

# 1. ILLUSTRATION OF FUNCTION PROGRAMMING

---

Develop a menu-driven program in Python using user defined functions to find the area of different shapes (Circle, Square and Rectangle).

## Source Code

---

```
from math import pi

def AreaSquare(s):
    area = s * s
    return area

def AreaRectangle(l, b):
    area = l * b
    return area

def AreaCircle(r):
    area = pi * (r**2)
    return area

while True:
    print("====")
    print("What would you like to do?")
    print("""
[1] Find the area of a Rectangle
[2] Find the area of a Square
[3] Find the area of a Circle
[4] Exit
""")

    ch = input("Enter your choice[1/2/3/4]: ")

    if ch == "1":
        l = int(input("Enter length of Rectangle: "))
        b = int(input("Enter breadth of Rectangle: "))
        area = AreaRectangle(l, b)
        print("Area of the Rectangle is", area)

    elif ch == "2":
        s = int(input("Enter side of Square: "))
        area = AreaSquare(s)
        print("Area of the Square is", area)
```

```
elif ch == "3":  
    s = int(input("Enter radius of Circle: "))  
    area = AreaCircle(s)  
    print("Area of the Circle is", area)  
  
elif ch == "4":  
    print("[ Exiting ]") # Break from the loop to exit  
    break  
  
else:  
    print("[ Invalid Choice ]") # In case user inputs a choice that was n
```

## OUTPUT

---

```
=====
```

What would you like to do?

- [1] Find the area of a Rectangle
- [2] Find the area of a Square
- [3] Find the area of a Circle
- [4] Exit

Enter your choice[1/2/3/4]: 1  
Enter length of Rectangle: 2  
Enter breadth of Rectangle: 3  
Area of the Rectangle is 6

```
=====
```

What would you like to do?

- [1] Find the area of a Rectangle
- [2] Find the area of a Square
- [3] Find the area of a Circle
- [4] Exit

Enter your choice[1/2/3/4]: 2  
Enter side of Square: 3  
Area of the Square is 9

```
=====
```

What would you like to do?

- [1] Find the area of a Rectangle
- [2] Find the area of a Square
- [3] Find the area of a Circle
- [4] Exit

Enter your choice[1/2/3/4]: 3  
Enter radius of Circle: 2  
Area of the Circle is 12.566370614359172

```
=====
```

What would you like to do?

- [1] Find the area of a Rectangle
- [2] Find the area of a Square
- [3] Find the area of a Circle
- [4] Exit

Enter your choice[1/2/3/4]: 4

[ Exiting ]

## 2.ILLUSTRATION OF FUNCTION PROGRAMMING (USING LIST)

---

Develop a Python Program whiccondition includes two user defined functions namely,

- create()- to create a list of roll numbers(integers)
- findroll()- to searcondition for a particular roll in a list

### Source Code

---

```
rolls = []

def create():
    while True:
        num = int(input("Enter roll number: "))
        rolls.append(num)
        ch = input("Would you like to add another number: ")
        if ch.lower() != "y":
            break

def findroll():
    num = int(input("Enter roll number to search for: "))
    if num in rolls:
        print("Entered roll number was found at index", rolls.index(num))
    else:
        print("Entered roll number was not found in list")

while True:
    print("====")
    print("What would you like to do?")
    print("""
[1] Create roll number list
[2] Find roll number
[3] Exit
""")

    ch = input("Enter your choice[1/2/3]: ")

    if ch == "1":
        create()

    elif ch == "2":
        findroll()
```

```
elif ch == "3":  
    print("[ Exiting ]") # Break from the loop to exit  
    break  
  
else:  
    print("[ Invalid Choice ]") # In case user inputs a choice that was n
```

## OUTPUT

---

```
=====
```

```
What would you like to do?
```

```
[1] Create roll number list  
[2] Find roll number  
[3] Exit
```

```
Enter your choice[1/2/3]: 1
```

```
Enter roll number: 1
```

```
Would you like to add another number: y
```

```
Enter roll number: 2
```

```
Would you like to add another number: y
```

```
Enter roll number: 3
```

```
Would you like to add another number: y
```

```
Enter roll number: 4
```

```
Would you like to add another number: n
```

```
=====
```

```
What would you like to do?
```

```
[1] Create roll number list  
[2] Find roll number  
[3] Exit
```

```
Enter your choice[1/2/3]: 2
```

```
Enter roll number to search for: 3
```

```
Entered roll number was found at index 2
```

```
=====
```

```
What would you like to do?
```

```
[1] Create roll number list  
[2] Find roll number  
[3] Exit
```

```
Enter your choice[1/2/3]: 2
```

```
Enter roll number to search for: 7
```

```
Entered roll number was not found in list
```

```
=====
```

```
What would you like to do?
```

```
[1] Create roll number list  
[2] Find roll number  
[3] Exit
```

```
Enter your choice[1/2/3]: 3  
[ Exiting ]
```

### 3. ILLUSTRATION OF MODULE PROGRAMMING

---

Create a package named "mymodule" containing three modules cube, sphere and cylinder to find the surface area and volume of these shapes. Later, import these modules into the main program and invoke the functions as a menu-driven program.

#### Source Code

---

```
import mymodule

while True:
    print("====")
    print("What would you like to do?")
    print("""
[1] Find surface area and volume of Cube
[2] Find surface area and volume of Sphere
[3] Find surface area and volume of Cylinder
[4] Exit
""")

    ch = input("Enter your choice[1/2/3/4]: ")

    if ch == "1":
        s = int(input("Enter side of Cube: "))
        mymodule.Cube(s)

    elif ch == "2":
        r = int(input("Enter radius of Sphere: "))
        mymodule.Sphere(r)

    elif ch == "3":
        r = int(input("Enter radius of Cylinder: "))
        h= int(input("Enter height of Cylinder: "))
        mymodule.Cylinder(r, h)

    elif ch == "4":
        print("[ Exiting ]) # Break from the loop to exit
        break

    else:
        print("[ Invalid Choice ]) # In case user inputs a choice that was n
```

# OUTPUT

---

```
=====
```

```
What would you like to do?
```

- [1] Find surface area and volume of Cube
- [2] Find surface area and volume of Sphere
- [3] Find surface area and volume of Cylinder
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
```

```
Enter side of Cube: 2
```

```
Surface area of the Cube is 24
```

```
Volume of the Cube is 8
```

```
=====
```

```
What would you like to do?
```

- [1] Find surface area and volume of Cube
- [2] Find surface area and volume of Sphere
- [3] Find surface area and volume of Cylinder
- [4] Exit

```
Enter your choice[1/2/3/4]: 2
```

```
Enter radius of Sphere: 2
```

```
Surface area of the Sphere is 50.26548245743669
```

```
Volume of the Sphere is 33.510321638291124
```

```
=====
```

```
What would you like to do?
```

- [1] Find surface area and volume of Cube
- [2] Find surface area and volume of Sphere
- [3] Find surface area and volume of Cylinder
- [4] Exit

```
Enter your choice[1/2/3/4]: 3
```

```
Enter radius of Cylinder: 2
```

```
Enter height of Cylinder: 3
```

```
Surface area of the Cylinder is 62.83185307179586
```

```
Volume of the Cylinder is 37.69911184307752
```

```
=====
```

```
What would you like to do?
```

- [1] Find surface area and volume of Cube
- [2] Find surface area and volume of Sphere
- [3] Find surface area and volume of Cylinder
- [4] Exit

