**Exp 13: Write a program using stack to convert decimal to binary number.**

def divideBy2(decNumber):

remstack = Stack()

while decNumber > 0:

rem = decNumber % 2

remstack.push(rem)

decNumber = decNumber // 2

binString = ""

while not remstack.isEmpty():

binString = binString + str(remstack.pop())

return binString

print(divideBy2(42))

**OUTPUT:**

101010

**Exp 14: Implement basic create, select, delete and update operations on database.**

**CREATE OPERATION:**

import sqlite3

conn = sqlite3.connect('test.db')

print "Opened database successfully";

conn.execute('''CREATE TABLE COMPANY

        (ID INT PRIMARY KEY     NOT NULL,

        NAME      TEXT NOT NULL,

        AGE      INT NOT NULL,

        ADDRESS      CHAR(50),

        SALARY      REAL);''')

print "Table created successfully";

conn.close()

**OUTPUT**

Opened database successfully  
Table created successfully

**INSERT OPERATION:**

import sqlite3  
  
conn = sqlite3.connect('test.db')  
print "Opened database successfully";  
  
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \  
     VALUES (1, 'Paul', 32, 'California', 20000.00 )");  
  
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \  
     VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");  
  
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \  
     VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )");  
  
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \  
     VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )");  
  
conn.commit()  
print "Records created successfully";  
conn.close()

**UPDATE OPERATION:**

import sqlite3  
  
conn = sqlite3.connect('test.db')  
print "Opened database successfully";  
  
conn.execute("UPDATE COMPANY set SALARY = 25000.00 where ID = 1")  
conn.commit  
print "Total number of rows updated :", conn.total\_changes  
  
cursor = conn.execute("SELECT id, name, address, salary from COMPANY")  
for row in cursor:  
  print "ID = ", row[0]  
  print "NAME = ", row[1]  
  print "ADDRESS = ", row[2]  
  print "SALARY = ", row[3], "\n"  
  
print "Operation done successfully";  
conn.close()

**OUTPUT:**

Opened database successfully  
Total number of rows updated : 1  
ID = 1  
NAME = Paul  
ADDRESS = California  
SALARY = 25000.0  
  
ID = 2  
NAME = Allen  
ADDRESS = Texas  
SALARY = 15000.0  
  
ID = 3  
NAME = Teddy  
ADDRESS = Norway  
SALARY = 20000.0  
  
ID = 4  
NAME = Mark  
ADDRESS = Rich-Mond  
SALARY = 65000.0  
  
Operation done successfully

**DELETE OPERATION:**

import sqlite3  
  
conn = sqlite3.connect('test.db')  
print "Opened database successfully";  
  
conn.execute("DELETE from COMPANY where ID = 2;")  
conn.commit()  
print "Total number of rows deleted :", conn.total\_changes  
  
cursor = conn.execute("SELECT id, name, address, salary from COMPANY")  
for row in cursor:  
  print "ID = ", row[0]  
  print "NAME = ", row[1]  
  print "ADDRESS = ", row[2]  
  print "SALARY = ", row[3], "\n"  
  
print "Operation done successfully";  
conn.close()

**OUTPUT:**

Opened database successfully  
Total number of rows deleted : 1  
ID = 1  
NAME = Paul  
ADDRESS = California  
SALARY = 20000.0  
  
ID = 3  
NAME = Teddy  
ADDRESS = Norway  
SALARY = 20000.0  
  
ID = 4  
NAME = Mark  
ADDRESS = Rich-Mond  
SALARY = 65000.0  
  
Operation done successfully

**Exp 15 : Create student registration form.**

import tkinter as tk

def submit():

print("Your form is submitted successfully!")

root=tk.Tk();

v=tk.IntVar();

root.config(bg='light blue')

name=tk.Label(root,text="Name:").grid(row=0)

last=tk.Label(root,text="Last name:").grid(row=2)

ety\_name=tk.Entry(root).grid(row=0,column=1)

ety\_last=tk.Entry(root).grid(row=2,column=1)

gender=tk.Label(root,text="Gender:").grid(row=4)

rd1=tk.Radiobutton(root,text="Male",value=1).grid(row=4,column=1)

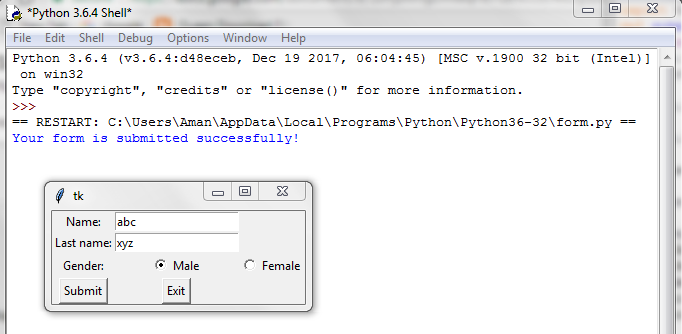
rd2=tk.Radiobutton(root,text="Female",value=2).grid(row=4,column=2)

bttn\_submit=tk.Button(root,text="Submit",command=submit).grid(row=5,column=0)

bttn\_quit=tk.Button(root,text="Exit",command=quit).grid(row=5,column=1)

root.mainloop();

**OUTPUT:**

****

**Exp 16 : Write a socket program to exchange information between server and client.**

**# TCP Server**

# ==========

import socket

# Creatin a TCP/IP socket

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

# Binding the socket to the port

serveraddress = ('localhost',12345)

print("Starting the server on %s port %s" % serveraddress)

serversock.bind(serveraddress)

# Listening to the incoming connections

serversock.listen(1)

while True:

# Waiting for a new connection

print("Waiting for a connection from a client ... ")

conn,clientaddress = serversock.accept()

try:

# Receiving the data fom the client and sending it again

print('Received Connection from',clientaddress)

while True:

data = conn.recv(1024)

print("Received data :", data.decode())

if data:

print("Sending data back to the client.")

conn.sendall(data)

else:

print("There is no more data.", clientaddress)

print("---------------")

break

finally:

# Closing the connection

conn.close()

**TCPClient.py**

## TCP Client

## ==========

import socket

# Creating a TCP/ IP socket

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

# Connecting the socket to the server's port

serveraddress = ("localhost",12345)

print("Connecting to the port %s port %s of server" % serveraddress)

clientsock.connect(serveraddress)

try:

# Sending data

senddata = ("This is Socket Programming in Python.")

print("Sending the message.... %s" % senddata)

clientsock.sendall(senddata.encode('utf-8'))

# Receiving data

recvdata = clientsock.recv(1024)

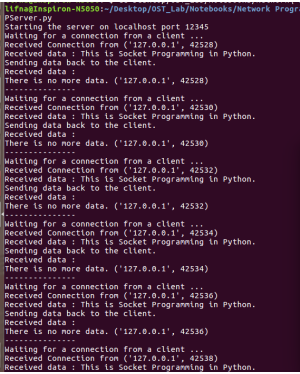
print("Received data : %s" % recvdata.decode())

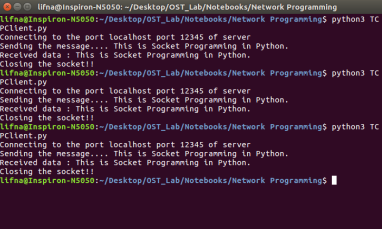
finally:

# Closing the connection

print("Closing the socket!!")

clientsock.close()





**Exp 17 : To implement arrays and their in-built functions using Perl and find out a factorial of a number.**

**factorial.pl**

#!usr/bin/perl

print "Enter the number : ";

my $num = <>;

my $i;

my $fact = 1;

if($num >= 1){

for($i = 1; $i <= $num; $i++){

$fact \*= $i;

}

print "Factorial = $fact \n";

}

else{

print "Factorial is undefined \n";

}

**OUTPUT:**

$ perl factorial.pl

Enter the number : 4

Factorial = 24

$ perl factorial.pl

Enter the number : 0

Factorial is undefined

$ perl factorial.pl

Enter the number : -1

Factorial is undefined

**Demonstrate use of array:**

**array\_ref.pl**

@arr1 = (1, 2, 3);

@arr2 = (4, 5, 6);

check\_size(@arr1, @arr2);

sub check\_size {

my (@a1, @a2) = @\_;

print “Result from check\_size : ”;

print @a1 == @a2 ? 'Yes' : 'No';

}

check\_size2(\@arr1, \@arr2);

sub check\_size2 {

my ($a1, $a2) = @\_;

print “Result from check\_size2 : ”;

print @$a1 == @$a2 ? 'Yes' : 'No';

}

**OUTPUT:**

$array\_ref.pl

Result from check\_size : No

Result from check\_size2 : Yes

**Exp 18 : Write a perl script to create a math package and demonstrate use of modules.**

**math\_pkg.pm**

#!usr/bin/perl

package math\_pkg;

sub add{

my ($a,$b) = @\_;

return $a+$b;

}

sub diff{

my ($a,$b) = @\_;

return $a-$b;

}1;

**use\_math\_pkg.pl**

#!usr/bin/perl

use math\_pkg;

print "Addition of two numbers\n";

print "First number : ";

my $a = <>;

print "Second number : ";

my $b = <>;

my $sum = math\_pkg::add($a,$b);

print "Sum = " . $sum . "\n";

print;

print "Difference of two numbers\n";

print "First number : ";

my $a = <>;

print "Second number : ";

my $b = <>;

my $sub = math\_pkg::diff($a,$b);

print "Difference = " . $sub . "\n";

print "Thank you\n";

**OUTPUT:**

$perl use\_math\_pkg.pl

Addition of two numbers

First number : 23

Second number : 34

Sum = 57

Difference of two numbers

First number : 23

Second number : 34

Difference = -11

Thank you

**Exp 19 : To create a class Person and create its objects using Perl.**

**Person.pm**

use strict;

use warnings;

package Person;

use Exporter;

our @ISA = ('Exporter');

our @EXPORT = ('get\_name','set\_name','new');

sub new {

   my $class = shift;

   my $name = shift;

   my $self = { name => $name };

    bless $self,$class;

    return $self;

}

sub get\_name {

   my $self = shift;

   return $self->{name};

}

sub set\_name {

   my $self = shift;

   $self->{name} = shift;

}

1;

**use\_Person.pl**

use Person;

my $obj = new Person('Dave');

print "Name : " . $obj->get\_name . "\n";

$obj->set\_name('David');

print "New Name : " . $obj->get\_name . "\n";

**Output**

$ perl use\_Person.pl

Name : Dave

New Name : David

**Exp 20 : WAP to demonstrate file handling (Create, Read)**

**file\_read.pl**

# open opens a file and associates it with filehandle

open(my $file, '<', 'in.txt');

#You can then read the file with <$file>

@lines = <$file>; # all lines

print @lines;

# Finally, close the file with close

close($file);

**in.txt**

Hello friends!!

Welcome to Perl Programming.

It's lots of fun.

Let’s explore…

**Output**

$ perl file\_read.pl

Hello friends!!

Welcome to Perl Programming.

It's lots of fun.

Let’s explore...

**file\_write.pl**

open my $file, '>>', 'in.txt';

print $file "First let's write a hello world program...\n";

close($file);

**Output:**

$ perl file\_write.pl

**in.txt**

Hello friends!!

Welcome to Perl Programming.

It's lots of fun.

Let’s explore…

First let's write a hello world program...