Python lab Answers

1. Implement a python program using string function

a) Develop a python program to print a string in the reverse order.

Input: python

Output: nohtyp

Program :

```
reverse = str(input("Enter the string :"))

print(reverse[::-1])
```

---

b) Create a python program to check whether a string is a palindrome or not.

Input: madam, racecar

Output: Palindrome

Program :

```
def is_palindrome(s):
    return s == s[::-1]

# Take user input
user_input = input("Enter a string: ")

# Check and print result
if is_palindrome(user_input):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

---

c) Design a python program to count the number of characters in the string.

Input: python

Output: 6

Program:

```
# Take user input
user_input = input("Enter a string: ")
# Count the characters
char_count = len(user_input)
# Print the result
```

```
print("Number of characters in the string:", char_count)
```

---

d) Develop a python program to replace characters in the string

Input: hello
Replace last character l as i
Output: Heiio

Program:

```
# Take user input

user_input = input("Enter a string: ")

char_to_replace = input("Enter the character to replace: ")

replacement_char = input("Enter the replacement character: ")

# Replace characters

modified_string = user_input.replace(char_to_replace, replacement_char)

# Print the result

print("Modified string:", modified_string)
```

---

e) Deploy the concept of Pangram to check whether a string contains all the alphabets or not.
Input: "The quick brown fox jumps over the lazy dog"
Output: is a Pangram

Program:

```
import string

def is_pangram(s):

    alphabet = set(string.ascii_lowercase)

    return set(s.lower()) >= alphabet

# Take user input

user_input = input("Enter a string: ")
```

```
# Check and print result
if is_pangram(user_input):
    print("The string is a pangram (contains all letters of the alphabet).")
else:
print("The string is not a pangram.")
```

---

f) Prepare a python program to find all duplicate characters in string.

Input: Python Programing

Output: P, o, n

Program:

```
user_input = input("Enter a string: ")
for char in set(user_input):
    if user_input.count(char) > 1:
        print(f"'{char}' appears more than once")
```

==========

2. Develop a python program using Functions

a) Implement a python program using functions to calculate GCD of two numbers

Input: 48, 60
Output: 12

Program:

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
# Input values
num1 = 48
```

```
num2 = 60

# Calculate GCD

result = gcd(num1, num2)

# Output result

print("GCD of", num1, "and", num2, "is:", result)
```

---

b) Develop a python program to calculate Factorial of given number using Recursive function

Input: 5
Output: 120

Program:

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
# Input value
num = 5
# Calculate Factorial
result = factorial(num)
# Output result
print("Factorial of", num, "is:", result)
```

---

c) Build a python program using a user defined non recursive function CalcFact() to calculate and display the factorial of a number .The function CalcFact() accepts the number num as an argument.

Input: 6
Output: 720

Program:

```
def CalcFact(num):
    fact = 1
    for i in range(1, num + 1):
        fact *= i
    return fact
```

```
# Input value
num = 5
# Calculate Factorial
result = CalcFact(num)

# Output result
print("Factorial of", num, "is:", result)
```

---

d) Design a python program using functions to generate Fibonacci series with 'n' terms

Input: 7
Output: 0 1 1 2 3 5 8 13

Program:

```
def fibonacci(n):
    fib_series = []
    a, b = 0, 1
    for _ in range(n):
        fib_series.append(a)
        a, b = b, a + b
    return fib_series
# Input value
n = 10
# Generate Fibonacci series
result = fibonacci(n)
# Output result
print("Fibonacci series with", n, "terms:", result)
```

---

e) Write a python program using function to print prime numbers within a given range

Input: 1,100
Output: 2 3 5 7 11 13 …… 97

Program:

```
def print_primes(start, end):
    for num in range(start, end + 1):
        if num > 1:
            for i in range(2, num):
                if num % i == 0:
                    break
```

```
        else:
            print(num, end=' ')
# Input range
start = 10
end = 50
# Print prime numbers within range
print("Prime numbers between", start, "and", end, "are:")
print_primes(start, end)
```

---

f) Construct a Python function that prompts the user to enter a word and counts vowels and consonant in a word.

Input: Enter a word = python Output
Count of vowel is = 1 Count of consonant is = 5

Program:

```
def count_vowels_consonants():

    word = input("Enter a word: ").lower()

    vowels = "aeiou"

    vowels_count = sum(1 for char in word if char in vowels)

    consonants_count = sum(1 for char in word if char.isalpha() and char not in vowels)

    print(f"Vowels: {vowels_count}, Consonants: {consonants_count}")

count_vowels_consonants()
```

==========

3. Solve a python problem using conditional statements and loops.

a) Develop a python program to find largest among three numbers

Sample Input:
Enter first number: 25
Enter second number:4
Enter third number: 10
Sample Output:
The Biggest number is 25

Program:

```
def find_largest():
    num1 = int(input("Enter first number: "))
    num2 = int(input("Enter second number: "))
    num3 = int(input("Enter third number: "))
    if num1 >= num2 and num1 >= num3:
        largest = num1
    elif num2 >= num1 and num2 >= num3:
        largest = num2
    else:
        largest = num3
    print("The Biggest number is", largest)
# Call the function
find_largest()
```

---

b) Create a program to check whether the given number is Armstrong Number or not.

Input : 153
Output : Armstrong Number

Program:

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

# initialize sum

sum = 0

# find the sum of the cube of each digit

temp = num

while temp > 0:

   digit = temp % 10

   sum += digit ** 3

   temp //= 10

# display the result

if num == sum:

   print(num,"is an Armstrong number")
```

```
else:
   print(num,"is not an Armstrong number")
```

---

c) Create a python program for Number Pattern (Right Angled Triangle)

Sample Input:
Enter a number: 5
Sample Output:

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

Program:

```
rows = int(input("Enter number of rows: "))
for i in range(rows):
    for j in range(i+1):
        print(j+1, end=" ")
    print()
```

---

d) Create a program to find the sum of digits of given number

Input : 1234
Output : 10

Program:

```
def sum_of_digits(number):
    total = sum(int(digit) for digit in str(number))
    print("Sum of digits:", total)
# Input value
num = int(input("Enter a number: "))
# Find sum of digits
sum_of_digits(num)
```

---

e) Create a program to calculate Mean, Median and Mode in python

Sample Input:
Enter numbers separated by space: 2 3 4 4 5 5 5 6
Sample Output:
Mean: 4.25
Median: 4.5
Mode: 5

Program:

```
from statistics import mean, median, mode

def calculate_statistics():

    numbers = list(map(int, input("Enter numbers separated by space: ").split()))

    mean_value = mean(numbers)

    median_value = median(numbers)

    mode_value = mode(numbers)

    print("Mean:", mean_value)

    print("Median:", median_value)

    print("Mode:", mode_value)

# Call the function

calculate_statistics()
```

---

f) Create a program to calculate Matrix Multiplication.

Sample Input:
Enter elements for A matrix (2x2): 1 2
3 4
Enter elements for B matrix (2x2): 5 6
7 8
Sample Output: Resultant Matrix (A × B): [19, 22]
[43, 50]

Program:

```
def matrix_multiplication():
```

```
    # Input for Matrix A
    print("Enter elements for A matrix (2x2):")
    A = [list(map(int, input().split())) for _ in range(2)]
    # Input for Matrix B
    print("Enter elements for B matrix (2x2):")
    B = [list(map(int, input().split())) for _ in range(2)]
    # Matrix multiplication
    result = [[0, 0], [0, 0]]
    for i in range(2):
        for j in range(2):
            result[i][j] = A[i][0] * B[0][j] + A[i][1] * B[1][j]
    # Output result
    print("Resultant Matrix (A × B):")
    for row in result:
        print(row)
# Call the function
matrix_multiplication()
```

==========

4. Construct a python program real-time applications using Sets and Dictionaries.

a) Develop a Python program to remove duplicates from a List using a Set

Input: customer_list = ["Alice", "Bob", "Alice", "David", "Bob"]
Output: Unique Customers: ['Alice', 'Bob', 'David']

Program:

```
def remove_duplicates(customer_list):
    unique_customers = list(set(customer_list))
    print("Unique Customers:", unique_customers)

# Input list
```

```
customer_list = ["Alice", "Bob", "Alice", "David", "Bob"]

# Remove duplicates
remove_duplicates(customer_list)
```

---

b) Develop a Python program to program implement a simple phonebook lookup using a dictionary.

Input: Enter a name to search: Bob
Output: Phone Number: 987-654-3210

Program:

```
def phonebook_lookup():
    phonebook = {
        "Alice": "123-456-7890",
        "Bob": "987-654-3210",
        "David": "555-555-5555"
    }

    name = input("Enter a name to search: ")
    phone_number = phonebook.get(name, "Not found")
    print("Phone Number:", phone_number)

# Call the function
phonebook_lookup()
```

---

c) Design a Python program to sort a dictionary containing sales data.