```
Enter your choice[1/2/3/4]: 4
```

```
[ Exiting ]
```



# 4. ILLUSTRATION OF TEXT FILE PROGRAMMING - I

---

Develop a python program to do the following tasks:

- Create a text file with multiple lines of text in it.
- Read the text File and display the number of vowels, consonants, lowercase, uppercase, spaces and digits in the file.

## Source Code

---

```
# Create the file
with open("lines.txt", "w") as f:
    f.write("Once upon a time\n")
    f.write("a man lived in the woods\n")
    f.write("He was a very old man\n")
    f.write("He was a 90 years old\n")
    f.write("He lived a happy life")

with open("lines.txt") as f:
    data = f.read()
    print(data)
    print()

# variables for taking count
vowels = 0
consonants = 0
lcase = 0
ucase = 0
spaces = 0
digits = 0

for letter in data:
    if letter.isalpha():
        if letter.lower() in ["a", "e", "i", "o", "u"]:
            vowels+=1
        else:
            consonants+=1

        if letter.isupper():
            ucase+=1
        if letter.islower():
            lcase+=1
```

```
elif letter.isdigit():
    digits+=1

elif letter.isspace():
    spaces+=1

print("There are", vowels, "vowels")
print("There are", consonants, "consonants")
print("There are", lcase, "lowercase letters")
print("There are", ucase, "uppercase letters")
print("There are", spaces, "spaces")
print("There are", digits, "digits")
```

## OUTPUT

---

Once upon a time  
a man lived in the woods  
He was a very old man  
He was a 90 years old  
He lived a happy life

There are 34 vowels  
There are 45 consonants  
There are 75 lowercase letters  
There are 4 uppercase letters  
There are 26 spaces  
There are 2 digits

# 5. ILLUSTRATION OF TEXT FILE PROGRAMMING-II

---

Develop a python program to do the following task:

- Create a text file with multiple lines of text in it and print the same, to the standard output device
- Copy all those lines that contain the word 'the' to a new file.

## Source Code

---

```
# Create the file
with open("lines.txt", "w") as f:
    f.write("Once upon a time\n")
    f.write("a man lived in the woods\n")
    f.write("He was a very old man\n")
    f.write("He was a 90 years old\n")
    f.write("He lived a happy life")

with open("lines.txt") as f:
    data = f.readlines()
    print("-- lines.txt --")
    print("".join(data))
    print()

newlines=[]
for line in data:
    words = line.split()
    if "the" in words:
        newlines.append(line)

with open("new lines.txt", "w") as f:
    for line in newlines:
        f.write(line)

with open("new lines.txt") as f:
    data = f.read()
    print("-- new lines.txt --")
    print(data)
    print()
```

## OUTPUT

---

-- lines.txt --  
Once upon a time  
a man lived in the woods  
He was a very old man  
He was a 90 years old  
He lived a happy life

-- new lines.txt --  
a man lived in the woods

# 6. ILLUSTRATION OF TEXT FILE PROGRAMMING-III

---

Develop a program to create a text file to with a story in it and do the following tasks:

- To count the frequency of an inputted word in the file
- To read a random line and display it.

## Source Code

---

```
import random

def create():
    with open("lines.txt", "w") as f:
        f.write("Once upon a time\n")
        f.write("a man lived in the woods\n")
        f.write("He was a very old man\n")
        f.write("He was a 90 years old\n")
        f.write("He lived a happy life")

def count():
    with open("lines.txt") as f:
        data = f.read()
        data = data.lower()
        words = data.split()

    chk_word = input("Enter word to count: ")
    count = words.count(chk_word)

    print(chk_word, "appeared", count, "times")

def random_line():
    with open("lines.txt") as f:
        lines = f.readlines()

    num = random.randrange(0, len(lines))

    print("The random line is:")
    print("    ", lines[num])

create()

while True:
```

```

print("====")
print("What would you like to do?")
print("""
[1] Count the frequency of a word
[2] Read a random line
[3] Exit
""")

ch = input("Enter your choice[1/2/3]: ")

if ch == "1":
    count()

elif ch == "2":
    random_line()

elif ch == "3":
    print("[ Exiting ]") # Break from the loop to exit
    break

else:
    print("[ Invalid Choice ]") # In case user inputs a choice that was not listed

```

## OUTPUT

---

```

=====
What would you like to do?

[1] Count the frequency of a word
[2] Read a random line
[3] Exit

Enter your choice[1/2/3]: 1
Enter word to count: he
he appeared 3 times
=====
What would you like to do?

[1] Count the frequency of a word
[2] Read a random line
[3] Exit

Enter your choice[1/2/3]: 2
The random line is:
    Once upon a time
=====
What would you like to do?

```

- [1] Count the frequency of a word
- [2] Read a random line
- [3] Exit

Enter your choice[1/2/3]: 3  
[ Exiting ]

# 7. ILLUSTRATION OF BINARY FILE PROGRAMMING-I

---

A binary file named "movies.dat" contain certain records of certain movies (movieID, movieName, rating).Write a menu driven python program to do the following tasks:

1. Append a movie
2. Search for a movie based on the Movie ID
3. Read and display all movies

## Source Code

---

```
import pickle

f = open("movies.dat", "a") # Ensure that the file exists
f.close()

def appendMovie():
    movieID = int(input("Enter movie ID: "))
    movieName = input("Enter movie name: ")
    rating = int(input("Enter movie rating out of 10: "))

    with open("movies.dat", "ab") as f:
        pickle.dump([movieID, movieName, rating], f)

def searchMovie():
    movies = []
    with open("movies.dat", "rb") as f:
        while True:
            try: # Using a try block to catch errors
                movie = pickle.load(f)
                movies.append(movie)
            except:
                break

    movieID = int(input("Enter MovieID: "))

    found = False # Using a loop to search for a values
    for item in movies:
        if item[0] == movieID:
            print("-----")
            print("ID      :", item[0])
```

```

        print("Name  :", item[1])
        print("Rating:", item[2])
        print("-----")
        found = True
        break

    if not found:
        print("Movie not found")

def showMovies():
    movies = []
    with open("movies.dat", "rb") as f:
        while True:
            try: # Using a try block to catch errors
                movie = pickle.load(f)
                movies.append(movie)
            except EOFError:
                break
        for item in movies:
            print("-----")
            print("ID   :", item[0])
            print("Name  :", item[1])
            print("Rating:", item[2])
            print("-----")

    while True:
        print("=====")
        print("What would you like to do?")
        print("""
[1] Append a movie
[2] Search for a movie
[3] Show all movies
[4] Exit
""")

        ch = input("Enter your choice[1/2/3/4]: ")

        if ch == "1":
            appendMovie()

        elif ch == "2":
            searchMovie()

        elif ch == "3":
            showMovies()

        elif ch == "4":
            print("[ Exiting ]") # Break from the loop to exit
            break

```

```
else:  
    print("[ Invalid Choice ]") # In case user inputs a choice that was n
```

## OUTPUT

---

```
=====
```

```
What would you like to do?
```

- [1] Append a movie
- [2] Search for a movie
- [3] Show all movies
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
```

```
Enter movie ID: 1
```

```
Enter movie name: Movie
```

```
Enter movie rating out of 10: 7
```

```
=====
```

```
What would you like to do?
```

- [1] Append a movie
- [2] Search for a movie
- [3] Show all movies
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
```

```
Enter movie ID: 2
```

```
Enter movie name: Movie 2
```

```
Enter movie rating out of 10: 9
```

```
=====
```

```
What would you like to do?
```

- [1] Append a movie
- [2] Search for a movie
- [3] Show all movies
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
```

```
Enter movie ID: 3
```

```
Enter movie name: Movie 2 Sequel
```

```
Enter movie rating out of 10: 10
```

```
=====
```

```
What would you like to do?
```

