLES:
```

In [56]:
```
def fun():  # empty function
    print(a)
    def fun2():
        print(b)
```

In [57]:
```
global_var = "This is Global Variable"
def test_lise_time():
    local_var = "This is Local Variable"
    print(local_var)
    print(global_var)
```

In [58]:
```
test_lise_time()
```

```
This is Local Variable
This is Global Variable
```

In [59]:
```
global_var
```

Out[59]:
```
'This is Global Variable'
```

In [60]:
```
# it is throwing error, because this 'local_var' work upto this function only:
# it won't work behind the function.
# we have declared (lacal_var) inside the function & global outside the function:
# global_var can call anywhere:

local_var
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_18596\2670361745.py in <module>
      4 # global_var can call anywhere:
      5
----> 6 local_var

NameError: name 'local_var' is not defined
```

9 Python Errors: Build in Exceptions:

page 154

In [61]:
```
# (a) Wrong Syntax:
# (b) While Executing::error:
```

In [62]:
```
if a > 3
```

```
  File "C:\Users\my pc\AppData\Local\Temp\ipykernel_18596\3366914128.py", line 1
    if a > 3
           ^
SyntaxError: invalid syntax
```

In [63]:
```
# ERROR NAME:ERROR DESCRIPTON:

1/0     # DIVIDED BY ZERO IS NOT POSSIBLE
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_18596\3807259595.py in <module>
      1 # ERROR NAME:ERROR DESCRIPTON:
      2
----> 3 1/0     # DIVIDED BY ZERO IS NOT POSSIBLE

ZeroDivisionError: division by zero
```

In [64]:
```
open("daskjdl.txt")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_18596\2225869048.py in <module>
----> 1 open("daskjdl.txt")

FileNotFoundError: [Errno 2] No such file or directory: 'daskjdl.txt'
```

In [65]:
```
# ERROR HANDILING:/EXCEPTION HANDLING:
# IT IS SOMETHING WHERE WE ARE GOING TO CATCH THE ERROR:
```

In [66]:
```
1/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_18596\2354412189.py in <module>
----> 1 1/0

ZeroDivisionError: division by zero
```

In [67]:
```
for i in range(5):
    try:
        print(i/0)
    except ZeroDivisionError as e:    # 'e' as an a exception:
            print(e,"__Division by Zero wont possible")
```

```
division by zero __Division by Zero wont possible
division by zero __Division by Zero wont possible
division by zero __Division by Zero wont possible
division by zero __Division by Zero wont possible
division by zero __Division by Zero wont possible
```

In [68]:
```
# NOTE:
# import sys:  to get all types of errors or exception names:
```

In [69]:
```
import sys
lst =["b",0,2]
for entry in lst:
    try:
        print("This entry is :",entry)
        r = 1/int(entry)
    except:
        print("oops...!",sys.exc_info()[0],"occured")
        print("Next Entry")
        print("*****************************************")
print("The reciprocal of : ", entry," is ",r)   # Here i'm coming out of the loop:
```

```
This entry is : b
oops...! <class 'ValueError'> occured
Next Entry
*****************************************
This entry is : 0
oops...! <class 'ZeroDivisionError'> occured
Next Entry
*****************************************
This entry is : 2
The reciprocal of :  2  is  0.5
```

Python Debug Process:

page 163

In [70]:
```
# PYTHON DEBUG :
# Helps us to Execute the logic or to identify the 'Logical Errors' when we go for python debugging:
# Debug = finding Bug:
# we can execute step by step, by using 'set_trace()' of python debug:
```

In [71]:
```
def sq2(n):
    for i in range(n):
        print(i)
    return
```

In [72]:
```
sq2(5)
```

```
0
1
2
3
4
```

In [1]:
```
# page 165 vvimp

import pdb

def sq4(n):
    for i in range(n):
        pdb.set_trace()   # Break point
        print(i)
    return
```

In [2]:
```
sq4(5)
```

```
> c:\users\my pc\appdata\local\temp\ipykernel_8536\717080291.py(8)sq4()

ipdb> c
0
> c:\users\my pc\appdata\local\temp\ipykernel_8536\717080291.py(7)sq4()

ipdb> q
```

```
---------------------------------------------------------------------------
BdbQuit                                   Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8536\805674923.py in <module>
----> 1 sq4(5)

~\AppData\Local\Temp\ipykernel_8536\717080291.py in sq4(n)
      5 def sq4(n):
      6     for i in range(n):
----> 7         pdb.set_trace()   # Break point
      8         print(i)
      9     return

~\AppData\Local\Temp\ipykernel_8536\717080291.py in sq4(n)
      5 def sq4(n):
      6     for i in range(n):
----> 7         pdb.set_trace()   # Break point
      8         print(i)
      9     return

~\anaconda3\lib\bdb.py in trace_dispatch(self, frame, event, arg)
     86             return # None
     87         if event == 'line':
---> 88             return self.dispatch_line(frame)
     89         if event == 'call':
     90             return self.dispatch_call(frame, arg)

~\anaconda3\lib\bdb.py in dispatch_line(self, frame)
    111         if self.stop_here(frame) or self.break_here(frame):
    112             self.user_line(frame)
--> 113             if self.quitting: raise BdbQuit
    114         return self.trace_dispatch
    115

BdbQuit:
```

In [3]:
```
# Here, we are Reading the 'Data' and 'Column' : VVIMP

f = open("D:/_All/1.Data sets/13.emp12.csv",'r')
for x in f:
    row = x.split(',')
    salary = int(row[5])
    print("Total Salary for : "+row[1]+" is "+str(salary))

    DA = salary * 0.25
    print("DA is: "+str(DA))
    TZ = salary * 0.15
    print("TZ is: "+str(TZ))
    HRA = salary * 0.16
    print("HRA is: "+str(HRA))
    CCA = salary * 0.02
    print("CCA is: "+str(CCA))
    MA = salary * 0.05
    print("MA is: "+str(MA))
    LIC = salary * 0.05
    print("LIC is: "+str(LIC))
    Net_sal = DA+TZ+HRA+CCA+MA+LIC
    print("Net_Salary for: "+row[1]+" is "+str(Net_sal))
    print("\n")
```

```
Total Salary for : Sheethal is 25000
DA is: 6250.0
TZ is: 3750.0
HRA is: 4000.0
CCA is: 500.0
MA is: 1250.0
LIC is: 1250.0
Net_Salary for: Sheethal is 17000.0


Total Salary for : Sonu is 28000
DA is: 7000.0
TZ is: 4200.0
HRA is: 4480.0
CCA is: 560.0
MA is: 1400.0
LIC is: 1400.0
Net_Salary for: Sonu is 19040.0
```

In [ ]:

In [ ]: