|  |
| --- |
| **SRM UNIVERSITY**  **DEPARTMENT OF COMPUTER SCIENCE** |
| **15IT322E – PYTHON PROGRAMMING** |
| **TUTORIAL RECORD** |
| **B. TECH V SEMESTER** |
| |  | | --- | |  |  |  |  | | --- | --- | | **REGISTER NUMBER** |  | | **NAME** |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week** | **Date** | **Name of the Experiment** | **Mark** | **Sign** |
| **0** |  | **Python Installation, Environment** |  |  |
| **Preliminaries** |
| **1** |  | **Lists and Tuples** |  |  |
| **2**  **3** |  | **Dictionaries and Strings** |  |  |
| **3** |  | **Classes** |  |  |
| **4** |  | **File Handling** |  |  |
| **5** |  | **Threads** |  |  |
| **6** |  | **Socket Programming** |  |  |
| **7** |  | **CGI scripts** |  |  |
| **9** |  | **GUI application using TKinter** |  |  |
| **10** |  | **Databases** |  |  |

**INDEX**

# Preliminaries

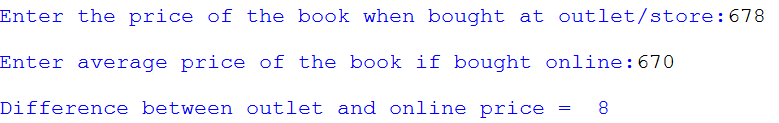
# Ungraded Introductory Experiments

# Part I – Operators and Expressions

1 Loss or Profit

Write a program to calculate the loss or profit (i.e., difference on price) in a book purchase at outlet or online. The program gets the price of the book you bought at outlet as input. Then it asks for the average price of the book, if bought online. The program computes the difference between these prices and displays the output.

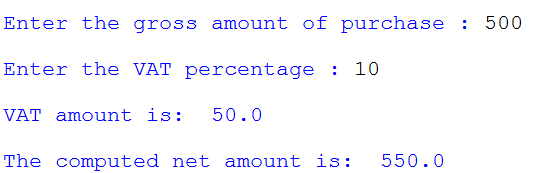
*Sample Output:*



2. VAT Calculator

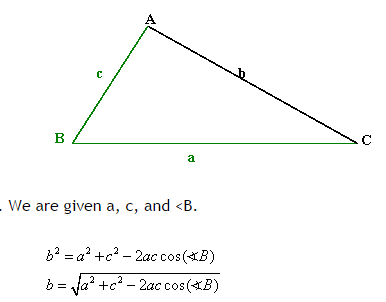
Write a python program for VAT Calculation. The program gets the net payable amount as input from the user. Then the program asks for VAT percentage from the user. The program computes the net amount and VAT amount and displays the result to the user.

*Sample Output:*



3. Side of a triangle

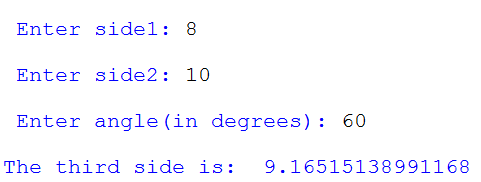
Write a python program to compute the third side of a triangle when two sides and an angle are given.



The program gets side1, side2 and angle 1 as input from the user and computes the side 3, using the above formula.

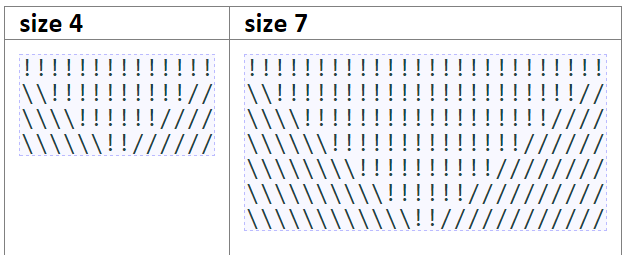
Hint : import math module. Use functionalities like sqrt, cos from the module. Cos function expects the parameter to be in radians. Convert the degress got from user to radians using the function, radians(angle)

*Sample Output:*



# Part II – Flow Controls & Functions

1. Write a program using nested loops to print the below pattern. Get the number of lines from user.



1. Zellers Algorithm

Write a program to compute Zeller’s algorithm, which is used to tell the day of the week, provided a date is given.

Ask the user for the month as a number between 1 – 12 where March is 1 and February is 12.

If born in Jan or Feb, enter previous year.

Zeller’s algorithm computation is defined as follows:

Let A, B, C, and D denote integer variables that have the following values:

A = the month of the year, with March having the value 1, April the value 2,

December the value 10, and January and February being counted as months 11 and

12 of the preceding year (in which case, subtract 1 from C)

B = the day of the month (1, 2, 3, … , 30, 31)

C = the year of the century (e.g. C = 89 for the year 1989)

D = the century (e.g. D = 19 for the year 1989)

Note: if the month is January or February, then the preceding year is used for computation. This is because there was a period in history when March 1st, not January 1st, was the beginning of the year.

Let W, X, Y, Z, R also denote integer variables. Compute their values in the following

order using integer arithmetic:

W = (13\*A - 1) / 5

X = C / 4

Y = D / 4

Z = W + X + Y + B + C - 2\*D

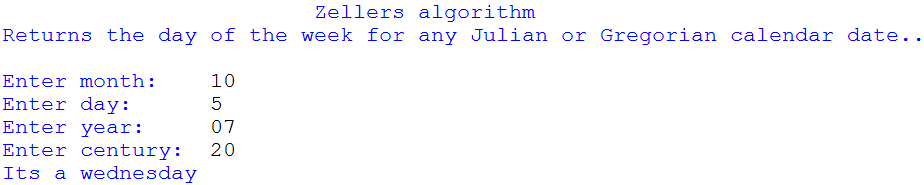
R = the remainder when Z is divided by 7

Hint: Use integer division “//” for division

Now the value of “R” is used to find the day of the week.

If R is zero, then it is a Sunday. If R is one, then it is Monday and so on. When R = 6, the day is Saturday.

*Sample Output:*



1. Luhn’s algorithm

Write a program to verify valid credit card number. A valid card number passes a digit-sum test known as the Luhn checksum algorithm. Luhn's algorithm states that if you sum the digits of the number in a certain way, the total sum must be a multiple of 10 for a valid number. Systems that accept credit cards perform a Luhn test before contacting the credit card company for final verification.

The algorithm for summing the digits is the following. Consider each digit of the credit card to have a zero-based index: the first (starting from right) is at index 0, and the last is at index 15. Start from the rightmost digit and process each digit one at a time. For digits at even-numbered indexes (the 14th digit, 12th digit, etc.), simply add that digit to the cumulative sum. For digits at odd-numbered indexes (the 15th, 13th, etc), double the digit's value, then if that doubled value is less than 10, add it to the sum. If the doubled number is 10 or greater, add each of its digits separately into the sum.

The following pseudocode describes the Luhn algorithm to sum the digits:

4408041254369873 is an example credit card number that passes the Luhn algorithm. The following figure shows the algorithm summing the latter number in detail. Notice how digits at even indexes are doubled and potentially split into two digits if they exceed 10 when doubled. For example, the number 7 at index 8 which is doubled to 14 which split to make 1+4.

|  |
| --- |
| An example checksum using the Luhn algorithm. |
| CC # 4408 0412 5436 9873  4 4 0 8 0 4 1 2 7 4 3 6 9 8 5 3  Scale \*2 \*2 \*2 \*2 \*2 \*2 \*2 \*2  --------------------------------------------------------------------  8 4 0 8 0 4 2 2 14 4 6 6 18 8 10 3  Sum = 8 + 4 + 0 + 8 + 0 + 4 + 2 + 2 + 1+4 + 4 + 6 + 6 + 1+8 + 8 + 1+0 + 3  = 70  70 is divisible by 10, therefore this card number is valid. |

Write a program where the user can type in a credit card number and receive a message stating whether the number was valid. The program should have a function validate\_number() that takes the credit card number as argument and prints the message “Valid Credit Card Number” or “Invalid Credit Card Number” accordingly. The program should print an error message and exit if the length of the credit card number is not equal to 16.

Sample Output:



**Program & Sample Output:**

**PART I:**

**1)**

price=int(input("Enter the price of the book when bought at outlet/store:"))

avg=int(input("enter average price of the book if bought online:"))

print("difference between outlet and online price =",price-avg)

**2)**

amt=int(input("enter the gross amount of purchase"))

vat=int(input("enter the vat percentage"))

vat\_amount=amt\*(vat/100)

print("the vat amount is:",vat\_amount)

net\_amount=amt+vat\_amount

print("the computed net amount is:",net\_amount)

**3)**

import math

side1=int(input("enter the side1"))

side2=int(input("enter the side12"))

angle1=int(input("enter the angle1"))

side3=math.sqrt((side1\*\*2)+(side2\*\*2)-(2\*side1\*side2\*math.cos(math.radians(angle1))))

print("the third side is",side3)

**4)**

s= int(input("enter size"))

for i in range(s):

for j in range(2\*i):

print("\\",end="")

for k in range (4\*s-(4\*i)-2):

print ("!",end="")

for l in range (2\*i):

print("/",end="")

print("")

**5)**

print ("Zeller's Algorithm")

print("Enter Your Day Value")

B=int(input(‘’))

print("Enter Your Month Value")

A=int(input(‘’))

print("Enter Your Year Value")

C=int(input(‘’))

print("Enter Your Century Value")

D=int(input(‘’))

W=(13\*A-1)/5

X=C/4

Y=D/4

Z=W + X + Y + B + C - 2 \* D

R=int(Z % 7)

if R==0:

print ('Sunday')

elif R==1:

print ('Monday')

elif R==2:

print ('Tuesday')

elif R==3:

print ('Wednesday')

elif R==4:

print ('Thursday')

elif R==5:

print ('Friday')

elif R==6:

print ('Saturday')

**6)**

def validate\_number(n):

s=n

sum1=0

if len(s) == 16:

for x in range(16):

new=int(s[x])

if x % 2 ==0:

new=new\*2

if new >=10:

sum1=sum1+int(new/10)

sum1=sum1+int(new%10)

else:

sum1=sum1+new

else:

sum1=sum1+new

print(sum1)

if sum1 % 10 ==0:

print("valid!!",sum1)

else:

print("invalid!!")

num=input("enter the number")

validate\_number(num)

**Result:**

Thus, the programs were executed successfully**.**

**Experiment 1**

**Lists and Tuples**

Graded Questions:

1a) Lists

Get a list of numbers from the user, till he types “end”. Write two functions “bubble\_sort” and “insertion\_sort”, to sort the function using bubble sort and insertion sort. Arguments to the function would be user entered list. The function returns the sorted list according to the algorithm.

Hint – Working of algorithms

Bubble Sort:

* Basic idea is to bubble up the largest(or smallest), then the 2nd largest and the the 3rd and so on to the end of the list.
* Each bubble up takes a full sweep through the list
* Do the sorting in place. Do not create a new list.

Insertion Sort:

* Take element by element and insert it at the right position
* Do the sorting in place. Do not create a new list.

1b) Tuples

Get a list of non-empty tuples from the user. You can use any end of sequence representation for tuples and list. Write a function sort\_tuple(). Input to the function is the list of tuples entered by the user. Output from the function is a sorted list according to the condition - sorted in increasing order by the last element in each tuple.

e.g. [(1, 7), (1, 3), (3, 4, 5), (2, 2)] yields

[(2, 2), (1, 3), (3, 4, 5), (1, 7)]

**Program & Sample Output - 1a**

numbers=[]

char=input("enter the number [or type end]")

while char != "end":

n=int(char)

numbers.append(n)

char=input("enter the number [or type end]")

def bubble\_sort ( num ):

for i in range(len(num)-1,0,-1):

for j in range(i):

if num[j]> num[j+1]:

temp= num[j]

num[j]= num[j+1]

num[j+1]= temp

return num

def insertion\_sort(num):

for i in range(1,len(num)):

j = i

while j > 0 and num[j] < num[j-1]:

num[j], num[j-1] = num[j-1], num[j]

j=j-1

return num

sorted\_numbers1 = bubble\_sort(numbers)

sorted\_numbers2 = insertion\_sort(numbers)

print("sorted by bubble sort", sorted\_numbers1)

print("sorted by insertion sort", sorted\_numbers2)

**Program & Sample Output – 1b**

user\_list=[]

check="y"

while check=="y":

user\_tuple=()

n=int(input("enter the tuple numbers or [end]"))

while n !="end":

user\_tuple += (int(n),)

n=(input("enter the tuple numbers or [end]"))

user\_list.append(user\_tuple)

check=input("want to add more tuples??(y) or (n)")

print("list before is",user\_list)

def sort\_tuple( lt ):

dic={}

for x in lt:

dic[x[-1]]=x

key=sorted(dic)

sorted\_user\_list=[]

for x in range(len(dic)):

sorted\_user\_list.append(dic[key[x]])

print("sorted is",sorted\_user\_list)

sort\_tuple(user\_list)

**Result:** Thus, the programs were executed successfully**.**

|  |  |
| --- | --- |
| **Ex. No: 1** | **Title: Lists and tuples** |
| **Program (5)** |  |
| **Coding Conventions (2)** |  |
| **Output (3)** |  |
| **Total (10)** |  |
| **Comments:** | **Signature:** |

**Additional Questions (Optional):**

1. Animate any object to move across the screen in any manner. Use Pygame. Pygame uses tuples extensively for screen co-ordinates representation.

