Experiment No. 1

Aim: Calculate factorial of a number inputted by user.

**Code**

def factorial(n):

    if n == 0:

        return 1

    else:

        return n \* factorial(n-1)

n = int(input("Enter a number: "))

print(factorial(n))

**Output**

#Run 1

Enter a number: 0

1

#Run 2

Enter a number: 69

171122452428141311372468338881272839092270544893520369393648040923257279754140647424000000000000000

#Run 3

Enter a number: 420

1179832395293178259148587778443982767423908163629667689799210969550884231351169347804766799500510294050388349696532084729374087533384204019322892961178819464698121263533012685335273004294789382652477324465427001701326230145911466316029644714371748823861128004214806081770714277374544632880180009063325310867611466814559562175609414340177417478580290981292661586700768075544788360242053436899439186009859147147653878644064667799709427693731208035920284052203131022083688425805265631534978481761954009800546844281261649619610291306374918025956972209823833523561696079181976208783662818235613615149296343931089295234402130043253489826928097199211074340929916161625854705227595565090740962113793308742649598603963747960941063835474664306971892700806057422478626083960243385932102946293048920279760860198799159782580284293120000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

Experiment No. 2

Aim: Check whether the number inputted by user is a prime number or not.

**Code**

def check\_prime():

    num = int(input("Enter a number: "))

    if num > 1:

        for i in range(2, num):

            if (num % i) == 0:

                print(num, "is not a prime number")

                break

        else:

            print(num, "is a prime number")

    else:

        print(num, "is not a prime number")

check\_prime()

**Output**

#Run 1

Enter a number: 0

0 is not a prime number

#Run 2

Enter a number: 1

1 is not a prime number

#Run 3

Enter a number: 5

5 is a prime number

#Run 4

Enter a number: 15

15 is not a prime number

Experiment No. 3

Aim: Search for the existence of a substring within an inputted string.

**Code**

def search\_in\_string():

    s = input("Enter a string: ")

    t = input("Enter a substring to search in string: ")

    if t in s:

        print("Substring found")

        print("The number of times the substring is present in the string is", s.count(t))

    else:

        print("Substring not found")

search\_in\_string()

**Output**

#Run 1

Enter a string: Hello World

Enter a substring to search in string: hello

Substring not found

#Run 2

Enter a string: Hello World

Enter a substring to search in string: Hello

Substring found

The number of times the substring is present in the string is 1

Experiment No. 4

Aim: Perform Bubble Sort using a user-defined function.

**Code**

def BubbleSort():

    L = [eval(i) for i in input("Enter the list items : ").split()]

    for i in range(len(L)):

        for j in range(len(L)-1):

            if L[j] > L[j+1]:

                L[j], L[j+1] = L[j+1], L[j]

    return L

print(BubbleSort())

**Output**

#Run 1

Enter the list items : 10 5 7 8 2 4 6 1 3 9

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

#Run 2

Enter the list items : 'Shehzad' 'Abdullahir' 'Taha' 'Farhan' 'Abhinav'

['Abdullahir', 'Abhinav', 'Farhan', 'Shehzad', 'Taha']

Experiment No. 5

Aim: Display file content line by line, with each word separated by '#".

**Code**

def split():

    f = open("data.txt", "r")

    for line in f.readlines():

        print(line.replace(' ', '#'), end='')

    f.close()

split()

**File Content**

Hello World

This is a new sentence

**Output**

Hello#World

This#is#a#new#sentence

Experiment No. 6

Aim: Count and display the number of vowels, consonants, lowercase and uppercase characters in a file.

**Code**

def count\_vowels():

    vowels = 0

    for line in L:

        for char in line:

            if char.lower() in 'aeiou':

                vowels += 1

    print("The number of vowels in the file is", vowels)

def count\_consonants():

    consonants = 0

    for line in L:

        for char in line:

            if char.isalpha() and char.lower() not in 'aeiou':

                consonants += 1

    print("The number of consonants in the file is", consonants)

def count\_lowercase():

    lowercase = 0

    for line in L:

        for char in line:

            if char.islower():

                lowercase += 1

    print("The number of lowercase letters in the file is", lowercase)

def count\_uppercase():

    uppercase = 0

    for line in L:

        for char in line:

            if char.isupper():

                uppercase += 1

    print("The number of uppercase letters in the file is", uppercase)

f = open("data.txt", "r")

L = f.readlines()

f.close()

count\_vowels()

count\_consonants()

count\_lowercase()

count\_uppercase()

**File Content**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent in placerat tortor, vitae hendrerit mauris. Vestibulum tempor non enim vitae tempor. Quisque nec varius dui, ut dapibus elit. Sed nec gravida libero. Integer euismod purus quis odio sagittis elementum. Mauris id placerat odio, sed mattis nulla. Duis eleifend eget ligula eu rhoncus. Mauris facilisis euismod quam vel rhoncus.