Input: sales = { "Alice": 1500, "Bob": 2500, "Charlie": 1800 }
Output: Sorted Sales Data: {'Bob': 2500, 'Charlie': 1800, 'Alice': 1500}

Program:

```
def sort_sales_data():
    sales = {"Alice": 1500, "Bob": 2500, "Charlie": 1800}
    sorted_sales = dict(sorted(sales.items(), key=lambda item: item[1], reverse=True))
    print("Sorted Sales Data:", sorted_sales)

# Call the function
sort_sales_data()
```

---

d) Create a Python program to find common categories of products in two stores(set).

Input: store1 = {"Electronics", "Groceries", "Clothing"} store2 = {"Furniture", "Electronics", "Clothing"}
Output: Common Categories: {'Electronics', 'Clothing'}

Program:

```
def find_common_categories():
    store1 = {"Electronics", "Groceries", "Clothing"}
    store2 = {"Furniture", "Electronics", "Clothing"}

    common_categories = store1.intersection(store2)
    print("Common Categories:", common_categories)

# Call the function
find_common_categories()
```

---

e) Develop a Python program to count the frequency of words in a given text using dictionary.

Input: sample_text = "apple orange apple banana apple orange"
Output: Word Frequencies: {'apple': 3, 'orange': 2, 'banana': 1}

Program:

```
def count_word_frequencies():
    sample_text = "apple orange apple banana apple orange"
    words = sample_text.split()
    word_count = {}

    for word in words:
        word_count[word] = word_count.get(word, 0) + 1

    print("Word Frequencies:", word_count)

# Call the function
count_word_frequencies()
```

f) Prepare a Python program to check if a given string has all unique characters.

Input: Enter a string: hello
Output: Has all unique characters? False

Program:

```
def has_unique_characters():
    string = input("Enter a string: ")
    unique_chars = set(string)

    print("Has all unique characters?", len(unique_chars) == len(string))

# Call the function
has_unique_characters()
```

==========

5. Contruct a python program real-time/technical applications using Lists and Tuples.

a) Develop a python program to Swap Two Elements in a List

Input: [1, 2, 3, 4, 5] i = 1, j = 3
Output: [1, 4, 3, 2, 5]

Program:

```
def swap_elements(lst, i, j):
    lst[i], lst[j] = lst[j], lst[i]
    return lst
# Input list
lst = [1, 2, 3, 4, 5]
i, j = 1, 3
# Swap elements
swapped_list = swap_elements(lst, i, j)
print("Swapped List:", swapped_list)
```

---

b) Create a python program to reverse a list.

Input: [2, 3, 4, 5,6]
Output: [6, 5, 4, 3, 2]

Program:

```
def reverse_list(lst):

    return lst[::-1]

# Input list

lst = [2, 3, 4, 5, 6]

# Reverse the list
```

```
reversed_list = reverse_list(lst)

print("Reversed List:", reversed_list)
```

C) Design a python program to sort a tuple in ascending order.

Input: (5, 2, 8, 1, 9)
Output: (1, 2, 5, 8, 9)

Program:

```
def sort_tuple(tpl):
    return tuple(sorted(tpl))

# Input tuple
tpl = (5, 2, 8, 1, 9)

# Sort the tuple
sorted_tuple = sort_tuple(tpl)
print("Sorted Tuple:", sorted_tuple)
```

---

d) Develop a python program to perform a Linear Search using List.

Input: [45,43,67,89,32]
Enter the element to be searched:67
Output: Element found at index: 2

Program:

```
def linear_search(lst, target):
    for i in range(len(lst)):
        if lst[i] == target:
            return i
    return -1

# Input list
lst = [45, 43, 67, 89, 32]
target = int(input("Enter the element to be searched: "))

# Perform linear search
index = linear_search(lst, target)
```

```
# Output result
if index != -1:
    print("Element found at index:", index)
else:
    print("Element not found")
```

---

e) Develop a Python program to implement a simple student database using tuples to store student information.

Input : Student ID: 12345, Student Name: Selva, Grade: A
Output: Student ID: 12345, Student Name: Selva, Grade: A

Program:

```
def student_database():
    student = ("12345", "Selva", "A")
    print("Student ID:", student[0])
    print("Student Name:", student[1])
    print("Grade:", student[2])

# Call the function
student_database()
```

---

f) Develop a Python program to find the maximum and minimum elements in a tuple.