PyGame Steps:

* 1. Initialize pygame
  2. Set the display window size
  3. Set the background colour of the window
  4. Load the image
  5. Blit the image on the display window at the initial (x.y) e.g 10,10
  6. Set a direction flag to indicate which co-ordinate to update.
  7. Create an infinite loop to display the image continuously.
  8. Adjust/increment the position of the image according to the window size and change the direction if it reaches the end.
  9. Set a clock and tick option to wait for sometime to avoid fast display change
  10. Inside the while loop, write code to quit in case of close button click.

To start with use the anim\_template.py from the course classoom, which has the structure of a pygame animation program. Fill the sections to get a animation display.

1. Get a list of number from the user (unsorted). Construct max and min heap from the unsorted list and output the list for min and max heap.

**Result:**

|  |  |
| --- | --- |
| **Ex. No : 1** | **Title : Lists and Tuples** |
| **Program (5)** |  |
| **Coding Conventions (2)** |  |
| **Output (3)** |  |
| **Total (10)** |  |
| **Comments:** | **Signature:** |

**Experiment 2**

**Dictionaries and Strings**

**Graded Questions:**

2a) Finding the most popular friend from dictionaries

Individual Student Friend Rating Score Dictionaries, (Construct a min of 10 student dictionaries)

Score 1 is lowest and Score of 5 is highest. Every student rates all the other students. So each dictionary has 9 elements.

Example with 3 students,

sam\_friends = {“Bob”:3, “Raj”:5}

bob\_friends = {“Sam”:5,”Raj”:1}

raj\_friends = {“Sam”:3,”Bob”:5}

Find who has the highest score and that student is the most popular friend. Print the most popular friend and his/her score.

2b) Get a string as input from the user and the sub string to be searched also from the user. Find and list the index positions of the sub string and the number of occurrences of the sub string.

**Program & Sample Output - 2a**

def conlist(namelist):

for i in range(0,len(namelist)):

ddict={}

temp={}

for j in range(0,len(namelist)):

if namelist[i] != namelist[j]:

print("enter the rating for",namelist[j],"from",namelist[i]);

temp = {namelist[j]: int(input(""))};

ddict.update(temp)

temp={}

dictlist.append(ddict)

def maxRating(namelist,dictlist):

for i in range(0,len(namelist)):

temp = [[k,v] for k,v in dictlist[i].items()]

for [key,value] in temp:

ratinglist[namelist.index(key)]+=value

max1=""

maxCount=0

for i in range(0,len(namelist)):

if maxCount<ratinglist[i]:

maxCount=ratinglist[i]

max1=namelist[i]

if max1: return max1

else: return "Everyone is the same"

namelist=[]

dictlist=[]

ratinglist=[]

n=int(input("Enter the number of people? "))

for i in range(0,n):

name=input("Enter name ")

namelist.append(name)

ratinglist.append(0)

conlist(namelist)

max1 = maxRating(namelist,dictlist)

print ("Most popular friend is: " , max1)

**Program & Sample Output – 2b:**

str1=input("Enter a string: ")

sub =input("Enter the substring: ")

index = 0

count= 0

while index < len(str1):

index = str1.find(sub, index)

if index == -1:

break

print ("%s found at %d" % (sub , index+1))

index += len(sub)

count+=1

print ("Total count = %d" % count)

**Additional Questions:**

1. Represent a matrix as nested lists. Get number of rows, number of columns and input elements from the user and represent it as matrix.

Example:

Matrix



can be represented as

mtx = [ [0,0,0,1,0], [0,0,0,0,0], [0,2,0,0,0], [0,0,0,0,0], [0,0,0,3,0] ]

Access rows and elements and print the same. Also, since the matrix is a sparse matrix, represent it using a dictionary. Construct the dictionary.

Example

mtx = {(0,3): 1, (2, 1): 2, (4, 3): 3}

* 1. The dictionary has entries for non-zero elements
  2. Key is a tuple that holds the row and column index
  3. Value is the element

1. Get a string from the user. Preprocess the string obtained using Porter Stemming algorithm

**Result:**

Thus, the programs were executed successfully

|  |  |
| --- | --- |
| **Ex. No: 2** | **Title: Dictionaries and Strings** |
| **Program (5)** |  |
| **Coding Conventions (2)** |  |
| **Output (3)** |  |
| **Total (10)** |  |
| **Comments:** | **Signature:** |

**Experiment 3**

**Classes**

**Graded Questions:**

3a)

Define a class named “Time”, with instance attributes hours, minutes and seconds. The class consists of below method, apart from constructor

input\_values() – Gets the values of attributes from the user.

print\_details() – Prints the values of the attributes

Overload the operators “+” and “-“ to add and subtract the corresponding attribute values and print accordingly.

For example creating two instances t1 and t2 for “Time” class with attribute values as below,

|  |  |  |  |
| --- | --- | --- | --- |
| Instance | Hours | Minutes | Seconds |
| t1 | 12 | 10 | 50 |
| t2 | 10 | 7 | 32 |

then, t1+ t2 should print 22:18:22 and t1-t2 should print 2:3:18

Another example,

|  |  |  |  |
| --- | --- | --- | --- |
| Instance | Hours | Minutes | Seconds |
| t1 | 20 | 18 | 31 |
| t2 | 14 | 53 | 12 |

then, t1+ t2 should print 11:11:43 and t1-t2 should print 5:25:19

Assume 24 hours clock. Ignore days manipulation.

3 b)

Write a parent class “Polygon”. This class has two attributes

no\_of\_sides - represent the number of sides . This is passed as argument to the constructor when the object is getting created.

sides – is a list representing the value of the sides, initialise the value of sides to 0 in constructor

The class has two methods

input\_sides() – Gets the sides from the user. This method would display messages like

“Enter the value for side1”

“Enter the value for side2”

….