**Output**

The number of vowels in the file is 142

The number of consonants in the file is 179

The number of lowercase letters in the file is 312

The number of uppercase letters in the file is 9

Experiment No. 7

Aim: Create a binary file to store Roll Number and Name and perform search operations on it.

**Code**

import pickle

L = []

def add\_student():

    f = open("student.dat", "wb+")

    while True:

        roll = int(input("Enter roll number: "))

        name = input("Enter name: ")

        L.append([roll, name])

        if input("Would you like to add another student? (y/n): ") != "y":

            break

        else:

            continue

    print()

    pickle.dump(L, f)

    f.close()

def search\_student():

    roll = int(input("Enter roll number to search: "))

    f = open("student.dat", "rb")

    c = pickle.load(f)

    for i in c:

        if roll == i[0]:

            print("Roll number:", i[0], "Name:", i[1], '\n')

            break

    else:

        print("Roll number not found")

    f.close()

while True:

    print("Menu")

    print("1. Add student\n2. Search student\n3. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:

        add\_student()

    elif choice == 2:

        search\_student()

    elif choice == 3:

        exit()

**Output**

Menu

1. Add student

2. Search student

3. Exit

Enter your choice: 1

Enter roll number: 001

Enter name: Taha

Would you like to add another student? (y/n): y

Enter roll number: 102

Enter name: Shehzad

Would you like to add another student? (y/n): n

Menu

1. Add student

2. Search student

3. Exit

Enter your choice: 2

Enter roll number to search: 102

Roll number: 102 Name: Shehzad

Menu

1. Add student

2. Search student

3. Exit

Enter your choice: 2

Enter roll number to search: 103

Roll number not found

Menu

1. Add student

2. Search student

3. Exit

Enter your choice: 3

Experiment No. 8

Aim: Create a binary file to store Roll Number and Name of student, and to perform search operations on it.

**Code**

import pickle

L = []

def add\_student():

    f = open("student.dat", "wb+")

    while True:

        roll = int(input("Enter roll number: "))

        name = input("Enter name: ")

        marks = int(input("Enter marks: "))

        L.append([roll, name, marks])

        if input("Would you like to add another student? (y/n): ") != "y":

            break

        else:

            continue

    print()

    pickle.dump(L, f)

    f.close()

def search\_student():

    roll = int(input("Enter roll number to search: "))

    f = open("student.dat", "rb")

    c = pickle.load(f)

    for i in c:

        if roll == i[0]:

            print("Roll number:", i[0], "Name:", i[1], "Marks: ", i[2], '\n')

            break

    else:

        print("Roll number not found")

    f.close()

def edit\_student():

    roll = int(input("Enter roll number to edit: "))

    mark = int(input("Enter new marks: "))

    f = open("student.dat", "rb")

    c = pickle.load(f)

    f.close()

    for i in c:

        if roll == i[0]:

            i[2] = mark

            f = open("student.dat", "wb")

            pickle.dump(c, f)

            f.close()

            break

    else:

        print("Roll number not found")

    print("Marks updated")

while True:

    print("Menu")

    print("1. Add student\n2. Search student\n3. Edit student\n4. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:

        add\_student()

    elif choice == 2:

        search\_student()

    elif choice == 3:

        edit\_student()

    elif choice == 4:

        exit()

**Output**

Menu

1. Add student

2. Search student

3. Edit student

4. Exit

Enter your choice: 1

Enter roll number: 001

Enter name: Taha

Enter marks: 99

Would you like to add another student? (y/n): y

Enter roll number: 102

Enter name: Shehzad

Enter marks: 95

Would you like to add another student? (y/n): n

Menu

1. Add student

2. Search student

3. Edit student

4. Exit

Enter your choice: 2

Enter roll number to search: 001

Roll number: 1 Name: Taha Marks: 99

Menu

1. Add student

2. Search student

3. Edit student

4. Exit

Enter your choice: 3

Enter roll number to edit: 102

Enter new marks: 99

Marks updated

Menu

1. Add student

2. Search student

3. Edit student

4. Exit

Enter your choice: 2

Enter roll number to search: 102

Roll number: 102 Name: Shehzad Marks: 99

Menu

1. Add student

2. Search student

3. Edit student

4. Exit

Enter your choice: 4

Experiment No. 9

Aim: Copy a file into a new file except for lines containing the letter 'a'.