- [1] Append a movie
- [2] Search for a movie
- [3] Show all movies
- [4] Exit

```
Enter your choice[1/2/3/4]: 3
```

```
-----  
ID      : 1  
Name    : Movie  
Rating: 7
```

```
-----  
-----  
ID      : 2  
Name    : Movie 2  
Rating: 9
```

```
-----  
-----  
ID      : 3  
Name    : Movie 2 Sequel  
Rating: 10
```

```
=====
```

```
What would you like to do?
```

- [1] Append a movie
- [2] Search for a movie
- [3] Show all movies
- [4] Exit

```
Enter your choice[1/2/3/4]: 2
```

```
Enter MovieID: 2
```

```
-----  
ID      : 2  
Name    : Movie 2  
Rating: 9
```

```
=====
```

```
What would you like to do?
```

- [1] Append a movie
- [2] Search for a movie
- [3] Show all movies
- [4] Exit

```
Enter your choice[1/2/3/4]: 4
```

```
[ Exiting ]
```

# 8. ILLUSTRATION OF BINARY FILE PROGRAMMING-II

---

A binary file named "flight.dat" which will contain certain records of flight (flightid, flightname and number of passengers). Write a menu driven program to do the following task:

1. Append a record
2. Delete a record
3. Read and display all.

## Source Code

---

```
import pickle

f = open("flight.dat", "a") # Ensure that the file exists
f.close()

flights = []
with open("flight.dat", "rb") as f:
    while True:
        try: # Using a try block to catch errors
            flight = pickle.load(f)
            flights.append(flight)
        except EOFError:
            break

def appendFlight():
    flightID = int(input("Enter flight ID: "))
    flightName = input("Enter flight name: ")
    nop = int(input("Enter number of passengers: "))

    with open("flight.dat", "ab") as f:
        pickle.dump([flightID, flightName, nop], f)

def deleteFlight():
    flightID = int(input("Enter flight ID: "))

    flights = []
    with open("flight.dat", "rb") as f:
        while True:
            try: # Using a try block to catch errors
```

```

        flight = pickle.load(f)
        flights.append(flight)
    except EOFError:
        break

newflights = []
for flight in flights:
    if flight[0] != flightID:
        newflights.append(flight)

with open("flight.dat", "wb") as f:
    for flight in newflights:
        pickle.dump(flight, f)

print("Removed flight with ID", flightID)

def showFlights():
    flights = []
    with open("flight.dat", "rb") as f:
        while True:
            try: # Using a try block to catch errors
                flight = pickle.load(f)
                flights.append(flight)
            except EOFError:
                break

    for flight in flights:
        print("-----")
        print("ID      :", flight[0])
        print("Name    :", flight[1])
        print("Passenger:", flight[2])
        print("-----")

while True:
    print("=====")
    print("What would you like to do?")
    print("""
[1] Append a flight
[2] Delete a flight
[3] Show all flights
[4] Exit
""")

    ch = input("Enter your choice[1/2/3/4]: ")

    if ch == "1":
        appendFlight()

```

```
    elif ch == "2":  
        deleteFlight()  
  
    elif ch == "3":  
        showFlights()  
  
    elif ch == "4":  
        print("[ Exiting ]") # Break from the loop to exit  
        break  
  
else:  
    print("[ Invalid Choice ]") # In case user inputs a choice that was n
```

## OUTPUT

---

```
=====
```

What would you like to do?

- [1] Append a flight
- [2] Delete a flight
- [3] Show all flights
- [4] Exit

Enter your choice[1/2/3/4]: 1

Enter flight ID: 1

Enter flight name: Air Jet

Enter number of passengers: 32

```
=====
```

What would you like to do?

- [1] Append a flight
- [2] Delete a flight
- [3] Show all flights
- [4] Exit

Enter your choice[1/2/3/4]: 1

Enter flight ID: 2

Enter flight name: Wind Jet

Enter number of passengers: 33

```
=====
```

What would you like to do?

- [1] Append a flight
- [2] Delete a flight
- [3] Show all flights
- [4] Exit

Enter your choice[1/2/3/4]: 1

Enter flight ID: 3

```
Enter flight name: Fly Jet
Enter number of passengers: 56
=====
What would you like to do?
```

- [1] Append a flight
- [2] Delete a flight
- [3] Show all flights
- [4] Exit

```
Enter your choice[1/2/3/4]: 3
```

```
-----
ID      : 1
Name    : Air Jet
Passenger: 32
-----
```

```
-----
ID      : 2
Name    : Wind Jet
Passenger: 33
-----
```

```
-----
ID      : 3
Name    : Fly Jet
Passenger: 56
-----
```

```
=====
What would you like to do?
```

- [1] Append a flight
- [2] Delete a flight
- [3] Show all flights
- [4] Exit

```
Enter your choice[1/2/3/4]: 2
```

```
Enter flight ID: 2
Removed flight with ID 2
=====
```

```
What would you like to do?
```

- [1] Append a flight
- [2] Delete a flight
- [3] Show all flights
- [4] Exit

```
Enter your choice[1/2/3/4]: 3
```

```
-----
ID      : 1
Name    : Air Jet
Passenger: 32
-----
```

ID : 3  
Name : Fly Jet  
Passengers: 56

-----  
=====

What would you like to do?

- [1] Append a flight
- [2] Delete a flight
- [3] Show all flights
- [4] Exit

Enter your choice[1/2/3/4]: 4

[ Exiting ]

# 9. ILLUSTRATION OF BINARY FILE PROGRAMMING-III

---

A binary file named inventory.dat contain certain records of stock (product id, product name, quantity and price). Write a menu driven python program to do the following task:

1. Append a product record
2. Update a product based on the product id
3. Read and display all products

## Source Code

---

```
import pickle

f = open("inventory.dat", "a") # Ensure that the file exists
f.close()

products = []
with open("inventory.dat", "rb") as f:
    while True:
        try: # Using a try block to catch errors
            product = pickle.load(f)
            products.append(product)
        except EOFError:
            break

def appendProduct():
    productID = int(input("Enter product ID: "))
    productName = input("Enter product name: ")
    productQnt = int(input("Enter quantity: "))
    productPrice = int(input("Enter product price: "))

    with open("inventory.dat", "ab") as f:
        pickle.dump([productID, productName, productQnt, productPrice], f)

def updateProduct():
    productID = int(input("Enter product ID: "))
    productName = input("Enter new product name: ")
    productQnt = int(input("Enter new quantity: "))
    productPrice = int(input("Enter new product price: "))
```

```

products = []
with open("inventory.dat", "rb") as f:
    while True:
        try: # Using a try block to catch errors
            product = pickle.load(f)
            products.append(product)
        except EOFError:
            break

newProducts = []
for product in products:
    if product[0] == productID:
        newProducts.append([productID, productName, productQnt, productPr
    else:
        newProducts.append(product)

with open("inventory.dat", "ab") as f:
    for product in newProducts:
        pickle.dump(product, f)

print("Updated product with ID", productID)
print("-----")
print("ID      :", productID)
print("Name    :", productName)
print("Quantity:", productQnt)
print("Price   :", productPrice)
print("-----")

def showProducts():
    products = []
    with open("inventory.dat", "rb") as f:
        while True:
            try: # Using a try block to catch errors
                product = pickle.load(f)
                products.append(product)
            except EOFError:
                break

    for product in products:
        print("-----")
        print("ID      :", product[0])
        print("Name    :", product[1])
        print("Quantity:", product[2])
        print("Price   :", product[3])
        print("-----")

    while True:

```

```

print("====")
print("What would you like to do?")
print("""
[1] Append a product
[2] Update a product
[3] Show all products
[4] Exit
""")

ch = input("Enter your choice[1/2/3/4]: ")

if ch == "1":
    appendProduct()

elif ch == "2":
    updateProduct()

elif ch == "3":
    showProducts()

elif ch == "4":
    print("[ Exiting ]") # Break from the loop to exit
    break

else:
    print("[ Invalid Choice ]") # In case user inputs a choice that was n

```

## OUTPUT

---

```

=====
What would you like to do?

[1] Append a product
[2] Update a product
[3] Show all products
[4] Exit

Enter your choice[1/2/3/4]: 1
Enter product ID: 1
Enter product name: A
Enter quantity: 234
Enter product price: 234
=====
What would you like to do?

[1] Append a product
[2] Update a product
[3] Show all products
[4] Exit

```

```
Enter your choice[1/2/3/4]: 1
Enter product ID: 2
Enter product name: B
Enter quantity: 234
Enter product price: 34
=====
What would you like to do?
```

- [1] Append a product
- [2] Update a product
- [3] Show all products
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
Enter product ID: 3
Enter product name: C
Enter quantity: 234
Enter product price: 45
=====
What would you like to do?
```

- [1] Append a product
- [2] Update a product
- [3] Show all products
- [4] Exit

```
Enter your choice[1/2/3/4]: 3
-----
ID      : 1
Name    : A
Quantity: 234
Price   : 234
-----
-----
ID      : 2
Name    : B
Quantity: 234
Price   : 34
-----
-----
ID      : 3
Name    : C
Quantity: 234
Price   : 45
=====
What would you like to do?
```

- [1] Append a product
- [2] Update a product
- [3] Show all products

[4] Exit

```
Enter your choice[1/2/3/4]: 2
Enter product ID: 2
Enter new product name: Z
Enter new quantity: 11
Enter new product price: 111
Updated product with ID 2
```

```
-----
ID      : 2
Name    : Z
Quantity: 11
Price   : 111
```

```
=====
What would you like to do?
```

- [1] Append a product
- [2] Update a product
- [3] Show all products
- [4] Exit

```
Enter your choice[1/2/3/4]: 3
```

```
-----
ID      : 1
Name    : A
Quantity: 234
Price   : 234
```

```
-----
ID      : 2
Name    : B
Quantity: 234
Price   : 34
```

```
-----
ID      : 3
Name    : C
Quantity: 234
Price   : 45
```

```
-----
ID      : 1
Name    : A
Quantity: 234
Price   : 234
```

```
-----
ID      : 2
Name    : Z
Quantity: 11
Price   : 111
```

```
-----  
-----  
ID      : 3  
Name    : C  
Quantity: 234  
Price   : 45  
-----
```

```
=====What would you like to do?
```

- [1] Append a product
- [2] Update a product
- [3] Show all products
- [4] Exit

```
Enter your choice[1/2/3/4]: 4
```

```
[ Exiting ]
```

# 10. SEARCH OPERATION USING DICTIONARY OBJECT

---

A binary file named "emp.dat" contain certain records of employees (empid, empname and salary).Write a menu driven python program to do the following tasks:

1. Append a record
2. Search a record
3. Read and display all

## Source Code

---

```
import pickle

f = open("emp.dat", "a") # Ensure that the file exists
f.close()

def appendEmp():
    empID = int(input("Enter employee ID: "))
    empName = input("Enter emp name: ")
    salary = int(input("Enter emp salary: "))

    with open("emp.dat", "ab") as f:
        pickle.dump({"empID":empID, "empName":empName, "salary":salary}, f)

def searchEmp():
    emps = []
    with open("emp.dat", "rb") as f:
        while True:
            try: # Using a try block to catch errors
                emp = pickle.load(f)
                emps.append(emp)
            except:
                break

    empID = int(input("Enter employee ID: "))

    found = False # Using a loop to search for a values
    for item in emps:
        if item["empID"] == empID:
            print("-----")
```

```

        print("ID      :", item["empID"])
        print("Name   :", item["empName"])
        print("Salary:", item["salary"])
        print("-----")
        found = True
        break

    if not found:
        print("Employee not found")

def showEmps():
    emps = []
    with open("emp.dat", "rb") as f:
        while True:
            try: # Using a try block to catch errors
                emp = pickle.load(f)
                emps.append(emp)
            except EOFError:
                break
    for item in emps:
        print("-----")
        print("ID      :", item["empID"])
        print("Name   :", item["empName"])
        print("Salary:", item["salary"])
        print("-----")

while True:
    print("=====")
    print("What would you like to do?")
    print("""
[1] Append an employee
[2] Search for an employee
[3] Show all employees
[4] Exit
""")

    ch = input("Enter your choice[1/2/3/4]: ")

    if ch == "1":
        appendEmp()

    elif ch == "2":
        searchEmp()

    elif ch == "3":
        showEmps()

    elif ch == "4":
        print("[ Exiting ]") # Break from the loop to exit
        break

```

```
else:  
    print("[ Invalid Choice ]") # In case user inputs a choice that was n
```

## OUTPUT

---

```
=====  
What would you like to do?
```

- [1] Append an employee
- [2] Search for an employee
- [3] Show all employees
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
```

```
Enter employee ID: 1
```

```
Enter emp name: Manu
```

```
Enter emp salary: 234244
```

```
=====  
What would you like to do?
```

- [1] Append an employee
- [2] Search for an employee
- [3] Show all employees
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
```

```
Enter employee ID: 2
```

```
Enter emp name: Ninu
```

```
Enter emp salary: 23444
```

```
=====  
What would you like to do?
```

- [1] Append an employee
- [2] Search for an employee
- [3] Show all employees
- [4] Exit

```
Enter your choice[1/2/3/4]: 2
```

```
Enter employee ID: 2
```

```
-----  
ID : 2
```

```
Name : Ninu
```

```
Salary: 23444
```

```
=====  
What would you like to do?
```

- [1] Append an employee

- [2] Search for an employee
- [3] Show all employees
- [4] Exit

Enter your choice[1/2/3/4]: 3

-----

ID : 1

Name : Manu

Salary: 234244

-----

-----

ID : 2

Name : Ninu

Salary: 23444

-----

=====

What would you like to do?

- [1] Append an employee
- [2] Search for an employee
- [3] Show all employees
- [4] Exit

Enter your choice[1/2/3/4]: 4

[ Exiting ]

# 11. ILLUSTRATION OF CSV FILE PROGRAMMING– I

---

Develop a menu driven program implementing the user defined functions to perform different functions on a csv file named mobile.csv(modelid, modelname, modelprice)

1. Append a record
2. Updating a record based on modelid
3. Display all
4. Exit

## Source Code

---