“Enter the value for siden”

and gets the values for the sides from the user and populates the list.

print\_sides() – Prints the values of the sides

Create a child class called Triangle, that calls the Parent class constructor with 3 as the number of sides. The child class has one additional method “findArea”, that finds the area of the triangle using the formula – (side1+side2+side3)/2

**Program & Sample Output – 3a**

import math

class Time:

hours=0

minutes=0

seconds=0

def \_\_init\_\_(self):

self.hours=0

self.minutes=0

self.seconds=0

def get\_values(self):

self.hours=int(input("Enter hours "))

self.minutes=int(input("Enter minutes "))

self.seconds=int(input("Enter seconds "))

def print\_values(self):

print ("Hours:",self.hours,": Minutes:", self.minutes,"Seconds:",self.seconds)

def \_\_add\_\_(self,other):

d = Time()

d.hours = self.hours + other.hours

d.minutes = self.minutes + other.minutes

if(d.minutes>=60):

d.minutes-=60

d.hours+=1

d.seconds = self.seconds + other.seconds

if(d.seconds>=60):

d.seconds-=60

d.minutes+=1

if(d.minutes==60):

d.minutes-=60

d.hours+=1

d.hours=int(d.hours)

d.minutes=int(d.minutes)

d.seconds=int(d.seconds)

return d

def \_\_sub\_\_(self,other):

d=Time()

h1,m1,s1 = self.hours, self.minutes , self.seconds

h2,m2,s2 = other.hours , other.minutes , other.seconds

t1 = s1\*60\*60 + m1\*60 + h1

t2 = s2\*60\*60 + m2\*60 + h2

t3 = math.fabs(t1-t2)

h3 = t3%60

m3 = (t3/60) % 60

s3 = ((t3/60)/60) % 60

d.hours , d.minutes , d.seconds = int(h3) , int(m3) , int(s3)

return d

t1 = Time()

t1.get\_values()

t2= Time()

t2.get\_values()

t3 = t1 + t2

t3.print\_values()

t4 = t1 - t2

t4.print\_values()

**Program & Sample Output – 3b:**

class Polygon:

no\_of\_sides = 0

sides = []

def \_\_init\_\_(self,ns):

self.no\_of\_sides=ns

def input\_sides(self):

for i in range(0,self.no\_of\_sides):

print ("Enter side",(i+1))

temp=int(input())

self.sides.append(temp)

def print\_sides(self):

for i in range(0, self.no\_of\_sides):

print ("Side no",i+1,"=" ,self.sides[i])

class Triangle(Polygon):

def \_\_init\_\_(self):

Polygon.\_\_init\_\_(self,3)

self.input\_sides()

def findArea(self):

sum1=0

for i in range(0,self.no\_of\_sides): sum1+=self.sides[i]

return (sum1)/2

t= Triangle()

print ("Area is:",t.findArea())

**Additional Question (Optional):**

1. Create a class Adder with two user defined methods listAdd() and dictAdd(), that are used to add two lists and two dictionaries respectively.

Maintain the count of objects/instances created for this class. Whenever the instance is created, a message is displayed to the user, saying <n>th object instance of Adder class is created.

Create 3 instances of this class and demonstrate the functionalities.

1. Write a parent class “Bank\_Acount”. This class has below attributes

Account ID ,AccountHolderName, Balance

The class has three methods

input\_values() – Gets the values of attributes from the user.

print\_details() – Prints the values of the attributes

computeDraftLimit() – Computes and prints draft limit value , which is 90% of balance amount.

Create a child class called “Current\_Account”. Apart from the attributes from the parent class, this child class has two additional attributes – OverDraftLimit, WaiverBalance.

The child class overrides the computeDraftLimit() method to compute and prints draft limit value , which is 120% of balance amount.

Create 2 instances – one of parent and one of child class and demonstrate the functionalities.

**Result:**

Thus, the programs were executed successfully.

|  |  |
| --- | --- |
| **Ex. No : 3** | **Title : Classes** |
| **Program (5)** |  |
| **Coding Conventions (2)** |  |
| **Output (3)** |  |
| **Total (10)** |  |
| **Comments:** | **Signature:** |

**Experiment 4**

**File Handling**

**Graded Questions:**

4 Python uses the # character to mark the beginning of a comment. The comment ends at the end of the line containing the # character. In this exercise, you will create a program that removes all the comments from a Python source file. Check each line in the file to determine if a # character is present. If it is then your program should remove all the characters from the # character to the end of the line (we’ll ignore the situation where the comment character occurs inside of a string). Save the modified file using a new name that will be entered by the user. The user will also enter the name of the input file. Ensure that an appropriate error message is displayed if a problem is encountered while accessing the files.

Example:

Enter input file name that is full of comments: comments.py

Enter the file name to output: withoutcomments.py

Comments.py

#hey this is comment

#supposed to be a py file

#lots of comments

a=1

b=2

print(a+b)

#over

WithoutComments.py

a=1

b=2

print(a+b)

**Program & Sample Output – 4**

inputFileName= input("Enter input file name that is full of comments ")

outputFileName= input("Enter the file name to output ")

try:

inFile= open(inputFileName,'r+')

inlines = inFile.readlines()

inFile.close()

outFile= open(outputFileName,'w+')

except (OSError, IOError,NameError) as e:

print (e)

for line in inlines:

if(line[0:1]=='#'): continue

else: outFile.write(line)

outFile.close()

**Additional Question (Optional):**

* + - 1. Files that have to be ready for the program – vocab.txt, stopwords.txt and book.txt. The file vocab.txt has a list of words. The file book.txt has a paragraph or a short story. The file stopwords.txt has the common words like - is, was, are, their, her, his etc. Your program should
         1. Remove the stop words from the content book.txt, referring to stopwords.txt.
         2. Find words from book.txt that are in vocab.txt. Find the frequency of occurrence of each word.
         3. Find the words from book.txt that are not in vocab.txt. List the words. Ask the user if these words could be added to vocab.txt.
         4. If the user enters yes, append the new words to vocab.txt, else end the program.
      2. Given a file name along with fully qualified path, find the inode information of the file.

**Result:**

Thus, the programs were executed successfully.

|  |  |
| --- | --- |
| **Ex. No : 4** | **Title : File Handling** |
| **Program (5)** |  |
| **Coding Conventions (2)** |  |
| **Output (3)** |  |
| **Total (10)** |  |
| **Comments:** | **Signature:** |

**Experiment 5**

**Threads**

**Graded Questions:**

1. Write a Python program to determine in a local network which addresses/computers are active. Get a list of ip addresses from the user(each ip address entered as a string). Every time we would have to wait a few seconds for the return values. A solution without threads is highly inefficient, because the script will have to wait for every ping.