**Code**

f1 = open("file1.txt", "r")

f2 = open("file2.txt", "w")

c = f1.readlines()

L = []

for i in c:

    if 'a' not in i.lower():

        L.append(i)

f2.writelines(L)

f1.close()

f2.close()

**File Content**

**File1.txt**

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Maecenas sed diam efficitur, pretium nulla eu, fermentum purus.

Nullam lobortis est a fermentum hendrerit.

Quisque efficitur, purus at consequat iaculis, metus purus euismod quam, at imperdiet felis metus at dolor.

Nulla porta tellus urna, non tincidunt diam bibendum at.

Quisque ac risus condimentum, suscipit leo aliquet, efficitur sapien.

Sed eleifend orci ac est placerat tempus.

Phasellus accumsan ullamcorper elementum.

Quisque sed ante mattis lacus porttitor interdum.

Nunc id magna nec dolor condimentum euismod.

Hello World

**File2.txt**

Hello World

Experiment No. 10

Aim: Create a CSV file and store employee ID, name and salary. Perform search operations.

**Code**

import csv

L = []

f = open('data.csv', 'w', newline='')

writer = csv.writer(f, delimiter=',')

writer.writerow(['Employee ID', 'Employee Name', 'Employee Salary'])

f.close()

def create():

    f = open('data.csv', 'a+', newline='')

    writer = csv.writer(f, delimiter=',')

    while True:

        idno = input("Enter Employee ID: ")

        name = input("Enter Employee Name: ")

        salr = input("Enter Employee Salary: ")

        L.append([idno, name, salr])

        if input("Do you want to continue? (y/n): ").lower() != 'y':

            break

    writer.writerows(L)

    f.close()

def search():

    f = open('data.csv', 'r')

    reader = csv.reader(f, delimiter=',')

    idno = input("Enter Employee ID: ")

    for row in reader:

        if idno == row[0]:

            print("Employee ID: ", row[0], "\nEmployee Name: ", row[1], "\nEmployee Salary: ", row[2])

            break

    f.close()

while True:

    print("Menu")

    print("1. Create\n2. Search\n3. Exit")

    ch = int(input("Enter your choice: "))

    if ch == 1:

        create()

    elif ch == 2:

        search()

    elif ch == 3:

        exit()

    else:

        print("Invalid choice")

**Output**

Menu

1. Create

2. Search

3. Exit

Enter your choice: 1

Enter Employee ID: 101

Enter Employee Name: Bob

Enter Employee Salary: 1000

Do you want to continue? (y/n): y

Enter Employee ID: 102

Enter Employee Name: Thomas

Enter Employee Salary: 1100

Do you want to continue? (y/n): n

Menu

1. Create

2. Search

3. Exit

Enter your choice: 2

Enter Employee ID: 101

Employee ID: 101

Employee Name: Bob

Employee Salary: 1000

Menu

1. Create

2. Search

3. Exit

Enter your choice: 2

Enter Employee ID: 103

Menu

1. Create

2. Search

3. Exit

Enter your choice: 3

Experiment No. 11

Aim: Simulate a dice.

**Code**

import random

import time

def simulate\_dice():

    try:

        print("Press Ctrl+C to stop the dice")

        while True:

            n = random.randint(1, 6)

            print(n, end=' ')  # This statement can be commented out to keep the terminal clean

            time.sleep(0.001)

    except KeyboardInterrupt:

        print("\nDice stopped")

        print("Your number is", n)

simulate\_dice()

**Output**

Press Ctrl+C to stop the dice

2 5 3 1 1 6 4 5 5 5 3 1 3 1 6 4 3 5 4 2 2 3 1 4 4 3 5 1 2 4 4 2 6 1 6 1 6 3 3 5 3 4 2 3 2 3 4 6 1 2 2 6 1 1 3 5 5 2 1 2 6 5 2 3 5 5 5 2 4 2 3 3 6 2 2 2 4 3 1 4 3 2 2 1 4 3 3 2 2 5 6 4 6 4 4 6 4 1 6 5 2 4 6 1 1 5 5 6 1 3 6 5 1 5 2 1 4 5 6 5 3 6 2 3 1 5 6 1 2 5 2 1 2 3 3

Dice stopped

Your number is 3

Experiment No. 12

Aim: Implement a stack in Python using lists.