```
import csv

f = open("mobile.csv", "a") # Ensure that the file exists
f.close()

def appendMobile():
    mID = input("Enter model ID: ")
    mName = input("Enter model Name: ")
    mPrice = input("Enter model Price: ")

    with open("mobile.csv", "a", newline="") as f:
        wr = csv.writer(f)
        wr.writerow([mID, mName, mPrice])

    print("Record appended to file")

def updateMobile():
    mID = input("Enter model ID: ")
    mName = input("Enter new model Name: ")
    mPrice = input("Enter new model Price: ")

    newrows = []
    with open("mobile.csv", "r", newline="") as f:
        re = csv.reader(f)
        for row in re:
            newrows.append(row)

    with open("mobile.csv", "w", newline="") as f:
        wr = csv.writer(f)
        wr.writerows(newrows)

    print("Record updated successfully")
```

```

for row in newrows:
    if row[0] == mID:
        wr.writerow([mID, mName, mPrice])
    else:
        wr.writerow(row)

print("Updated record")
print("-----")
print("modelID:", mID)
print("Name : ", mName)
print("Price : ", mPrice)
print("-----")

def showMobiles():
    rows = []
    with open("mobile.csv", "r", newline="") as f:
        re = csv.reader(f)
        for row in re:
            rows.append(row)

    for row in rows:
        print("-----")
        print("modelID:", row[0])
        print("Name : ", row[1])
        print("Price : ", row[2])
        print("-----")

    while True:
        print("=====")
        print("What would you like to do?")
        print("""
[1] Append a Model
[2] Update a Model
[3] Show all Models
[4] Exit
""")

        ch = input("Enter your choice[1/2/3/4]: ")

        if ch == "1":
            appendMobile()

        elif ch == "2":
            updateMobile()

        elif ch == "3":
            showMobiles()

        elif ch == "4":
            print("[ Exiting ]") # Break from the loop to exit
            break

```

```
else:  
    print("[ Invalid Choice ]") # In case user inputs a choice that was n
```

## OUTPUT

---

```
=====
```

```
What would you like to do?
```

- [1] Append a Model
- [2] Update a Model
- [3] Show all Models
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
```

```
Enter model ID: 1
```

```
Enter model Name: Samsung S21
```

```
Enter model Price: 32000
```

```
Record appended to file
```

```
=====
```

```
What would you like to do?
```

- [1] Append a Model
- [2] Update a Model
- [3] Show all Models
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
```

```
Enter model ID: 2
```

```
Enter model Name: IPhone 13
```

```
Enter model Price: 45000
```

```
Record appended to file
```

```
=====
```

```
What would you like to do?
```

- [1] Append a Model
- [2] Update a Model
- [3] Show all Models
- [4] Exit

```
Enter your choice[1/2/3/4]: 3
```

```
-----  
modelID: 1
```

```
Name : Samsung S21
```

```
Price : 32000
```

```
-----  
-----
```

```
modelID: 2
```

```
Name : IPhone 13
```

```
Price : 45000
```

```
=====
What would you like to do?
```

- [1] Append a Model
- [2] Update a Model
- [3] Show all Models
- [4] Exit

```
Enter your choice[1/2/3/4]: 2
```

```
Enter model ID: 1
```

```
Enter new model Name: Samsung S21 Ultra
```

```
Enter new model Price: 44000
```

```
Updated record
```

```
-----
modelID: 1
```

```
Name    : Samsung S21 Ultra
```

```
Price   : 44000
```

```
=====
What would you like to do?
```

- [1] Append a Model
- [2] Update a Model
- [3] Show all Models
- [4] Exit

```
Enter your choice[1/2/3/4]: 3
```

```
-----
modelID: 1
```

```
Name    : Samsung S21 Ultra
```

```
Price   : 44000
```

```
-----
modelID: 2
```

```
Name    : IPhone 13
```

```
Price   : 45000
```

```
=====
What would you like to do?
```

- [1] Append a Model
- [2] Update a Model
- [3] Show all Models
- [4] Exit

```
Enter your choice[1/2/3/4]: 4
```

```
[ Exiting ]
```

# 12. ILLUSTRATION OF CSV FILE PROGRAMMING– II

---

Develop a menu driven program implementing the user defined functions to perform different functions on a csv file named library.csv(bookid, bookname, nofcopies)

1. Append a record
2. Searching a record based on bookid
3. Display all
4. Exit

## Source Code

---

```
import csv

f = open("library.csv", "a") # Ensure that the file exists
f.close()

def appendBook():
    bID = input("Enter book ID: ")
    bName = input("Enter book Name: ")
    bCopy = input("Enter number of copies: ")

    with open("library.csv", "a", newline="") as f:
        wr = csv.writer(f)
        wr.writerow([bID, bName , bCopy])

    print("Record appended to file")

def searchBook():
    bID = input("Enter book ID: ")

    with open("library.csv", "r", newline="") as f:
        re = csv.reader(f)
        found = False # Using a loop to search for a values
        for row in re:
            if row[0] == bID:
                print("-----")
                print("bookID      :", row[0])
                print("Name       :", row[1])
                print("No of Copies  :", row[2])
                print("-----")
```

```

        found = True
        break
    if not found:
        print("Book with entered ID was not found")

def showBooks():
    with open("library.csv", "r", newline="") as f:
        re = csv.reader(f)
        for row in re:
            print("-----")
            print("bookID      : ", row[0])
            print("Name       : ", row[1])
            print("No of Copies   : ", row[2])
            print("-----")

    while True:
        print("====")
        print("What would you like to do?")
        print("""
[1] Append a Book
[2] Search for a Book
[3] Show all Books
[4] Exit
""")

        ch = input("Enter your choice[1/2/3/4]: ")

        if ch == "1":
            appendBook()

        elif ch == "2":
            searchBook()

        elif ch == "3":
            showBooks()

        elif ch == "4":
            print("[ Exiting ]) # Break from the loop to exit
            break

        else:
            print("[ Invalid Choice ]) # In case user inputs a choice that was n

```

## OUTPUT

---

```

=====
What would you like to do?
```

- [1] Append a Book
- [2] Search for a Book
- [3] Show all Books
- [4] Exit

Enter your choice[1/2/3/4]: 1

Enter book ID: 1

Enter book Name: Huff

Enter number of copies: 23

Record appended to file

=====

What would you like to do?

- [1] Append a Book
- [2] Search for a Book
- [3] Show all Books
- [4] Exit

Enter your choice[1/2/3/4]: 1

Enter book ID: 2

Enter book Name: Puff

Enter number of copies: 45

Record appended to file

=====

What would you like to do?

- [1] Append a Book
- [2] Search for a Book
- [3] Show all Books
- [4] Exit

Enter your choice[1/2/3/4]: 2

Enter book ID: 2

-----

bookID : 2

Name : Puff

No of Copies : 45

-----

=====

What would you like to do?

- [1] Append a Book
- [2] Search for a Book
- [3] Show all Books
- [4] Exit

Enter your choice[1/2/3/4]: 3

-----

bookID : 1

Name : Huff

No of Copies : 23

-----

```
-----  
bookID      : 2  
Name        : Puff  
No of Copies : 45  
-----
```

```
=====
```

What would you like to do?

- [1] Append a Book
- [2] Search for a Book
- [3] Show all Books
- [4] Exit

Enter your choice[1/2/3/4]: 4  
[ Exiting ]

# 13. ILLUSTRATION OF CSV FILE PROGRAMMING– III

---

Develop a menu driven program implementing the user defined functions to perform different functions on a csv file named contacts.csv(name, phone)

1. Append a contact
2. Count the number of contacts in the file
3. Display all contacts
4. Exit

## Source Code

---

```
import csv

f = open("contacts.csv", "a") # Ensure that the file exists
f.close()

def appendContact():
    cName = input("Enter name: ")
    cNum = input("Enter number: ")

    with open("contacts.csv", "a", newline="") as f:
        wr = csv.writer(f)
        wr.writerow([cName, cNum])

    print("New Contact Added")

def countContacts():
    with open("contacts.csv", "r", newline="") as f:
        re = csv.reader(f)
        print("There are", len(list(re)), "contacts in the file.")

def showContacts():
    with open("contacts.csv", "r", newline="") as f:
        re = csv.reader(f)
        for row in re:
            print("-----")
            print("Name      :", row[0])
            print("Phone No  :", row[1])
            print("-----")
```

```

while True:
    print("====")
    print("What would you like to do?")
    print("""
[1] Append a Contact
[2] Count the number of Contacts
[3] Show all Contacts
[4] Exit
""")

    ch = input("Enter your choice[1/2/3/4]: ")

    if ch == "1":
        appendContact()

    elif ch == "2":
        countContacts()

    elif ch == "3":
        showContacts()

    elif ch == "4":
        print("[ Exiting ]") # Break from the loop to exit
        break

    else:
        print("[ Invalid Choice ]") # In case user inputs a choice that was not listed

```

## OUTPUT

---

```

=====
What would you like to do?