**Program & Sample Output – 5**

import os

import time

from threading import Thread

def pingcheck(counter,ip):

counter+=1

print ("In thread no "+str(counter)+" pinging "+ip)

response=os.system("ping -c 1 " + ip)

if response==0: print(ip,"is active")

else: print (ip,"is inactive")

print ("Sleeping for 5 secs")

time.sleep(5)

print('slept')

return

iplist=[]

counter=0

print ("Enter the list of IPs you want to ping ")

while True:

d=input()

if d=="end": break

iplist.append(d)

**Program & Sample Output – 5**

for ip in iplist:

t= Thread(target=pingcheck, args=(counter,ip,))

t.start()

**Additional Question (Optional):**

1. Get a list of URLs from user. , Connect to the URL of a website, and print out the first 1024 bytes of the page. Use threads for each url connection.

**Result:**

Thus, the programs were executed successfully.

|  |  |
| --- | --- |
| **Ex. No: 5** | **Title: Threads** |
| **Program (5)** |  |
| **Coding Conventions (2)** |  |
| **Output (3)** |  |
| **Total (10)** |  |
| **Comments:** | **Signature:** |

**Experiment 6**

**Sockets**

**Graded Questions:**

1. Write a TCP/IP server and client program. It is a echo client program. The message typed by the client should be echoed back to the client by the server. Each client connection should be handled by a thread.

**Program & Sample Output – 6**

**Server.py**

import socket

import sys

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_address = ('localhost', 6000)

print ('starting up on',server\_address)

sock.bind(server\_address)

sock.listen(5)

while True:

print ('waiting for a connection')

connection, client\_address = sock.accept()

try:

print ('connection from', client\_address)

while True:

data = connection.recv(1024)

print ('received',data)

if data:

print ('sending data back to the client')

connection.sendall(data)

else:

print ('no more data from', client\_address)

break

finally:

connection.close()

**Program& Sample Output – 6**

**Client.py**

import socket

import sys

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_address = ('localhost', 6000)

print ('connecting to ' , server\_address)

sock.connect(server\_address)

try:

message = 'This is the message. It will be repeated.'

print ('sending', message)

sock.sendall(message)

amount\_received = 0

amount\_expected = len(message)

while amount\_received < amount\_expected:

data = sock.recv(16)

amount\_received += len(data)

print('received',data)

finally:

print('closing socket')

sock.close()

**Additional Question (Optional):**

1. Implement a UDP server and client program. It is a echo client program. The message typed by the client should be echoed back to the client by the server. Each client connection should be handled by a thread.

**Result:**

Thus, the programs were executed successfully.

|  |  |
| --- | --- |
| **Ex. No: 6** | **Title: Sockets** |
| **Program (5)** |  |
| **Coding Conventions (2)** |  |
| **Output (3)** |  |
| **Total (10)** |  |
| **Comments:** | **Signature:** |

**Experiment 7**

**CGI Scripts**

**Graded Questions:**

1. Create a web application to query the data base (Book database mentioned in experiment 5). Get the Book Name or Author Name from the user (two text boxes). When the user clicks Submit (button), fetch the query results and display it in a table. Write a local server that processes the CGI scripts.

**Program & Sample Output – 7**

from flask import Flask, render\_template, jsonify, request, Response

import sqlite3, json ,os

array=[["121","Half Girlfriend","Chetan Bhagat", "135"],["123","Harry Potter Series","JK Rowling","200"],["124","One day at a call center","Chetan Bhagat","200"],["125","Best Day Ever","Random Author","100"]]

db=sqlite3.connect("yada.sqlite")

iterator = db.cursor()

try:

iterator.execute("create table books(customerid , bookname , author, price)")

except sqlite3.OperationalError:

pass

finally:

print "Tables have been created"

def root\_dir(): # Get root dir

return os.path.abspath(os.path.dirname(\_\_file\_\_))

def get\_file(filename): # Get file name

try:

src = os.path.join(root\_dir(), filename)

return open(src).read()

except IOError as exc:

return str(exc)

def namesearch(query):

iterator.execute("select \* from books where bookname like '" + query+ "%' ;")

return iterator.fetchall()

def authsearch(query):

iterator.execute("select \* from books where author like '" + query + "%' ;")

return iterator.fetchall()

**Program & Sample Output – 7**

for i in range(0, len(array)):

iterator.execute("""insert into books values(?,?,?,?)""",(array[i]))

app = Flask(\_\_name\_\_)

@app.route("/")

def main():

return render\_template('index.html')

@app.route('/<path:path>')

def get\_resource(path):

mimetypes = {".css": "text/css", ".html": "text/html",".js": "application/javascript" }

complete\_path = os.path.join(root\_dir(), path)

ext = os.path.splitext(path)[1]

mimetype = mimetypes.get(ext, "text/html")

content = get\_file(complete\_path)

return Response(content, mimetype=mimetype)

@app.route('/books',methods=["POST"])

def search():

if request.form['type'] == "name":

result = namesearch(request.form['name']+'%')

print result

response = app.response\_class(response=json.dumps(result), status=200,mimetype='application/json’)

return response

elif request.form['type'] == "author":

result = authsearch(request.form['name']+'%')

print result

response = app.response\_class( response=json.dumps(result),status=200, mimetype='application/json')

return response

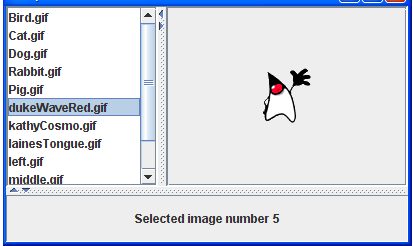
return "NULL"

if \_\_name\_\_ =="\_\_main\_\_":

app.run()

**Additional Question (Optional):**

1. Write a CGI script to display images selected and the index of the image.Write a local server that processes the CGI scripts.



**Result:**

Thus, the programs were executed successfully.

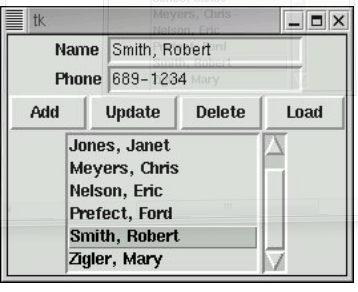
|  |  |
| --- | --- |
| **Ex. No : 7** | **Title : CGI Scripts** |
| **Program (5)** |  |
| **Coding Conventions (2)** |  |
| **Output (3)** |  |
| **Total (10)** |  |
| **Comments:** | **Signature:** |

**Experiment 8**

**GUI application using TKinter**

**Graded Questions:**

1. Write a tkinter program to edit the phone list. Create the below form. This application requires creating a SQLite table called “PhoneList” with Phone\_number (Primary Key) and Name as the columns.



