PROGRAM:

```
user = str
end = "0"
hours = round(40,2)
print("One Stop Shop Payroll Calculator")

hours = (float(input("Please enter hours worked: ", )))
payrate =(float(input("Please enter your payrate: $", )))
if hours < 40:
    print("Employee's name: ", user)
    print("Overtime hours: 0")
    print("Overtime Pay: $0.00")
    regularpay = round(hours * payrate, 2)
    print("Gross Pay: $", regularpay)
elif hours > 40:
    overtimehours = round(hours - 40.00,2)
    print("Overtime hours: ", overtimehours)
    print("Employee's name: ", user)
    regularpay = round(hours * payrate,2)
    overtimerate = round(payrate * 1.5, 2)
    overtimepay = round(overtimehours * overtimerate)
    grosspay = round(regularpay+overtimepay,2)
    print("Regular Pay: $", regularpay)
    print("Overtime Pay: $",overtimepay)
    print("Gross Pay: $", grosspay)

while user != end:
    print()
    user = input("Please enter your name or type '0' to quit: ")
if user == end:
   print("End of Report")
```

PROGRAM :

```
# Program to Calculate compound interest
principle=1000
rate=10.25
time=5
Amount = principle * (pow((1 + rate / 100), time))
CI = Amount - principle
print("Compound interest is", CI)
```

PROGRAM :

```
num = 76542
reverse_number = 0
print("Given Number ", num)
while num > 0:
    reminder = num % 10
    reverse_number = (reverse_number * 10) + reminder
    num = num // 10
 print("Revered Number ", reverse_number)
```

PROGRAM :

```
input_number = 6
for i in range(1, input_number + 1):
    print("Current Number is :", i, " and the cube is", (i * i * i))
```

PROGRAM :

```
number_of_terms = 5
start = 2
sum = 0
for i in range(0, number_of_terms):
    print(start, end=" ")
    sum += start
    start = (start * 10) + 2
print("\nSum of above series is:", sum)
```

PROGRAM:

```
X = [[12,7,3],
        [4 ,5,6],
       [7 ,8,9]]
Y = [[5,8,1],
        [6,7,3],
       [4,5,9]]
result = [[0,0,0],
              [0,0,0],
              [0,0,0]]
# iterate through rows
for i in range(len(X)):
   # iterate through columns
   for j in range(len(X[0])):
       result[i][j] = X[i][j] + Y[i][j]
for r in result:
   print(r)
```

PROGRAM :

```
# Taking kilometers input from the user
kilometers = float(input("Enter value in kilometers: "))
conv_fac = 0.621371
# calculate miles
miles = kilometers * conv_fac
print('%0.2f kilometers is equal to %0.2f miles' %(kilometers,miles))
```

PROGRAM :

```
import random
def Rand(start, end, num):
    res = []

    for j in range(num):
        res.append(random.randint(start, end))

    return res

# Driver Code
num = 10
start = 20
end = 40
print(type(Rand))
print(Rand(start, end, num))
```

PROGRAM :

```
def fib_intervall(x):
    if x < 0:
        return -1
    (old,new) = (0,1)
    while True:
        if new < x:
            (old,new) = (new,old+new)
        else:
            if new == x:
                new = old+new
            return (old, new)
while True:
    x = int(input("Your number: "))
    if x <= 0:
        break
    (lub, sup) = fib_intervall(x)
    print("Largest Fibonacci Number smaller than x: " + str(lub))
    print("Smallest Fibonacci Number larger than x: " + str(sup))
```

PROGRAM:

```
def make_bold(fn):
    def wrapped():
        return "<b>" + fn() + "</b>"
    return wrapped
def make_italic(fn):
    def wrapped():
        return "<i>" + fn() + "</i>"
    return wrapped
def make_underline(fn):
    def wrapped():
        return "<u>" + fn() + "</u>"
    return wrapped
@make_bold
@make_italic
@make_underline
def hello():
    return "hello world"
print(hello()) ## returns "<b><i><u>hello world</u></i></b>"
```

PROGRAM:

```
def test(a):
    def add(b):
        nonlocal a
        a += 1
        return a+b
    return add
func= test(4)
print(func(4))
```