Input: (5, 2, 8, 1, 9)
Output: Maximum: 9, Minimum: 1

Program:

```
def find_max_min(tpl):
    print("Maximum:", max(tpl))
    print("Minimum:", min(tpl))

# Input tuple
tpl = (5, 2, 8, 1, 9)

# Find max and min
find_max_min(tpl)
```

6)Create a Python program to demonstrate polymorphism with inheritance. (Single, Multilevel Inheritance, Hierarchical)

a) Create a Python program to demonstrate single inheritance using a Dog class that inherits from an Animal class.

Input: Dog
Output: Woof!

```
# Base class
class Animal:
    def speak(self):
        print("Some generic sound")

# Derived class
class Dog(Animal):
    def speak(self):
        print("Woof!")

# Input
animal_type = input("Input: ")

# Create instance based on input
if animal_type.lower() == "dog":
    pet = Dog()
    print("Output: ", end="")
    pet.speak()
else:
    print("Unknown animal")
```

Output:
Input: Dog
Output: Woof!

---

b) Develop a Python program to show multilevel inheritance using a Cat class that inherits from a Mammal class, which in turn inherits from an Animal class.
Input: Cat
Output: Meow!

```
# Base class
class Animal:
    def _init_(self):
        print("Animal created")

# Intermediate class
class Mammal(Animal):
    def _init_(self):
        super()._init_()
        print("Mammal created")
```

```
# Derived class
class Cat(Mammal):
    def _init_(self):
        super()._init_()
        print("Cat created")

    def speak(self):
        print("Meow!")

# Input
animal_type = input("Input: ")

# Create instance based on input
if animal_type.lower() == "cat":
    pet = Cat()
    print("Output: ", end="")
    pet.speak()
else:
    print("Unknown animal")
```

Output:
Input: Cat
Animal created
Mammal created
Cat created
Output: Meow!

---

c) Design a Python program to illustrate hierarchical inheritance using a Dog and Cat class that both inherit from an Animal class.
Input: Dog, Cat
Output: Woof!, Meow!

```
# Base class
class Animal:
    def speak(self):
        print("Animal sound")

# Derived class
class Dog(Animal):
    def speak(self):
        print("Woof!")

class Cat(Animal):
```

```
    def speak(self):
        print("Meow!")

# Input
animals = input("Input: ").split(", ")

# Output
print("Output:")
for animal in animals:
    if animal.lower() == "dog":
        Dog().speak()
    elif animal.lower() == "cat":
        Cat().speak()
    else:
        print("Unknown animal")
```

Output:

```
Input: Dog, Cat
Output:
Woof!
Meow!
```

---

d) Create a Python program to demonstrate polymorphism using method overriding in a Dog and Cat class.
Input: Dog, Cat
Output: Woof!, Meow!

```
# Base class
class Animal:
    def speak(self):
        print("Some animal sound")

# Derived classes
class Dog(Animal):
    def speak(self):
        print("Woof!")

class Cat(Animal):
    def speak(self):
        print("Meow!")

# Input
animals = input("Input: ").split(", ")

# Output
print("Output:")
```

```
for animal in animals:
    if animal.lower() == "dog":
        Dog().speak()
    elif animal.lower() == "cat":
        Cat().speak()
    else:
        print("Unknown animal")
```

Output:
Input: Dog, Cat
Output:
Woof!
Meow!

---

e)Develop a Python program to show polymorphism using method overloading in a Shape class with Circle and Rectangle subclasses.
Input: Circle, Rectangle
Output: Area of Circle, Area of Rectangle

```
import math

# Base class
class Shape:
    def area(self):
        pass

class Circle(Shape):
    def _init_(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

class Rectangle(Shape):
    def _init_(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

# Input and Output
shape_input = input("Input (Circle or Rectangle): ").strip().lower()

if shape_input == "circle":
```

```
    c = Circle(radius=5)
    print("Output: Area of Circle =", round(c.area(), 2))
elif shape_input == "rectangle":
    r = Rectangle(length=4, width=6)
    print("Output: Area of Rectangle =", r.area())
else:
    print("Unknown shape")
```

Output:
Input (Circle or Rectangle): Circle
Output: Area of Circle = 78.54

---

f) Prepare a Python program to demonstrate polymorphism using operator overloading in a Vector class.
Input: Vector1, Vector2
Output: Vector addition

```
class Vector:
    def _init_(self, x, y):
        self.x = x
        self.y = y

    def _add_(self, other):
        return Vector(self.x + other.x, self.y + other.y)

    def _str_(self):
        return f"({self.x}, {self.y})"

# Input
print("Input: Vector1 and Vector2")
v1 = Vector(2, 3)
v2 = Vector(4, 5)

# Addition
v3 = v1 + v2

# Output
print("Output: Vector Addition =", v3)
```

Output:
Input: Vector1 and Vector2
Output: Vector Addition = (6, 8)

==========

7.a) Implement a simple calendar in python program without using the calendar module using string array or list.