Initial Display – Name and Phone are displaying the first record. Records from the table are listed in the list box below. List box should have scroll bar.

Add – Tries to add the entry in the “Name” and “Phone” fields to the database.

Update – Tries to update the name (as typed in the “Name” text box) for the phone number in the “Phone” text box.

Delete – Select an item from the list box. The details should be displayed in the Name and Phone text boxes. After selecting an item if the Delete button is clicked, the row is removed from the table and the result is displayed to the user in a separate alert box.

Load – Refreshes the list box display from the entries in the data base.

**Program & Sample Output – 8**

import cPickle

from Tkinter import \*

FILENAME = 'book.pk'

def store(book, filename=FILENAME):

with open(filename, 'w') as f:

cPickle.dump(book, f)

def load(filename=FILENAME):

with open(filename, 'r') as f:

return cPickle.load(f)

def general\_find(criteria, term):

for entry in book:

if entry[criteria] == term:

return entry

def find\_by\_name(name):

return general\_find('name', name)

def find\_by\_initials(initials):

def to\_initials(name):

return ''.join([i[0] for i in name.split(' ')])

for entry in book:

if to\_initials(entry['name']) == initials:

return entry

def add\_entry(entry):

if entry not in book:

book.append(entry)

def remove\_entry(entry):

book.remove(entry)

def string\_to\_entry(s):

tempList = s.split('|')

keyvalstrings = [tuple(i.split('=')) for i in tempList]

new\_entry = {}

for key, val in keyvalstrings:

def user\_find\_by\_initials():

i = raw\_input("Enter initials: ")

e = find\_by\_initials(i)

if e:

print\_entry(e)

else:

print " it's not there."

def user\_find\_by\_name():

n = raw\_input("Enter name: ")

e = find\_by\_name(n)

if e:

print\_entry(e)

else:

print "don't really know this person do you?"

def show\_by\_name():

n = raw\_input("Enter name: ")

e = find\_by\_name(n)

if e:

show\_entry(e)

else:

print "Cannot show GUI"

def show\_entry(entry):

def make\_label\_entry(key, value):

l = Label(parent, text=key)

e = Entry(parent)

e.insert(0, value)

return (l, e)

def grid\_widgets(low):

for i, w in enumerate(low):

w[0].grid(row=i, column=0)

w[1].grid(row=i, column=1)

def save():

for l, e in widgets:

entry[l.config()['text'][-1]] = e.get()

parent =Tk()

parent.title('Entry Display')

widgets = [make\_label\_entry(key, val) for key, val in entry.items()]

grid\_widgets(widgets)

save\_button = Button(parent, text='save', command=save)

save\_button.grid(row=len(widgets)+1, column=1)

parent.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

try:

book = load()

except IOError:

book = []

main\_loop()

**Program & Sample Output – 8**

new\_entry[key.strip()] = val.strip()

return new\_entry

def main\_loop():

while True:

choice = raw\_input("""What's next:(q) Quit(a) Add new Entry (v) View all Entries

(s) General Search(si) Search by Initials(sn) Search by Name(g) Search by Name GUI Display> """)

if choice == 'q':

break

elif choice == 'a':

user\_add\_new()

elif choice == 'v':

print\_book()

elif choice == 's':

user\_general\_find()

elif choice == 'si':

user\_find\_by\_initials()

elif choice == 'sn':

user\_find\_by\_name()

elif choice == 'g':

show\_by\_name()

else: print "You entered something incorrectly, try again."

def user\_add\_new():

s = raw\_input("""

Enter a new entry

Format: name=Bob Smith | phone= 452-2355 | email = bob@g.com> """)

try:

add\_entry(string\_to\_entry(s))

print "Successful"

except: print "went horribly wrong!"

def print\_book():

for entry in book:

print\_entry(entry)

def print\_book():

for entry in book:

print\_entry(entry)

def user\_general\_find():

k = raw\_input("Enter the key (eg name, state, etc): ")

v = raw\_input("Enter the value (eg Fred, CA, etc): ")

e = general\_find(k, v)

if e:

for key, val in e.items():

print key, '=', val

else:

print "couldn't find it at all!"

def user\_find\_by\_initials():

i = raw\_input("Enter initials: ")

e = find\_by\_initials(i)

if e:

print\_entry(e)

else:

print " it's not there."

def user\_find\_by\_name():

n = raw\_input("Enter name: ")

e = find\_by\_name(n)

if e:

print\_entry(e)

else:

print "don't really know this person do you?"

def show\_by\_name():

n = raw\_input("Enter name: ")

e = find\_by\_name(n)

if e:

show\_entry(e)

else:

print "Cannot show GUI"

def show\_entry(entry):

def make\_label\_entry(key, value):

l = Label(parent, text=key)

e = Entry(parent)

e.insert(0, value)

return (l, e)

def grid\_widgets(low):

for i, w in enumerate(low):

w[0].grid(row=i, column=0)

w[1].grid(row=i, column=1)

def save():

for l, e in widgets:

entry[l.config()['text'][-1]] = e.get()

parent =Tk()

parent.title('Entry Display')

widgets = [make\_label\_entry(key, val) for key, val in entry.items()]

grid\_widgets(widgets)

save\_button = Button(parent, text='save', command=save)

save\_button.grid(row=len(widgets)+1, column=1)

parent.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

try:

book = load()

except IOError:

book = []

main\_loop()

def user\_general\_find():

k = raw\_input("Enter the key (eg name, state, etc): ")

v = raw\_input("Enter the value (eg Fred, CA, etc): ")

e = general\_find(k, v)

if e:

for key, val in e.items():

print key, '=', val

else: print "couldn't find it at all!"

def user\_find\_by\_initials():

i = raw\_input("Enter initials: ")

e = find\_by\_initials(i)

if e:

print\_entry(e)

else:

print " it's not there."

def user\_find\_by\_name():

n = raw\_input("Enter name: ")

e = find\_by\_name(n)

if e:

print\_entry(e)

else:

print "don't really know this person do you?"

def show\_by\_name():

n = raw\_input("Enter name: ")

e = find\_by\_name(n)

if e: show\_entry(e)

else: print "Cannot show GUI"

def show\_entry(entry):

def make\_label\_entry(key, value):

l = Label(parent, text=key)

e = Entry(parent)

e.insert(0, value)

return (l, e)

def grid\_widgets(low):

for i, w in enumerate(low):

w[0].grid(row=i, column=0)

w[1].grid(row=i, column=1)

def save():

for l, e in widgets:

entry[l.config()['text'][-1]] = e.get()

parent =Tk()

parent.title('Entry Display')

widgets = [make\_label\_entry(key, val) for key, val in entry.items()]

grid\_widgets(widgets)

save\_button = Button(parent, text='save', command=save)

save\_button.grid(row=len(widgets)+1, column=1)

parent.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

try:

book = load()

except IOError:

book = []

main\_loop()

**Result:**

Thus, the programs were executed successfully.