PROGRAM:

```
def unique_list(l):
  x = []
  for a in l:
    if a not in x:
      x.append(a)
  return x

print(unique_list([1,2,3,3,3,3,4,5]))
```

PROGRAM:

```
import string, sys
def ispangram(str1, alphabet=string.ascii_lowercase):
    alphaset = set(alphabet)
    return alphaset<= set(str1.lower())
print ( ispangram('The quick brown fox jumps over the lazy dog'))
```

PROGRAM:

```
class BankAccount:
    def __init__(self):
        self.balance = 0
    def withdraw(self, amount):
        self.balance -= amount
        return self.balance
    def deposit(self, amount):
        self.balance += amount
        return self.balance
a = BankAccount()
b = BankAccount()
print(a.deposit(100))
print(b.deposit(50))
print(b.withdraw(10))
print(a.withdraw(10))
```

PROGRAM :

```
class Employee:
    def __init__(self): #Constructor
        self.__id = 0
        self.__name = ""
        self.__gender = ""
        self.__city = ""
        self.__salary = 0
        print("Object Initialized.")
    def __del__(self): #Destructor
        print("Object Destroyed.")
    def setData(self):
        self.__id=int(input("Enter Id\t:"))
        self.__name = input("Enter Name\t:")
        self.__gender = input("Enter Gender:")
        self.__city = input("Enter City\t:")
        self.__salary = int(input("Enter Salary:"))
    def __str__(self):
        data =
"["+str(self.__id)+","+self.__name+","+self.__gender+","+self.__city+","+str(self.__salary)+"]"
        return data
    def showData(self):
        print("Id\t\t:",self.__id)
        print("Name\t:", self.__name)
        print("Gender\t:", self.__gender)
        print("City\t:", self.__city)
        print("Salary\t:", self.__salary)


def main():
    #Employee Object
    emp=Employee()
    emp.setData()
    emp.showData()
    print(emp)

if __name__=="__main__":
    main()
```

PROGRAM :

```
class Student:
    def __init__(self, name, id, age):
        self.name = name
        self.id = id
        self.age = age
# creates the object of the class Student
s = Student("John", 101, 22)
# prints the attribute name of the objects
print(getattr(s, 'name'))
# reset the value of attribute age to 23
setattr(s, "age", 23)
# prints the modified value of age
print(getattr(s, 'age'))
print(hasattr(s, 'id'))
# deletes the attribute age
delattr(s, 'age')
```

PROGRAM :

```
class calc:
    def getDetail(self):
        self.total_computer=258
        self.total_hour=6
    def calculatesecondsperDay(self):
        Second_per_Day=self.total_hour*60*60
        print('Total Seconds per Day:',Second_per_Day)
    def calculateminutesperWeek(self):
        Minutes_per_Week=self.total_hour*60*7
        print("Total Minutes per Week:",Minutes_per_Week)
    def calculatehourperMonth(self):
        Hour_per_Month=self.total_hour*30
        print("Total Hour per Month:",Hour_per_Month)
    def calculatedayperyear(self):
        Day_per_Year=(self.total_hour*365)/24
        print("Total Day per Year:",Day_per_Year)
to=calc()
to.getDetail()
to.calculatesecondsperDay()
to.calculateminutesperWeek()
to.calculatehourperMonth()
to.calculatedayperyear()
```

Program:

```
import tkinter as tk
from tkinter import *
root=tk.Tk()
def rightclick(event):
    print("rightclick")

def leftclick(event):
    print("leftclick")

def middleclick(event):
    print("middleclick")
frame=Frame(root,width=300,height=200)
frame.bind('<Button-1>',leftclick)
frame.bind('<Button-2>',middleclick)
frame.bind('<Button-3>',rightclick)
frame.pack()
root.mainloop()
```

PROGRAM :

```
from tkinter import *
import tkinter as tk
class App(tk.Tk):
    def __init__(self):
        super().__init__()
        frame = tk.Frame(self, bg="green",  height=100, width=100)
        frame.bind("<Button-1>", self.print_event)
        frame.bind("<Double-Button-1>", self.print_event)
        frame.bind("<ButtonRelease-1>", self.print_event)
        frame.bind("<B1-Motion>", self.print_event)
        frame.bind("<Enter>", self.print_event)
        frame.bind("<Leave>", self.print_event)
        frame.pack(padx=50, pady=50)

    def print_event(self, event):
        position = "(x={}, y={})".format(event.x, event.y)
        print(event.type, "event", position)

if __name__ == "__main__":
    app = App()
app.mainloop()
```