Sample Input 1:

Enter month (1-12): 2

Enter year: 2024

Sample Output:

February 2024

Sun Mon Tue Wed Thu Fri Sat

1 2 3 4 5 6 7

8 9 10 11 12 13 14

15 16 17 18 19 20 21

22 23 24 25 26 27 29

29

```
# List of days and months
days = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]
months = ["January", "February", "March", "April", "May", "June",
          "July", "August", "September", "October", "November", "December"]
month_days = [31, 28, 31, 30, 31, 30,
              31, 31, 30, 31, 30, 31]

# Function to check for leap year
def is_leap(year):
    return (year % 4 == 0 and (year % 100 != 0 or year % 400 == 0))

# Zeller’s Congruence to find the day of week
def get_start_day(month, year):
    if month < 3:
        month += 12
        year -= 1
    q = 1
    k = year % 100
    j = year // 100
    h = (q + 13*(month+1)//5 + k + k//4 + j//4 + 5*j) % 7
    return (h + 6) % 7  # Convert to 0=Sun, ..., 6=Sat

# Input
month = int(input("Enter month (1-12): "))
year = int(input("Enter year: "))

# Adjust February days if leap year
if is_leap(year):
    month_days[1] = 29

# Output
print(f"\n{months[month-1]} {year}")
```

```
print(" ".join(days))

# Print calendar
start_day = get_start_day(month, year)
num_days = month_days[month-1]

# Print leading spaces
print("    " * start_day, end="")

# Print days
for date in range(1, num_days + 1):
    print(f"{date:2}", end="  ")
    start_day += 1
    if start_day == 7:
        start_day = 0
        print()
```

Output:

```
Enter month (1-12): 2
Enter year: 2024
February 2024
Sun Mon Tue Wed Thu Fri Sat
1   2   3   4   5   6   7
8   9   10  11  12  13  14
15  16  17  18  19  20  21
22  23  24  25  26  27  29
29
```

7.b) Develop a Python program to determine if a given year is a leap year without using the datetime module. Implement the leap year check manually based on divisibility rules.

Sample Input:

Enter a year: 2024

Sample Output :

2024 is a Leap Year.

```
# Input
year = int(input("Enter a year: "))

# Logic
if (year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)):
    print(f"{year} is a Leap Year.")
else:
    print(f"{year} is NOT a Leap Year.")
```

Output:

Enter a year: 2024
2024 is a Leap Year.

==========

8.a) Write a Python program to demonstrate a user-defined exception for validating a person's age. If the age is below 18, raise a custom exception called Underage Error.
Sample Input 1:
Enter your age: 16
Sample Output 1:
Error: Age must be 18 or above.
Sample Input 2:
Enter your age: 20
Sample Output 2:
Age 20 is valid. You are eligible.

```
# Custom Exception
class UnderageError(Exception):
    pass

# Input
try:
    age = int(input("Enter your age: "))
    if age < 18:
        raise UnderageError("Error: Age must be 18 or above.")
    else:
        print(f"Age {age} is valid. You are eligible.")
except UnderageError as e:
    print(e)
```

Output:

Enter your age: 16
Error: Age must be 18 or above.

b) Write a Python program to demonstrate handling the built-in Zero Division Error exception when dividing two numbers.
Sample Input 1:
Enter numerator: 10
Enter denominator: 2
Sample Output 1:
Result: 5.0
Sample Input 2:

Enter numerator: 10
Enter denominator: 0
Sample Output 2:
Error: Division by zero is not allowed.

```
# Input
try:
    numerator = float(input("Enter numerator: "))
    denominator = float(input("Enter denominator: "))
    result = numerator / denominator
    print(f"Result: {result}")
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
```

Output:
Enter numerator: 10
Enter denominator: 0
Error: Division by zero is not allowed.