[1] Append a Contact
[2] Count the number of Contacts
[3] Show all Contacts
[4] Exit

Enter your choice[1/2/3/4]: 1
Enter name: Alice
Enter number: 1234567890
New Contact Added
=====
What would you like to do?

[1] Append a Contact
[2] Count the number of Contacts
[3] Show all Contacts
[4] Exit

```

```
Enter your choice[1/2/3/4]: 1
Enter name: Bob
Enter number: 0987654321
New Contact Added
=====
What would you like to do?
```

- [1] Append a Contact
- [2] Count the number of Contacts
- [3] Show all Contacts
- [4] Exit

```
Enter your choice[1/2/3/4]: 2
There are 2 contacts in the file.
=====
What would you like to do?
```

- [1] Append a Contact
- [2] Count the number of Contacts
- [3] Show all Contacts
- [4] Exit

```
Enter your choice[1/2/3/4]: 3
-----
Name      : Alice
Phone No : 1234567890
-----
-----
Name      : Bob
Phone No : 0987654321
-----
=====
```

What would you like to do?

- [1] Append a Contact
- [2] Count the number of Contacts
- [3] Show all Contacts
- [4] Exit

```
Enter your choice[1/2/3/4]: 4
[ Exiting ]
```

# 14. ILLUSTRATION OF STACK PROGRAMMING USING LIST OF INTEGERS – I

---

Develop a program to implement the following stack operation in python using list of integers according to the user's choice

1. Push an integer to the stack
2. Pop integer from the stack
3. Display the stack
4. Exit

## Source Code

---

```
stack = [] # stack is a global variable

def push(num):
    stack.append(num)
    print(num, "was pushed to the stack")

def pop():
    if stack != []:
        num = stack.pop()
        print(num, "was popped from the stack")
    else:
        print("Stack Underflow")
        print("There are no items in the stack")

def showStack():
    print("Stack:")
    print("      ", stack)

while True:
    print("=====")
    print("What would you like to do?")
    print("""
[1] Push an integer to the stack
[2] Pop integer from the stack
[3] Display the stack
[4] Exit
""")
```

```

ch = input("Enter your choice[1/2/3/4]: ")

if ch == "1":
    inp = int(input("Enter number to push to stack: "))
    push(inp)

elif ch == "2":
    pop()

elif ch == "3":
    showStack()

elif ch == "4":
    print("[ Exiting ]") # Break from the loop to exit
    break

else:
    print("[ Invalid Choice ]") # In case user inputs a choice that was n

```

## OUTPUT

---

```
=====
What would you like to do?
```

- [1] Push an integer to the stack
- [2] Pop integer from the stack
- [3] Display the stack
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
Enter number to push to stack: 23
23 was pushed to the stack
=====
```

```
What would you like to do?
```

- [1] Push an integer to the stack
- [2] Pop integer from the stack
- [3] Display the stack
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
Enter number to push to stack: 32
32 was pushed to the stack
=====
```

```
What would you like to do?
```

- [1] Push an integer to the stack
- [2] Pop integer from the stack

```
[3] Display the stack  
[4] Exit
```

```
Enter your choice[1/2/3/4]: 1  
Enter number to push to stack: 43  
43 was pushed to the stack  
=====
```

```
What would you like to do?
```

```
[1] Push an integer to the stack  
[2] Pop integer from the stack  
[3] Display the stack  
[4] Exit
```

```
Enter your choice[1/2/3/4]: 1  
Enter number to push to stack: 55  
55 was pushed to the stack  
=====
```

```
What would you like to do?
```

```
[1] Push an integer to the stack  
[2] Pop integer from the stack  
[3] Display the stack  
[4] Exit
```

```
Enter your choice[1/2/3/4]: 3  
Stack:  
[23, 32, 43, 55]  
=====
```

```
What would you like to do?
```

```
[1] Push an integer to the stack  
[2] Pop integer from the stack  
[3] Display the stack  
[4] Exit
```

```
Enter your choice[1/2/3/4]: 2  
55 was popped from the stack  
=====
```

```
What would you like to do?
```

```
[1] Push an integer to the stack  
[2] Pop integer from the stack  
[3] Display the stack  
[4] Exit
```

```
Enter your choice[1/2/3/4]: 3  
Stack:  
[23, 32, 43]  
=====
```

```
What would you like to do?
```

- [1] Push an integer to the stack
- [2] Pop integer from the stack
- [3] Display the stack
- [4] Exit

Enter your choice[1/2/3/4]: 4  
[ Exiting ]

# 15. ILLUSTRATION OF STACK PROGRAMMING USING LIST OF VOTERS – II

---

Develop a python program to implement the following operations on a stack containing voters list details (ID,NAME,AGE)as per user's choice.

1. Push a voter record to the stack
2. Pop a voter record from the stack
3. Display the list of voters
4. Exit

## Source Code

---

```
stack = [] # stack is a global variable

def push(record):
    stack.append(record)
    print(record[1], "was added to the stack")

def pop():
    if stack != []:
        voter = stack.pop()
        print(voter[1], "was popped from the stack")
    else:
        print("Stack Underflow")
        print("There are no items in the stack")

def showStack():
    for voter in stack:
        print("-----")
        print("ID   :", voter[0])
        print("Name :", voter[1])
        print("Age  :", voter[2])
        print("-----")

while True:
    print("=====")
    print("What would you like to do?")
    print("""
[1] Push a voter record to the stack
[2] Pop a voter record from the stack
[3] Show the list of voters
[4] Exit
    """)
```

```

[2] Pop a voter record from the stack
[3] Display the list of voters
[4] Exit
""")

ch = input("Enter your choice[1/2/3/4]: ")

if ch == "1":
    ID = int(input("Enter Voter ID: "))
    Name = input("Enter Voter Name: ")
    Age = int(input("Enter Voter Age: "))
    push([ID, Name, Age])

elif ch == "2":
    pop()

elif ch == "3":
    showStack()

elif ch == "4":
    print("[ Exiting ]") # Break from the loop to exit
    break

else:
    print("[ Invalid Choice ]") # In case user inputs a choice that was n

```

## OUTPUT

---

```
=====
What would you like to do?
```

```
[1] Push a voter record to the stack
[2] Pop a voter record from the stack
[3] Display the list of voters
[4] Exit
```

```
Enter your choice[1/2/3/4]: 1
Enter Voter ID: 1
Enter Voter Name: Alice
Enter Voter Age: 32
Alice was added to the stack
=====
```

```
What would you like to do?