PROGRAM :

```
from tkinter import *
import tkinter as tk
import tkinter as event

root = Tk()
def key(event):
    print("pressed", repr(event.char))
def callback(event):
    frame.focus_set()
    print("clicked at", event.x, event.y)
frame = Frame(root, width=100, height=100)
frame.bind("<Key>", key)
frame.bind("<Button-1>", callback)
frame.pack()
root.mainloop()
```

PROGRAM :

```
import tkinter as tk
class App(tk.Tk):
    def __init__(self):
        super().__init__()
        entry = tk.Entry(self)
        entry.bind("<FocusIn>", self.print_type)
        entry.bind("<Key>", self.print_key)
        entry.pack(padx=20, pady=20)

    def print_type(self, event):
        print(event.type)

    def print_key(self, event):
        args = event.keysym, event.keycode, event.char
        print("Symbol: {}, Code: {}, Char: {}".format(*args))
if __name__ == "__main__":
    app = App()
app.mainloop()
```

PROGRAM:

```
#import the required packages
import sqlite3

#create a connection
con = sqlite3.connect('Students.db')

#create a cursor object
c=con.cursor()

#Create a table:
c.execute('''CREATE TABLE student(roll_no INTEGER,name TEXT,age INTEGER);''')

#now to insert data:
c.execute('''INSERT INTO student VALUES(49,'Aman Bhai Patel',19)''')

#commit the changes to the database
con.commit()

#to see the data
for row in c.execute('''SELECT * FROM student'''):
    print(row)
```

PROGRAM:

```
# Import required packages
import sqlite3

# create a connection and cursor
con = sqlite3.connect("file3.db")
c = con.cursor()

# create the 2 tables and insert some random data
c.execute('''CREATE TABLE t1(id INTEGER,name TEXT);''')
c.execute('''CREATE TABLE t2(id INTEGER,job TEXT);''')

c.execute('''INSERT INTO t1 VALUES(1,'Aman'),(2,'Aviraj'),(1,'Nithish'),(2,'Venkat');''')
c.execute('''INSERT INTO t2 VALUES(1,'Job1'),(2,'Job2');''')

# commit changes
con.commit()

# send the select command to the sqlite3 backend:
task = '''SELECT t1.name,t2.job FROM t1,t2 WHERE t1.id = t2.id;'''
for row in c.execute(task):
    print(row)
```

PROGRAM:

```
# import required packages
import sqlite3

# Create a connection anf cursor
con = sqlite3.connect('file4.db')
c = con.cursor()

# create a function to print the whole table:
def printall():
    global c
    for row in c.execute("SELECT * FROM datatable"):
        print(row)


# Create a sample table and insert data into it:
c.execute("CREATE TABLE datatable(ID INTEGER,Name TEXT);")

# INSERT
namelist = [(1,'Aman'),(2,'Aviraj'),(3,'Venkat')]
c.executemany("INSERT INTO datatable VALUES(?,?)",namelist)
con.commit()

print("\nInitial table: ")
printall()

# DELETE
print("\nDeleting an enrty : (3,Venkat)")
c.execute("DELETE FROM datatable WHERE Name = 'Venkat';")
print("\nTable is now: ")
printall()

# UPDATE
print("\nUpdating a name to full name:")
print("\nTable is now: ")
c.execute("UPDATE datatable SET Name='AMAN BHAI PATEL' WHERE ID=1;")
printall()
```