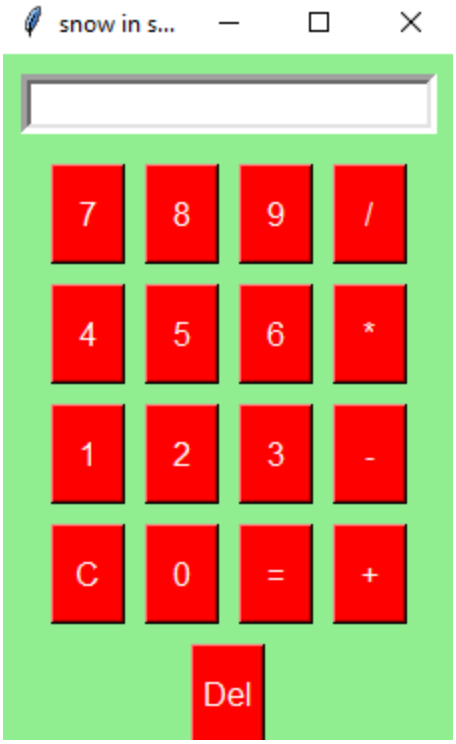
==========

9.a) Design and implement a simple GUI-based Calculator using Tkinter in Python. The application should allow the user to enter numbers and perform basic arithmetic operations like addition, subtraction, multiplication, and division.

```
import tkinter as tk

def on_click(event):
    text = event.widget.cget("text")
    if text == "=":
        try:
            result = eval(entry.get())
            entry.delete(0, tk.END)
            entry.insert(tk.END, result)
        except:
            entry.delete(0, tk.END)
            entry.insert(tk.END, "Error")
    elif text == "C":
        entry.delete(0, tk.END)
    elif text == "Del":
        current = entry.get()
        entry.delete(0, tk.END)
        entry.insert(tk.END, current[:-1])
    else:
        entry.insert(tk.END, text)
```

```
root = tk.Tk()
root.title("Simple Calculator")
root.configure(bg="green")

entry = tk.Entry(root, font=("Arial", 20), bd=5)
entry.pack(fill=tk.BOTH, ipadx=8, pady=10, padx=10)

buttons = [
    ["7", "8", "9", "/"],
    ["4", "5", "6", "*"],
    ["1", "2", "3", "-"],
    ["C", "0", "=", "+"],
    ["Del"]
]

for row in buttons:
    frame = tk.Frame(root, bg="green")
    frame.pack()
    for btn_text in row:
        btn = tk.Button(frame, text=btn_text, font=("Arial", 18), bg="red", fg="white", width=5,
height=2)
        btn.pack(side=tk.LEFT, padx=5, pady=5)
        btn.bind("<Button-1>", on_click)

root.mainloop()
```

Output:



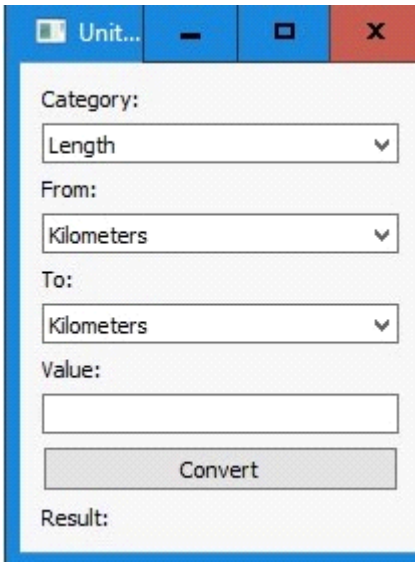
---

9. b) Develop a PyQt5 GUI application that allows users to convert between different units (length, weight, or volume). For example, convert kilometers to miles, grams to ounces, or liters to gallons.
Sample Input:
Category: Length
Convert from: Kilometers
Convert to: Miles
Value: 5
Sample Output:
Result: 3.10686 miles

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QComboBox, QLineEdit,
QPushButton, QVBoxLayout
conversion_factors = {'Length': {'Kilometers': {'Miles': 0.621371},'Miles': {'Kilometers':
1.60934}},'Weight':{'Grams':{'Ounces': 0.035274},'Ounces':{'Grams':
28.3495}},'Volume':{'Liters': {'Gallons': 0.264172},'Gallons': {'Liters': 3.78541}}}
class UnitConverter(QWidget):
    def _init_(self):
        super()._init_()
        self.setWindowTitle('Unit Converter')
        self.initUI()
    def initUI(self):
```