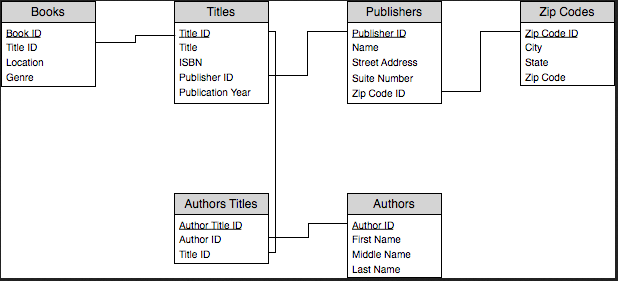
|  |  |
| --- | --- |
| **Ex. No: 8** | **Title: GUI application using TKinter** |
| **Program (5)** |  |
| **Coding Conventions (2)** |  |
| **Output (3)** |  |
| **Total (10)** |  |
| **Comments:** | **Signature:** |

**Experiment 9**

**Databases**

**Graded Questions:**

1. Create the below table in SQLite. Perform insert, update and delete rows depending on user choice.Write separate python script for table creation. Write additional script for the above actions, based on user input.

****

**Program & Sample Output – 9**

import sqlite3,json, os,sys

import pandas as pd

db=sqlite3.connect("yada.db")

iterator=db.cursor()

def createTables():

try:

iterator.execute("create table Books (BookID text PRIMARY KEY, titleID text not null, location text not null, genre text )")

iterator.execute("create table Titles (titleID text not null, title text not null, ISBN text not null, publisherID text not null, publicationYear text, foreign key(titleID) references Books(titleID))")

iterator.execute("create table publishers (publisherID text not null, name text not null, streetAdd text, suiteno text, zip text, foreign key(publisherID) references Titles(publisherID))")

iterator.execute("create table AuthorTit (authorTitID text not null, authID text not null, titleID not null, foreign key(titleID) references Titles(titleID))")

iterator.execute("create table authors (authID text not null , firstName text not null, middleName text , lastName text , foreign key(authID) references AuthorTit(authID))")

iterator.execute("create table zipCodes (zipCodeID text not null , city text, state text, zipcode text, foreign key(zipCodeID) references publishers(zip))")

db.commit()

except sqlite3.OperationalError:

pass

finally:

print "Tables have been created"

def insertInto(tID):

if(tID==0):

bookID=raw\_input("Enter BookID? ")

titleID=raw\_input("Enter titleID? ")

location=raw\_input("Enter location? ")

genre=raw\_input("Enter genre? ")

try:

iterator.execute("insert into Books values (?,?,?,?)",(bookID,titleID,location,genre))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==1):

titleID=raw\_input("Enter titleID? ")

title=raw\_input("Enter title? ")

ISBN=raw\_input("Enter ISBN? ")

publisherID=raw\_input("Enter publisherID? ")

publicationYear=raw\_input("Enter publicationYear? ")

try:

iterator.execute("insert into Titles values (?,?,?,?,?)",(titleID,title,ISBN,publisherID,publicationYear))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==2):

publisherID=raw\_input("Enter publisherID? ")

name=raw\_input("Enter name? ")

streetAdd=raw\_input("Enter street Address? ")

suiteno=raw\_input("Enter Suite Number? ")

zipx=raw\_input("Enter Zip? ")

try:

iterator.execute("insert into publishers values (?,?,?,?,?)",(publisherID,name,streetAdd,suiteno,zipx))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==3):

authorTitID=raw\_input("Enter Author Title ID? ")

authID=raw\_input("Enter Author ID? ")

titleID=raw\_input("Enter Title ID? ")

try:

iterator.execute("insert into AuthorTit values (?,?,?,?,?)",(publisherID,name,streetAdd,suiteno,zipx))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==4):

authID=raw\_input("Enter Author ID? ")

firstName=raw\_input("Enter first Name? ")

middleName=raw\_input("Enter middle Name? ")

lastName=raw\_input("Enter last Name? ")

try:

iterator.execute("insert into authors values (?,?,?,?)",(authID,firstName,middleName,lastName))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==5):

ZipCodeID=raw\_input("Enter Zip Code ID? ")

city=raw\_input("Enter city? ")

state=raw\_input("Enter State? ")

zipCode=raw\_input("Enter Zip Code? ")

try:

iterator.execute("insert into zipCodes values(?,?,?,?", (ZipCodeID,city,state,zipCode))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

def selectFromAllTables(tID):

try:

if(tID==0): iterator.execute('select \* from Books')

elif(tID==1): iterator.execute('select \* from Titles')

elif(tID==2): iterator.execute('select \* from publishers')

elif(tID==3): iterator.execute('select \* from AuthorTit')

elif(tID==4): iterator.execute('select \* from authors')

elif(tID==5): iterator.execute('select \* from zipCodes')

print pd.DataFrame(iterator.fetchall())

except pd.io.sql.DatabaseError , AttributeError:

print "No such records found"

createTables()

while True:

print "1. Insert into Tables "

print "2. Display All Tables "

print "3. Exit"

subchoice=int(raw\_input("Enter choice? "))

if subchoice == 3:

db.close()

break

print "You want to operate on which Table?"

print "1. Books Table"

print "2. Titles Table"

print "3. Publishers Table"

print "4. Author Table"

print "5. Authors Table"

print "6. ZipCodes Table"

choice=int(raw\_input("Enter the table? ")) - 1

if(subchoice==1): insertInto(choice)

elif(subchoice==2): selectFromAllTables(choice)

else: print "Try Again"

**Program & Sample Output – 9**

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==1):

titleID=raw\_input("Enter titleID? ")

title=raw\_input("Enter title? ")