PROGRAM:

```
import sqlite3
# connect to a database
conn = sqlite3.connect('test.db')

print ("Opened database successfully");
# To Create a table
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()

# To insert records into a table
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()

# To display the data from the table
import sqlite3

conn = sqlite3.connect('test.db')
print("Opened database successfully")

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()

# To display all columns from a database
import sqlite3
conn = sqlite3.connect('test.db')
print("Opened database successfully")
conn.execute('''CREATE TABLE COMPANY12345
        (ID INT PRIMARY KEY     NOT NULL,
        NAME           TEXT    NOT NULL,
        AGE            INT     NOT NULL,
        ADDRESS        CHAR(50),
        SALARY         REAL);''')
print("Table created successfully")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY)\
     VALUES (1, 'Paul', 32, 'California', 20000.00 )")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY) \
     VALUES (2, 'Allen', 25, 'Texas', 15000.00 )")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY) \
     VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY) \
     VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )")

conn.commit()
print("Records created successfully")
cursor = conn.execute("SELECT ID,NAME,AGE,ADDRESS,SALARY from COMPANY12345")
for row in cursor:
   print("ID = ", row[0])
   print("NAME = ", row[1])
   print("AGE=", row[2])
   print("ADDRESS = ", row[3])
   print("SALARY = ", row[4])

print("Operation done successfully");
conn.close()

# To update the table
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()

# To perform 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()
```

PROGRAM :

```
my_list = [1, 2, 3, 4, 5]
sum = 0
for x in my_list:
    sum += x
print(sum)
```

PROGRAM :

```
sample_characters = ["p","y","t","h","o","n"]
sample_string = ''
sample_string
sample_string = sample_string + sample_characters[0]
sample_string ="p"
sample_string = sample_string + sample_characters[1]
sample_string ="py"
sample_string = sample_string + sample_characters[2]
sample_string ="pyt"
sample_string = sample_string + sample_characters[3]
sample_string  ="pyth"
sample_string = sample_string + sample_characters[4]
sample_string ="pytho"
sample_string = sample_string + sample_characters[5]
sample_string  ="python"
print(sample_string)
```

PROGRAM :

```
sample_characters = ["w","e","l","c","o","m","e"]
sample_string = ''
sample_string
for c in sample_characters:
    sample_string = sample_string + c
    print(sample_string)
```

PROGRAM:

```
for num in range(10,20):
   for i in range(2,num):
      if num%i == 0:
         j=num/i
         print(num,i,j)
   else:
      print(num, "is a prime number")
```

PROGRAM :

```
from multiprocessing import Pool

def f(x):
    return x*x

if __name__ == '__main__':
    with Pool(5) as p:
        print(p.map(f, [1, 2, 3]))
```

PROGRAM :

```
from multiprocessing import Process
import os

def info(title):
    print(title)
    print('module name:', __name__)
    print('parent process:', os.getppid())
    print('process id:', os.getpid())

def f(name):
    info('function f')
    print('hello', name)

if __name__ == '__main__':
    info('main line')
    p = Process(target=f, args=('bob',))
    p.start()
    p.join()
```

PROGRAM :

```
from multiprocessing import Process, Lock

def f(l, i):
    l.acquire()
    try:
        print('hello world', i)
    finally:
        l.release()

if __name__ == '__main__':
    lock = Lock()

    for num in range(10):
        Process(target=f, args=(lock, num)).start()
```

PROGRAM :

```
from multiprocessing import Process, Value, Array

def f(n, a):
    n.value = 3.1415927
    for i in range(len(a)):
        a[i] = -a[i]
if __name__ == '__main__':
    num = Value('d', 0.0)
    arr = Array('i', range(10))
    p = Process(target=f, args=(num, arr))
    p.start()
    p.join()
    print(num.value)
    print(arr[:])
```

PROGRAM :

```
import threading
x = 0     # A shared value
COUNT = 10
def incr():
    global x
    for i in range(COUNT):
        x += 1
        print(x)
def decr():
    global x
    for i in range(COUNT):
        x -= 1
        print(x)
t1 = threading.Thread(target=incr)
t2 = threading.Thread(target=decr)
t1.start()
t2.start()
t1.join()
t2.join()
print(x)
```

PROGRAM :

```
import threading
x = 0     # A shared value
COUNT = 10
lock = threading.Lock()
def incr():
    global x
    lock.acquire()
    print("thread locked for increment cur x=",x)
    for i in range(COUNT):
        x += 1
        print(x)
    lock.release()
    print("thread release from increment cur x=",x)
def decr():
    global x
    lock.acquire()
    print("thread locked for decrement cur x=",x)
    for i in range(COUNT):
        x -= 1
        print(x)
    lock.release()
    print("thread release from decrement cur x=",x)
t1 = threading.Thread(target=incr)
t2 = threading.Thread(target=decr)
t1.start()
t2.start()
t1.join()
t2.join()
```