```
        layout = QVBoxLayout()
        self.category = QComboBox()
        self.category.addItems(conversion_factors.keys())
        self.category.currentIndexChanged.connect(self.update_units)
        self.from_unit = QComboBox()
        self.to_unit = QComboBox()
        self.value_input = QLineEdit()
        self.result_label = QLabel("Result: ")
        convert_btn = QPushButton("Convert")
        convert_btn.clicked.connect(self.convert)
        layout.addWidget(QLabel("Category:"))
        layout.addWidget(self.category)
        layout.addWidget(QLabel("From:"))
        layout.addWidget(self.from_unit)
        layout.addWidget(QLabel("To:"))
        layout.addWidget(self.to_unit)
        layout.addWidget(QLabel("Value:"))
        layout.addWidget(self.value_input)
        layout.addWidget(convert_btn)
        layout.addWidget(self.result_label)
        self.setLayout(layout)
        self.update_units()
    def update_units(self):
        category = self.category.currentText()
        units = list(conversion_factors[category].keys())
        self.from_unit.clear()
        self.to_unit.clear()
        self.from_unit.addItems(units)
        self.to_unit.addItems(units)
    def convert(self):
        category = self.category.currentText()
        from_u = self.from_unit.currentText()
        to_u = self.to_unit.currentText()
        try:
            value = float(self.value_input.text())
            factor = conversion_factors[category][from_u][to_u]
            result = value * factor
            self.result_label.setText(f"Result: {result:.6f} {to_u}")
        except:
            self.result_label.setText("Invalid conversion")
app = QApplication(sys.argv)
window = UnitConverter()
window.show()
sys.exit(app.exec_())
```

Output:



Exp.no:10

- **a) Implementing a web application with MySQL database integration for CRUD operations (Flask / Django Framework)**

Sample Input & Output for CRUD Operations

- **Creating a New Student (INSERT Operation)**

Input (Form Submission) Name:
John Doe
Age: 21
Course: Computer Science
[Submit]

Output (Success Message)
Student 'John Doe' added successfully!

Database Record After Insertion

ID	Name	Age	Course
1	John Doe	21	Computer Science

- **Viewing All Students (READ Operation)**

Output (Student List)

ID	Name	Age	Course
1	John Doe	21	Computer Science

- Jane Smith 22 Data Science

- Alice Lee 20 Cyber Security

- **Updating a Student's Details (UPDATE Operation) Input (Form Submission)**

Select Student ID: 1 New
Age: 22
New Course: AI & ML [Update]

Output (Success Message)
Student 'John Doe' updated successfully!

Database Record After Update

ID	Name	Age	Course
1	John Doe	22	AI & ML

- **Deleting a Student (DELETE Operation)**

Input (Delete Request)
Select Student ID: 3 [Delete]
Output (Success Message)
Student 'Alice Lee' deleted successfully!

Database Record After Deletion

ID	Name	Age	Course
1	John Doe	22	AI & ML
2	Jane Smith	22	Data Science

```
from flask import Flask, request, render_template, redirect
from flask_mysqldb import MySQL

app = Flask(_name_)

# MySQL configurations
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'your_user'
app.config['MYSQL_PASSWORD'] = 'your_password'
app.config['MYSQL_DB'] = 'studentdb'

mysql = MySQL(app)

@app.route('/')
def index():
```

```
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM students")
    data = cur.fetchall()
    return render_template('index.html', students=data)

@app.route('/add', methods=['POST'])
def add():
    name = request.form['name']
    age = request.form['age']
    course = request.form['course']
    cur = mysql.connection.cursor()
    cur.execute("INSERT INTO students(name, age, course) VALUES (%s, %s, %s)", (name,
age, course))
    mysql.connection.commit()
    return redirect('/')

@app.route('/update', methods=['POST'])
def update():
    id = request.form['id']
    age = request.form['age']
    course = request.form['course']
    cur = mysql.connection.cursor()
    cur.execute("UPDATE students SET age=%s, course=%s WHERE id=%s", (age, course, id))
    mysql.connection.commit()
    return redirect('/')