[1] Push a voter record to the stack
[2] Pop a voter record from the stack
[3] Display the list of voters
[4] Exit
```

```
Enter your choice[1/2/3/4]: 1
Enter Voter ID: 2
Enter Voter Name: Bob
Enter Voter Age: 44
Bob was added to the stack
=====
What would you like to do?
```

- [1] Push a voter record to the stack
- [2] Pop a voter record from the stack
- [3] Display the list of voters
- [4] Exit

```
Enter your choice[1/2/3/4]: 1
Enter Voter ID: 3
Enter Voter Name: Charlie
Enter Voter Age: 55
Charlie was added to the stack
=====
What would you like to do?
```

- [1] Push a voter record to the stack
- [2] Pop a voter record from the stack
- [3] Display the list of voters
- [4] Exit

```
Enter your choice[1/2/3/4]: 3
-----
ID   : 1
Name : Alice
Age   : 32
-----
ID   : 2
Name : Bob
Age   : 44
-----
ID   : 3
Name : Charlie
Age   : 55
-----
What would you like to do?
```

- [1] Push a voter record to the stack
- [2] Pop a voter record from the stack
- [3] Display the list of voters
- [4] Exit

```
Enter your choice[1/2/3/4]: 2
Charlie was popped from the stack
```

```
=====
```

What would you like to do?

- [1] Push a voter record to the stack
- [2] Pop a voter record from the stack
- [3] Display the list of voters
- [4] Exit

Enter your choice[1/2/3/4]: 3

```
-----
```

ID : 1  
Name : Alice  
Age : 32

```
-----
```

```
-----
```

ID : 2  
Name : Bob  
Age : 44

```
-----
```

```
=====
```

What would you like to do?

- [1] Push a voter record to the stack
- [2] Pop a voter record from the stack
- [3] Display the list of voters
- [4] Exit

Enter your choice[1/2/3/4]: 4

[ Exiting ]

# 16. ILLUSTRATION OF STACK PROGRAMMING USING LIST OF PRODUCTS – III

---

Develop a program to implement the following stack operations in python using list of product names.

1. Push an item name to the stack(the products whose name starts with b/B)
2. Pop item name from the stack
3. Display the stack
4. Exit

## Source Code

---

```
stack = [] # stack is a global variable

def push(name):
    if name[0] in "Bb":
        stack.append(name)
        print(name, "was added to the stack")
    else:
        print("Product name does not start with 'b'")

def pop():
    if stack != []:
        product = stack.pop()
        print(product, "was popped from the stack")
    else:
        print("Stack Underflow")
        print("There are no items in the stack")

def showStack():
    print("-----")
    for product in stack:
        print(product)
    print("-----")

while True:
    print("=====")
    print("What would you like to do?")
```

```

print("""
[1] Push a product to the stack (if it's name starts with 'b')
[2] Pop a product from the stack
[3] Display the list of products
[4] Exit
""")

ch = input("Enter your choice[1/2/3/4]: ")

if ch == "1":
    name = input("Enter product name: ")
    push(name)

elif ch == "2":
    pop()

elif ch == "3":
    showStack()

elif ch == "4":
    print("[ Exiting ]") # Break from the loop to exit
    break

else:
    print("[ Invalid Choice ]") # In case user inputs a choice that was n

```

## OUTPUT

---

```

=====
What would you like to do?

[1] Push a product to the stack (if it's name starts with 'b')
[2] Pop a product from the stack
[3] Display the list of products
[4] Exit

```

Enter your choice[1/2/3/4]: 1

Enter product name: B Pen

B Pen was added to the stack

=====

What would you like to do?

```

[1] Push a product to the stack (if it's name starts with 'b')
[2] Pop a product from the stack
[3] Display the list of products
[4] Exit

```

Enter your choice[1/2/3/4]: 1

Enter product name: Berries

```
Berries was added to the stack
=====
What would you like to do?

[1] Push a product to the stack (if it's name starts with 'b')
[2] Pop a product from the stack
[3] Display the list of products
[4] Exit

Enter your choice[1/2/3/4]: 1
Enter product name: Apples
Product name does not start with 'b'
=====
What would you like to do?

[1] Push a product to the stack (if it's name starts with 'b')
[2] Pop a product from the stack
[3] Display the list of products
[4] Exit

Enter your choice[1/2/3/4]: 3
-----
B Pen
Berries
-----
=====

What would you like to do?

[1] Push a product to the stack (if it's name starts with 'b')
[2] Pop a product from the stack
[3] Display the list of products
[4] Exit

Enter your choice[1/2/3/4]: 2
Berries was popped from the stack
=====
What would you like to do?

[1] Push a product to the stack (if it's name starts with 'b')
[2] Pop a product from the stack
[3] Display the list of products
[4] Exit

Enter your choice[1/2/3/4]: 3
-----
B Pen
-----
=====

What would you like to do?

[1] Push a product to the stack (if it's name starts with 'b')
[2] Pop a product from the stack
```

[3] Display the list of products

[4] Exit

Enter your choice[1/2/3/4]: 4

[ Exiting ]

# 17. ILLUSTRATION OF CONNECTIVITY PROGRAMMING-I

---

Integrate SQL with Python by importing the MYSQL module and to implement the DML command (SELECT). Create a table STUDENT (Roll, Name, Stream, Section). Populate the table with 4 records of your choice. Display all the records of student table.

## Source Code

---

```
import mysql.connector as msc

try: # Using a try block to catch errors
    conn = msc.connect(host='localhost',user='root',password='password',database='student')

    if (conn.is_connected()): #checking if the connection is established
        print('Connected')
    else:
        print('Connection not established')

    cur = conn.cursor()

    cur.execute('select * from student')
    rows = cur.fetchall() #retrieving data from the result set

    print('Data from the student table is as follows:\n')

    for i in rows: #displaying the table
        print(i[0], ' ', i[1], ' ', i[2], ' ', i[3])

    cur.close()
    conn.close()

except Exception as e:
    print(e)
```

## OUTPUT

---

```
Connected
Data from the student table is as follows:
```

1	Alice	COMPUTER SCIENCE	A
2	Bob	COMMERCE	B
3	Charlie	HUMANITIES	C
4	David	COMPUTER SCIENCE	D

# 18. ILLUSTRATION OF CONNECTIVITY PROGRAMMING-II

---

Integrate SQL with python by importing the MYSQL module and to implement the DML commands(INSERT and SELECT). Populate the STUDENT(Roll, Name, Stream, Section) table with 4 records of your choice using INSERT query and display all the records by using the appropriate Query

## Source Code

---

```
import mysql.connector as msc

try: # Using a try block to catch errors
    conn = msc.connect(host='localhost',user='root',password='password',database='student')

    if (conn.is_connected()): #checking if the connection is established
        print('Connected')
    else:
        print('Connection not established')

    cur = conn.cursor()

    while True:
        print("====")
        print("What would you like to do?")
        print("""
[1] Insert new record
[2] Display the table
[3] Exit
""")

        ch = input("Enter your choice[1/2/3]: ")

        if ch == "1":
            roll = input("Enter roll no of student: ")
            name = input("Enter name of student: ")
            stream = input("Enter stream of student: ")
            section = input("Enter section of student: ")

            cur.execute("insert into student values({}, '{}', '{}', '{}')".format(roll, name, stream, section))

            conn.commit()
            print("New record inserted into table")
```

```

    elif ch == "2":
        cur.execute('select * from student')
        rows = cur.fetchall() #retrieving data from the result set

        print('Data from the student table is as follows:\n')

        for i in rows: #displaying the table
            print(i[0], ' ', i[1], ' ', i[2], ' ', i[3])

    elif ch == "3":
        print("[ Exiting ]") # Break from the loop to exit
        break

    else:
        print("[ Invalid Choice ]") # In case user inputs a choice that w

    cur.close()
    conn.close()

except Exception as e:
    print(e)

```

## OUTPUT

---

```

Connected
=====
What would you like to do?