**Code**

S = []

def isEmpty(S):

    return True if len(S) == 0 else False

def push(S, x):

    S.append(x)

    top = len(S) - 1

def pop(S):

    return "Underflow" if isEmpty(S) else S.pop()

def peek(S):

    return "Underflow" if isEmpty(S) else S[-1]

def show(S):

    print("No elements in the stack") if isEmpty(S) else None

    t = len(S) - 1

    while t >= 0:

        print(S[t], end=' ')

        t -= 1

    print()

while True:

    print("Menu")

    print("1. Push\n2. Pop\n3. Peek\n4. Show\n5. Exit")

    c = int(input("Enter your choice: "))

    if c == 1:

        x = input("Enter the element to be pushed: ")

        push(S, x)

    elif c == 2:

        print(pop(S))

    elif c == 3:

        print(peek(S))

    elif c == 4:

        show(S)

    elif c == 5:

        exit()

    else:

        print("Invalid choice")

**Output**

Menu

1. Push

2. Pop

3. Peek

4. Show

5. Exit

Enter your choice: 1

Enter the element to be pushed: 1

Menu

1. Push

2. Pop

3. Peek

4. Show

5. Exit

Enter your choice: 1

Enter the element to be pushed: 3

Menu

1. Push

2. Pop

3. Peek

4. Show

5. Exit

Enter your choice: 1

Enter the element to be pushed: 5

Menu

1. Push

2. Pop

3. Peek

4. Show

5. Exit

Enter your choice: 2

5

Menu

1. Push

2. Pop

3. Peek

4. Show

5. Exit

Enter your choice: 3

3

Menu

1. Push

2. Pop

3. Peek

4. Show

5. Exit

Enter your choice: 4

3 1

Menu

1. Push

2. Pop

3. Peek

4. Show

5. Exit

Enter your choice: 5

Experiment No. 13

Aim: Linear and Binary Search using lists.

**Code**

def LinearSearch():

    L = [eval(i) for i in input("Enter the list items : ").split()]

    c = eval(input("Enter element to search : "))

    for i in range(len(L)):

        if L[i] == c:

            print("Element found at index", i)

            break

    else:

        print("Element not found")

def BinarySearch():

    L = [eval(i) for i in input("Enter the list items : ").split()]

    c = eval(input("Enter element to search : "))

    L.sort()

    low = found = 0

    high = len(L) - 1

    while low <= high:

        mid = (low + high) // 2

        if L[mid] == c:

            found = 1

            break

        elif L[mid] > c:

            high = mid - 1

        else:

            low = mid + 1

    if found == 1:

        print("Element found")

    else:

        print("Element not found")

LinearSearch()

BinarySearch()

**Output**

Enter the list items : 1 2 4 5 7 8 9 10 3 5 6

Enter element to search : 11

Element not found

Enter the list items : 1 2 4 5 7 8 9 10 3 5 6

Enter element to search : 1

Element found

Experiment No. 14

Aim: Find the most common domain in phishing emails.

**Code**

L = []

d = {}

def findmostoccuringdomain():

    while True:

        c = input("Enter phishing emails (enter q to exit loop): ")

        if c == "q":

            break

        else:

            L.append(c)

    for i in range(len(L)):

        L[i] = L[i].split("@")[1]

    for i in L:

        d[i] = d.get(i, 0) + 1

    max = 0

    for i in d:

        if d[i] > max:

            max = d[i]

            domain = i

    print("\nMost occuring domain is", domain, "with", max, "occurences")

findmostoccuringdomain()

**Output**

Enter phishing emails (enter q to exit loop): jackpotwin@lottery.com

Enter phishing emails (enter q to exit loop): claimtheprize@mylife.com

Enter phishing emails (enter q to exit loop): youarethewinner@lottery.com

Enter phishing emails (enter q to exit loop): luckywinner@mylife.com

Enter phishing emails (enter q to exit loop): spinthewheel@flipkart.com

Enter phishing emails (enter q to exit loop): dealwinner@snapdeal.com

Enter phishing emails (enter q to exit loop): luckywinner@snapdeal.com

Enter phishing emails (enter q to exit loop): luckyjackpot@americanlottery.com

Enter phishing emails (enter q to exit loop): claimtheprize@lootolottery.com

Enter phishing emails (enter q to exit loop): youarelucky@mylife.com

Enter phishing emails (enter q to exit loop): q

Most occuring domain is mylife.com with 3 occurences

Experiment No. 15

Aim: Connect with a database and store employee details and display them.