@app.route('/delete/<id>')
def delete(id):
    cur = mysql.connection.cursor()
    cur.execute("DELETE FROM students WHERE id=%s", (id,))
    mysql.connection.commit()
    return redirect('/')

if _name_ == '_main_':
    app.run(debug=True)
```

Output:

--- CREATE OPERATION ---
Input:
Name: John Doe
Age: 21
Course: Computer Science
[Submit]

Output:
Student 'John Doe' added successfully!

Database Table After Insert:
ID Name Age Course
1 John Doe 21 Computer Science

--- READ OPERATION ---
Output:
ID Name Age Course
1 John Doe 21 Computer Science
2 Jane Smith 22 Data Science
3 Alice Lee 20 Cyber Security

--- UPDATE OPERATION ---
Input:
Select Student ID: 1
New Age: 22
New Course: AI & ML
[Update]

Output:
Student 'John Doe' updated successfully!

Database Table After Update:
ID Name Age Course
1 John Doe 22 AI & ML
2 Jane Smith 22 Data Science
3 Alice Lee 20 Cyber Security

--- DELETE OPERATION ---
Input:
Select Student ID: 3
[Delete]

Output:
Student 'Alice Lee' deleted successfully!

Database Table After Deletion:
ID Name Age Course
1 John Doe 22 AI & ML
2 Jane Smith 22 Data Science

10.b) Implementing a web application with SQLite database integration for CRUD operations (Flask / Django Framework)
Sample Input & Output for CRUD Operations

- **Creating a New Student (INSERT Operation)**

Input (Form Submission) Name:
John Doe
Age: 21
Course: Computer Science
[Submit]

Output (Success Message)
Student 'John Doe' added successfully!

Database Record After Insertion

ID	Name	Age	Course
1	John Doe	21	Computer Science

- **Viewing All Students (READ Operation)**

Output (Student List)

ID	Name	Age	Course
1	John Doe	21	Computer Science

- Jane Smith 22 Data Science
- Alice Lee 20 Cyber Security

- **Updating a Student's Details (UPDATE Operation) Input (Form Submission)**

Select Student ID: 1 New
Age: 22

New Course: AI & ML [Update]

Output (Success Message)
Student 'John Doe' updated successfully!

Database Record After Update

ID	**Name**	**Age**	**Course**
1	John Doe	22	AI & ML

- **Deleting a Student (DELETE Operation)**

Input (Delete Request)
Select Student ID: 3 [Delete]
Output (Success Message)
Student 'Alice Lee' deleted successfully!

Database Record After Deletion

ID	Name	Age	Course
1	John Doe	22	AI & ML

```
from flask import Flask, request, render_template, redirect
import sqlite3

app = Flask(_name_)
DATABASE = 'students.db'

def get_db():
    conn = sqlite3.connect(DATABASE)
    conn.row_factory = sqlite3.Row
    return conn

@app.route('/')
def index():
    conn = get_db()
    cur = conn.cursor()
    cur.execute("SELECT * FROM students")
    data = cur.fetchall()
    return render_template('index.html', students=data)

@app.route('/add', methods=['POST'])
def add():
    name = request.form['name']
    age = request.form['age']
    course = request.form['course']
    conn = get_db()
    cur = conn.cursor()
    cur.execute("INSERT INTO students(name, age, course) VALUES (?, ?, ?)", (name, age,
course))
    conn.commit()
    return redirect('/')

@app.route('/update', methods=['POST'])
def update():
    id = request.form['id']
    age = request.form['age']
    course = request.form['course']
    conn = get_db()
    cur = conn.cursor()
    cur.execute("UPDATE students SET age=?, course=? WHERE id=?", (age, course, id))
    conn.commit()
```

```
    return redirect('/')

@app.route('/delete/<id>')
def delete(id):
    conn = get_db()
    cur = conn.cursor()
    cur.execute("DELETE FROM students WHERE id=?", (id,))
    conn.commit()
    return redirect('/')

if _name_ == '_main_':
    # Make sure to create the 'students' table in 'students.db' before running
    app.run(debug=True)
```

Output:

--- CREATE OPERATION ---
Input:
Name: John Doe
Age: 21
Course: Computer Science
[Submit]

Output:
Student 'John Doe' added successfully!

Database Table After Insert:
ID Name Age Course
1 John Doe 21 Computer Science

--- READ OPERATION ---
Output:
ID Name Age Course
1 John Doe 21 Computer Science
2 Jane Smith 22 Data Science
3 Alice Lee 20 Cyber Security

--- UPDATE OPERATION ---
Input:
Select Student ID: 1
New Age: 22
New Course: AI & ML
[Update]

Output:
Student 'John Doe' updated successfully!

Database Table After Update:
ID Name Age Course
1 John Doe 22 AI & ML
2 Jane Smith 22 Data Science
3 Alice Lee 20 Cyber Security

--- DELETE OPERATION ---
Input:
Select Student ID: 3
[Delete]

Output:
Student 'Alice Lee' deleted successfully!

Database Table After Deletion:
ID Name Age Course
1 John Doe 22 AI & ML
2 Jane Smith 22 Data Science