[1] Insert new record
[2] Display the table
[3] Exit

Enter your choice[1/2/3]: 2
Data from the student table is as follows:

1      Alice      COMPUTER SCIENCE      A
2      Bob        COMMERCE          B
3      Charlie    HUMANITIES        C
4      David      COMPUTER SCIENCE      D
=====

What would you like to do?

[1] Insert new record
[2] Display the table
[3] Exit

```

```

Enter your choice[1/2/3]: 1
Enter roll no of student: 5
Enter name of student: Ellie

```

```
Enter stream of student: COMMERCE
```

```
Enter section of student: C
```

```
New record inserted into table
```

```
=====
```

```
What would you like to do?
```

```
[1] Insert new record
```

```
[2] Display the table
```

```
[3] Exit
```

```
Enter your choice[1/2/3]: 2
```

```
Data from the student table is as follows:
```

```
1      Alice      COMPUTER SCIENCE      A
```

```
2      Bob        COMMERCE          B
```

```
3      Charlie    HUMANITIES       C
```

```
4      David     COMPUTER SCIENCE      D
```

```
5      Ellie     COMMERCE          C
```

```
=====
```

```
What would you like to do?
```

```
[1] Insert new record
```

```
[2] Display the table
```

```
[3] Exit
```

```
Enter your choice[1/2/3]: 3
```

```
[ Exiting ]
```

# 19. ILLUSTRATION OF CONNECTIVITY PROGRAMMING-III

---

Integrate SQL with python by importing the MYSQL module and to implement the DML commands(UPDATE and SELECT). Populate the STUDENT(Roll, Name, Stream, Section)table with 4 records of your choice and do the following tasks:

- Accept the roll no: of the student to be Updated.
- Update the Record
- Display all the records (After Updation)

## Source Code

---

```
import mysql.connector as msc

try: # Using a try block to catch errors
    conn = msc.connect(host='localhost',user='root',password='password',database='student')

    if (conn.is_connected()): #checking if the connection is established
        print('Connected')
    else:
        print('Connection not established')

    cur = conn.cursor()

    while True:
        print("====")
        print("What would you like to do?")
        print("""
[1] Update a record
[2] Display the table
[3] Exit
""")

        ch = input("Enter your choice[1/2/3]: ")

        if ch == "1":
            roll = input("Enter roll no of student to update: ")
            name = input("Enter new name of student: ")
            stream = input("Enter new stream of student: ")
            section = input("Enter new section of student: ")

            cur.execute("update student set name='{}', stream='{}', section='{}' where roll={}".format(name, stream, section, roll))

        elif ch == "2":
            cur.execute("select * from student")
            rows = cur.fetchall()
            for row in rows:
                print(row)

        elif ch == "3":
            break

        else:
            print("Invalid choice")
```

```

        conn.commit()
        print("Record has been updated")

    elif ch == "2":
        cur.execute('select * from student')
        rows = cur.fetchall() #retrieving data from the result set

        print('Data from the student table is as follows:\n')

        for i in rows: #displaying the table
            print(i[0], ' ', i[1], ' ', i[2], ' ', i[3])

    elif ch == "3":
        print("[ Exiting ]) # Break from the loop to exit
        break

    else:
        print("[ Invalid Choice ]) # In case user inputs a choice that w

    cur.close()
    conn.close()

except Exception as e:
    print(e)

```

## OUTPUT

---

```

Connected
=====
What would you like to do?

[1] Update a record
[2] Display the table
[3] Exit

Enter your choice[1/2/3]: 2
Data from the student table is as follows:

1      Alice      COMPUTER SCIENCE      A
2      Bob        COMMERCE          B
3      Charlie     HUMANITIES       C
4      David      COMPUTER SCIENCE      D
=====

What would you like to do?

[1] Update a record
[2] Display the table
[3] Exit

```

```
Enter your choice[1/2/3]: 1
Enter roll no of student to update: 4
Enter new name of student: David
Enter new stream of student: COMMERCE
Enter new section of student: D
Record has been updated
=====
What would you like to do?
```

- [1] Update a record
- [2] Display the table
- [3] Exit

```
Enter your choice[1/2/3]: 2
Data from the student table is as follows:
```

1	Alice	COMPUTER SCIENCE	A
2	Bob	COMMERCE	B
3	Charlie	HUMANITIES	C
4	David	COMMERCE	D

```
=====
What would you like to do?
```

- [1] Update a record
- [2] Display the table
- [3] Exit

```
Enter your choice[1/2/3]: 3
[ Exiting ]
```

# 20. ILLUSTRATION OF CONNECTIVITY PROGRAMMING-IV

---

Integrate SQL with python by importing the MYSQL module and to implement the DML commands(DELETE and SELECT). Populate the STUDENT(Roll, Name, Stream, Section)table with 4 records of your choice and do the following tasks:

- Accept the roll no: of the student to be deleted.
- Delete the Record
- Display all the records (After deletion)

## Source Code

---

```
import mysql.connector as msc

try: # Using a try block to catch errors
    conn = msc.connect(host='localhost',user='root',password='password',database='student')

    if (conn.is_connected()): #checking if the connection is established
        print('Connected')
    else:
        print('Connection not established')

    cur = conn.cursor()

    while True:
        print("====")
        print("What would you like to do?")
        print("""
[1] Delete a record
[2] Display the table
[3] Exit
""")

        ch = input("Enter your choice[1/2/3]: ")

        if ch == "1":
            roll = input("Enter roll no of student to delete: ")

            cur.execute("delete from student where roll={}".format(roll))

            conn.commit()
            print("Record has been deleted")
```

```

    elif ch == "2":
        cur.execute('select * from student')
        rows = cur.fetchall() #retrieving data from the result set

        print('Data from the student table is as follows:\n')

        for i in rows: #displaying the table
            print(i[0], ' ', i[1], ' ', i[2], ' ', i[3])

    elif ch == "3":
        print("[ Exiting ]") # Break from the loop to exit
        break

    else:
        print("[ Invalid Choice ]") # In case user inputs a choice that w

    cur.close()
    conn.close()

except Exception as e:
    print(e)

```

## OUTPUT

---

```

Connected
=====
What would you like to do?

[1] Delete a record
[2] Display the table
[3] Exit

Enter your choice[1/2/3]: 2
Data from the student table is as follows:

1      Alice      COMPUTER SCIENCE      A
2      Bob        COMMERCE          B
3      Charlie     HUMANITIES       C
4      David      COMPUTER SCIENCE      D
=====

What would you like to do?

[1] Delete a record
[2] Display the table
[3] Exit

```

```

Enter your choice[1/2/3]: 1
Enter roll no of student to delete: 4

```

Record has been deleted

=====

What would you like to do?

- [1] Delete a record
- [2] Display the table
- [3] Exit

Enter your choice[1/2/3]: 2

Data from the student table is as follows:

1	Alice	COMPUTER SCIENCE	A
2	Bob	COMMERCE	B
3	Charlie	HUMANITIES	C

=====

What would you like to do?

- [1] Delete a record
- [2] Display the table
- [3] Exit

Enter your choice[1/2/3]: 3

[ Exiting ]