ISBN=raw\_input("Enter ISBN? ")

publisherID=raw\_input("Enter publisherID? ")

publicationYear=raw\_input("Enter publicationYear? ")

try:

iterator.execute("insert into Titles values (?,?,?,?,?)",(titleID,title,ISBN,publisherID,publicationYear))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==2):

publisherID=raw\_input("Enter publisherID? ")

name=raw\_input("Enter name? ")

streetAdd=raw\_input("Enter street Address? ")

suiteno=raw\_input("Enter Suite Number? ")

zipx=raw\_input("Enter Zip? ")

try:

iterator.execute("insert into publishers values (?,?,?,?,?)",(publisherID,name,streetAdd,suiteno,zipx))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==3):

authorTitID=raw\_input("Enter Author Title ID? ")

authID=raw\_input("Enter Author ID? ")

titleID=raw\_input("Enter Title ID? ")

try:

iterator.execute("insert into AuthorTit values (?,?,?,?,?)",(publisherID,name,streetAdd,suiteno,zipx))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==4):

authID=raw\_input("Enter Author ID? ")

firstName=raw\_input("Enter first Name? ")

middleName=raw\_input("Enter middle Name? ")

lastName=raw\_input("Enter last Name? ")

try:

iterator.execute("insert into authors values (?,?,?,?)",(authID,firstName,middleName,lastName))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==5):

ZipCodeID=raw\_input("Enter Zip Code ID? ")

city=raw\_input("Enter city? ")

state=raw\_input("Enter State? ")

zipCode=raw\_input("Enter Zip Code? ")

try:

iterator.execute("insert into zipCodes values(?,?,?,?", (ZipCodeID,city,state,zipCode))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

def selectFromAllTables(tID):

try:

if(tID==0): iterator.execute('select \* from Books')

elif(tID==1): iterator.execute('select \* from Titles')

elif(tID==2): iterator.execute('select \* from publishers')

elif(tID==3): iterator.execute('select \* from AuthorTit')

elif(tID==4): iterator.execute('select \* from authors')

elif(tID==5): iterator.execute('select \* from zipCodes')

print pd.DataFrame(iterator.fetchall())

except pd.io.sql.DatabaseError , AttributeError:

print "No such records found"

createTables()

while True:

print "1. Insert into Tables "

print "2. Display All Tables "

print "3. Exit"

subchoice=int(raw\_input("Enter choice? "))

if subchoice == 3:

db.close()

break

print "You want to operate on which Table?"

print "1. Books Table"

print "2. Titles Table"

print "3. Publishers Table"

print "4. Author Table"

print "5. Authors Table"

print "6. ZipCodes Table"

choice=int(raw\_input("Enter the table? ")) - 1

if(subchoice==1): insertInto(choice)

elif(subchoice==2): selectFromAllTables(choice)

else: print "Try Again"

iterator.execute("insert into AuthorTit values (?,?,?,?,?)",(publisherID,name,streetAdd,suiteno,zipx))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==4):

authID=raw\_input("Enter Author ID? ")

firstName=raw\_input("Enter first Name? ")

middleName=raw\_input("Enter middle Name? ")

lastName=raw\_input("Enter last Name? ")

try:

iterator.execute("insert into authors values (?,?,?,?)",(authID,firstName,middleName,lastName))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

elif(tID==5):

ZipCodeID=raw\_input("Enter Zip Code ID? ")

city=raw\_input("Enter city? ")

state=raw\_input("Enter State? ")

zipCode=raw\_input("Enter Zip Code? ")

try:

iterator.execute("insert into zipCodes values(?,?,?,?", (ZipCodeID,city,state,zipCode))

choice=raw\_input("Are you sure (Yes/No) ?")

if choice=="Yes" or choice=="YES" or choice=="yes": db.commit()

else: db.rollback()

except sqlite3.IntegrityError:

print "Please enter a UNIQUE Primary key or foriegn Key"

def selectFromAllTables(tID):

try:

if(tID==0): iterator.execute('select \* from Books')

elif(tID==1): iterator.execute('select \* from Titles')

elif(tID==2): iterator.execute('select \* from publishers')

elif(tID==3): iterator.execute('select \* from AuthorTit')

elif(tID==4): iterator.execute('select \* from authors')

elif(tID==5): iterator.execute('select \* from zipCodes')

print pd.DataFrame(iterator.fetchall())

except pd.io.sql.DatabaseError , AttributeError:

print "No such records found"

createTables()

while True:

print "1. Insert into Tables "

print "2. Display All Tables "

print "3. Exit"

subchoice=int(raw\_input("Enter choice? "))

if subchoice == 3:

db.close()

break

print "You want to operate on which Table?"

print "1. Books Table"

print "2. Titles Table"

print "3. Publishers Table"

print "4. Author Table"

print "5. Authors Table"

print "6. ZipCodes Table"

choice=int(raw\_input("Enter the table? ")) - 1

if(subchoice==1): insertInto(choice)

elif(subchoice==2): selectFromAllTables(choice)

else: print "Try Again"

elif(tID==1): iterator.execute('select \* from Titles')

elif(tID==2): iterator.execute('select \* from publishers')

elif(tID==3): iterator.execute('select \* from AuthorTit')

elif(tID==4): iterator.execute('select \* from authors')

elif(tID==5): iterator.execute('select \* from zipCodes')

print pd.DataFrame(iterator.fetchall())

except pd.io.sql.DatabaseError , AttributeError:

print "No such records found"

createTables()

while True:

print "1. Insert into Tables "

print "2. Display All Tables "

print "3. Exit"

subchoice=int(raw\_input("Enter choice? "))

if subchoice == 3:

db.close()

break

print "You want to operate on which Table?"

print "1. Books Table"

print "2. Titles Table"

print "3. Publishers Table"

print "4. Author Table"

print "5. Authors Table"

print "6. ZipCodes Table"

choice=int(raw\_input("Enter the table? ")) - 1

if(subchoice==1): insertInto(choice)

elif(subchoice==2): selectFromAllTables(choice)

else: print "Try Again"

**Additional Question (Optional):**

1. Write a python program to query the metadata of a database
   * + - 1. List all the tables in the database
         2. For each table, list all the column names, description and data types

**Result:**

Thus, the programs were executed successfully.

|  |  |
| --- | --- |
| **Ex. No: 9** | **Title: Databases** |
| **Program (5)** |  |
| **Coding Conventions (2)** |  |
| **Output (3)** |  |
| **Total (10)** |  |
| **Comments:** | **Signature:** |