**Code**

from mysql.connector import connect as c

cdb = c(user='root', password='root', host='localhost')

db = cdb.cursor()

db.execute("CREATE DATABASE IF NOT EXISTS company")

db.execute("USE company")

db.execute("CREATE TABLE IF NOT EXISTS employee (id INT PRIMARY KEY, name VARCHAR(255), salary INT, department VARCHAR(255))")

cdb.commit()

def add\_records():

    while True:

        id = int(input("Enter ID: "))

        name = input("Enter Name: ")

        salary = int(input("Enter Salary: "))

        department = input("Enter Department: ")

        db.execute("INSERT INTO employee VALUES (%s, %s, %s, %s)", (id, name, salary, department))

        cdb.commit()

        print("Record Added Successfully")

        ch = input("Do you want to add more records? (y/n): ")

        if ch != 'y':

            break

def show\_records():

    db.execute("SELECT \* FROM employee")

    rs = db.fetchall()

    print("%10s" % "Employee ID", "%20s" % "Employee Name", "%10s" % "Salary", "%20s" % "Department")

    for i in rs:

        print("%7s" % i[0], "%20s" % i[1], "%12s" % i[2], "%17s" % i[3])

while True:

    print("Menu")

    print("1. Add Records\n2. Show Records\n3. Exit")

    ch = int(input("Enter your choice: "))

    if ch == 1:

        add\_records()

    elif ch == 2:

        show\_records()

    elif ch == 3:

        exit()

**Output**

Menu

1. Add Records

2. Show Records

3. Exit

Enter your choice: 1

Enter ID: 101

Enter Name: Taha

Enter Salary: 10000

Enter Department: HR

Record Added Successfully

Do you want to add more records? (y/n): y

Enter ID: 102

Enter Name: Shehzad

Enter Salary: 10000

Enter Department: HR

Record Added Successfully

Do you want to add more records? (y/n): n

Menu

1. Add Records

2. Show Records

3. Exit

Enter your choice: 2

Employee ID Employee Name Salary Department

101 Taha 10000 HR

102 Shehzad 10000 HR

Menu

1. Add Records

2. Show Records

3. Exit

Enter your choice: 3

Experiment No. 16

Aim: Connect with a database and search employee details and display them.

**Code**

from mysql.connector import connect as c

cdb = c(user='root', password='root', host='localhost')

db = cdb.cursor()

db.execute("CREATE DATABASE IF NOT EXISTS company")

db.execute("USE company")

db.execute("CREATE TABLE IF NOT EXISTS employee (id INT PRIMARY KEY, name VARCHAR(255), salary INT, department VARCHAR(255))")

cdb.commit()

def search\_records():

    id = int(input("Enter ID: "))

    db.execute("SELECT \* FROM employee WHERE id = %s", (id,))

    rs = db.fetchall()

    if len(rs) == 0:

        print("Employee not found")

    else:

        print("Employee ID: ", rs[0][0], "\nEmployee Name: ", rs[0][1], "\nSalary: ", rs[0][2], "\nDepartment: ", rs[0][3], "\n")

print("Employee Search")

search\_records()

**Output**

#Run 1

Employee Search

Enter ID: 101

Employee ID: 101

Employee Name: Taha

Salary: 10000

Department: HR

#Run 2

Employee Search

Enter ID: 102

Employee ID: 102

Employee Name: Shehzad

Salary: 10000

Department: HR

Experiment No. 17

Aim: Connect with a database and update employee details.

**Code**

from mysql.connector import connect as c

cdb = c(user='root', password='root', host='localhost')

db = cdb.cursor()

db.execute("CREATE DATABASE IF NOT EXISTS company")

db.execute("USE company")

db.execute("CREATE TABLE IF NOT EXISTS employee (id INT PRIMARY KEY, name VARCHAR(255), salary INT, department VARCHAR(255))")

cdb.commit()

def edit\_records():

    id = int(input("Enter ID: "))

    db.execute("SELECT \* FROM employee WHERE id = %s", (id,))

    rs = db.fetchall()

    if len(rs) == 0:

        print("Employee not found")

        exit()

    else:

        print("Employee ID: ", rs[0][0], "\nEmployee Name: ", rs[0][1], "\nSalary: ", rs[0][2], "\nDepartment: ", rs[0][3], "\n")

    name = input("Enter new name: ")

    salary = int(input("Enter new salary: "))

    department = input("Enter new department: ")

    db.execute("UPDATE employee SET name = %s, salary = %s, department = %s WHERE id = %s", (name, salary, department, id))

    cdb.commit()

    print("Record updated successfully")

print("Employee Updation")

edit\_records()

**Output**

Employee Updation

Enter ID: 101

Employee ID: 101

Employee Name: Taha

Salary: 10000

Department: HR

Enter new name: Taha Parker

Enter new salary: 20000

Enter new department: IT

Record updated successfully

Experiment No. 18

Aim: Connect with a database and delete employee details.

**Code**

from mysql.connector import connect as c

cdb = c(user='root', password='root', host='localhost')

db = cdb.cursor()

db.execute("CREATE DATABASE IF NOT EXISTS company")

db.execute("USE company")

db.execute("CREATE TABLE IF NOT EXISTS employee (id INT PRIMARY KEY, name VARCHAR(255), salary INT, department VARCHAR(255))")

cdb.commit()

def delete\_records():

    id = int(input("Enter ID: "))

    db.execute("SELECT \* FROM employee WHERE id = %s", (id,))

    rs = db.fetchall()

    if len(rs) == 0:

        print("Employee not found")

        exit()

    else:

        print("Employee ID: ", rs[0][0], "\nEmployee Name: ", rs[0][1], "\nSalary: ", rs[0][2], "\nDepartment: ", rs[0][3], "\n")

    ch = input("Do you want to delete this record? (y/n): ")

    if ch == "y":

        db.execute("DELETE FROM employee WHERE id = %s", (id,))

        cdb.commit()

        print("Employee deleted successfully")

    else:

        print("Employee not deleted")

        exit()

print("Employee Deletion")

delete\_records()

**Output**

Employee Deletion

Enter ID: 102

Employee ID: 102

Employee Name: Shehzad

Salary: 10000

Department: HR

Do you want to delete this record? (y/n): y

Record deleted successfully

Experiment No. 19

Aim: Write a method CREATE() to create an EMP.csv file with the following details:

* id to store employee number of integer type
* name to store employee name of string type
* dept to store their respective department of string type
* basic to store basic salary of respective employee
* hra to be calculated from his/her basic salary which is 10% of basic
* sal to be calculated as salary = basic + hra

**Code**

import csv

def create():

    f = open('employee.csv', 'w', newline='')

    w = csv.writer(f)

    w.writerow(['ID', 'Name', 'Salary', 'Department'])

    f.flush()

    while True:

        id = int(input("Enter ID: "))

        name = input("Enter Name: ")

        dept = input("Enter Department: ")

        basic = int(input("Enter Basic Salary: "))

        hra = 0.1 \* basic  # House Rent Allowance

        sal = basic + hra

        w.writerow([id, name, dept, str(basic), str(hra), str(sal)])

        f.flush()

        ch = input("Do you want to add more records? (y/n): ")

        if ch != 'y':

            break

    f.close()

create()

**Output**

Enter ID: 101

Enter Name: Taha

Enter Department: IT

Enter Basic Salary: 1000

Do you want to add more records? (y/n): y

Enter ID: 102

Enter Name: Shehzad

Enter Department: HR

Enter Basic Salary: 1000

Do you want to add more records? (y/n): n

**File Content**

ID,Name,Salary,Department

101,Taha,IT,1000,100.0,1100.0

102,Shehzad,HR,1000,100.0,1100.0

Experiment No. 20

Aim: Copy file1.csv into file2.csv.

**Code**

import csv

def copy():

    f = open('csv1.csv', 'r')

    if not f:

        print("File not found!")

        exit()

    r = csv.reader(f)

    f1 = open('csv2.csv', 'w', newline='')

    w = csv.writer(f1)

    for row in r:

        w.writerow(row)

    f.close()

    f1.close()

copy()

**File Content**

**csv1.csv**

Name,Abbreviation,Numeric

January,Jan,1

Feburary,Feb,2

March,Mar,3

April,Apr,4

May,May,5

June,June,6

July,July,7

August,Aug,8

September,Sept,9

October,Oct,10

November,Nov,11

December,Dec,12

**csv2.csv**

Name,Abbreviation,Numeric

January,Jan,1

Feburary,Feb,2

March,Mar,3

April,Apr,4

May,May,5

June,June,6

July,July,7

August,Aug,8

September,Sept,9

October,Oct,10

November,Nov,11

December,Dec,12