Advance Programming

Assessment 1

Muhammad Awais

Chapter 1

Python Refresher

Chap-1 Ex-1

```python
# Chapter 1 Exercise 1: User Input Output

# Using print for the opening statement
print("Hello! Mr/Mrs,")

# Using input function to gather data from the user
User_name = input("Please state your name?\n").title()
# Using input function to gather data from the user
Users_age = int(input("And your age?\n"))

# Using print function to print again and calling the variable for the users name
print(f"It is very nice meeting you, {User_name}.")

# Writing the code for the calculations of the length of the name using len function
name_length = len(User_name)
# Printing the length of the name and calling the variable of the length
print(f"Your name consists of '{name_length}' numbers of alphabets.")

# Writing the code for adding one in the users age by calling the variable
adding_age = Users_age + 1
print(f"You will be turning {adding_age} in a years time.")
```

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe "C:\Users\work\PycharmProjects\pythonProject1\Chap-1_ Ex-1.py"
Hello! Mr/Mrs,
Please state your name?
 Muhammad Awais
And your age?
20
It is very nice meeting you,  Muhammad Awais.
Your name consists of '15' numbers of alphabets.
You will be turning 21 in a years time.

Process finished with exit code 0
```

Chap-1 Ex-2

```python
# Chapter 1 Exercise 2: Maths

# Getting the integer from the user
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

# Creating variables for the calculations and writing codes for it
cal_sum = num1 + num2
cal_diff = num1 - num2
cal_product = num1 * num2
# Writing code for checking if the second number is not zero before performing division
if num2 != 0:
    cal_quot = num1 / num2
    cal_remain = num1 % num2
else: # If number is 0 then print the following
    cal_quot = "Undefined (division by zero)"
    cal_remain = "Undefined (division by zero)"

# Printing the result of the calculations
print(f"Sum: {cal_sum}")
print(f"Difference: {cal_diff}")
print(f"Product: {cal_product}")
print(f"Quotient: {cal_quot}")
print(f"Remainder: {cal_remain}")
```

output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Ex-2.py
Enter the first number: 14
Enter the second number: 12
Sum: 26
Difference: 2
Product: 168
Quotient: 1.1666666666666667
Remainder: 2

Process finished with exit code 0
```

Chap-1 Ex-3

```python
# Chapter 1 Exercise 3: Is it Triangle

# Defining a function to check if the entered lengths form a triangle
1 usage
def check_triangle(a, b, c):
    return a + b > c and b + c > a and c + a > b

# Defining a function to determine the type of triangle based on its side lengths
1 usage
def type_of_triangle(a, b, c):
    # Checking if all three sides are equal, which indicates an equilateral triangle.
    if a == b == c:
        return "Equilateral"
    # Checking if at least two sides are equal, which indicates an isosceles triangle.
    elif a == b or b == c or c == a:
        return "Isosceles"
    # If conditions are not met, then it's a scalene triangle.
    else:
        return "Scalene"

1 usage
def valid():
    # The main function for taking user input and determining triangle validity and type
    print("Enter the three sides of the triangle to check if it's valid or invalid:")
    side_a = float(input("Side A: "))
    side_b = float(input("Side B: "))
    side_c = float(input("Side C: "))

    if check_triangle(side_a, side_b, side_c):
        # If the sides are forming a valid triangle, then determining the type
        print("The entered lengths of the sides form a valid triangle.")
        classifying_triangle = type_of_triangle(side_a, side_b, side_c)
        print(f"The triangle is {classifying_triangle}.")
```

```python
1 usage
def valid():
    # The main function for taking user input and determining triangle validity and type
    print("Enter the three sides of the triangle to check if it's valid or invalid:")
    side_a = float(input("Side A: "))
    side_b = float(input("Side B: "))
    side_c = float(input("Side C: "))

    if check_triangle(side_a, side_b, side_c):
        # If the sides are forming a valid triangle, then determining the type
        print("The entered lengths of the sides form a valid triangle.")
        classifying_triangle = type_of_triangle(side_a, side_b, side_c)
        print(f"The triangle is {classifying_triangle}.")
    else:
        # If the sides do not form a valid triangle, inform the user
        print("The entered lengths of the sides do not form a valid triangle.")

# Executing the main function
valid()
```

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Ex-3.py
Enter the three sides of the triangle to check if it's valid or invalid:
Side A: 12
Side B: 12
Side C: 12
The entered lengths of the sides form a valid triangle.
The triangle is Equilateral.

Process finished with exit code 0
```

Chap 1 Ex-4

```python
# Chapter 1 Exercise 4: Largest Number

# Defining a function to find the largest number among three input numbers
1 usage
def find_largest_number(a, b, c):
    # Comparing the numbers to find the largest
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c

# Defining main function to take user input and find the largest number
def main():
    print("Enter the three numbers:")
    num1 = float(input("Number 1: "))
    num2 = float(input("Number 2: "))
    num3 = float(input("Number 3: "))

    # Calling the function to find the largest number
    largest_number = find_largest_number(num1, num2, num3)
    print(f"The largest number is: {largest_number}")

# Executing the main function
main()
```

output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Ex-4.py
Enter the three numbers:
Number 1: 65
Number 2: 84
Number 3: -99
The largest number is: 84.0

Process finished with exit code 0
```

Chap-1 Ex-5

```python
# Chapter 1 Exercise 5: Continue

# Defining main function to ask the user if they want to continue repeatedly
def main():
    Times = 0  # Initializing a counter for the number of times the loop is executed

    while True:
        # Prompting the user to enter 'Y' or 'N' to continue or exit the loop
        user_input = input("Do you want to continue? (Y/N): ").upper()

        # Exiting the loop if the user enters 'N'
        if user_input == 'N':
            break

        # Increasing the counter if the user enters 'Y'
        if user_input == 'Y':
            Times += 1
        else:
            print("Invalid input. Please enter 'Y' or 'N'.")

    # Displaying the number of times the loop was executed
    print(f"The loop was executed {Times} times.")

# Executing the main function
main()
```

output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Ex-5.py
Do you want to continue? (Y/N): y
Do you want to continue? (Y/N): Y
Do you want to continue? (Y/N): y
Do you want to continue? (Y/N): n
The loop was executed 3 times.

Process finished with exit code 0
```

Chap-1 Ex-6

```python
# Chapter 1 Exercise 6: FizzBuzz

# Defining function to perform the FizzBuzz operation from numbers 1 to 100
1 usage
def fizz_or_buzz():
    for a in range(1, 101):
        # Checking if the number is divisible by both 3 and 5
        if a % 3 == 0 and a % 5 == 0:
            print("FizzBuzz")
        # Checking if the number is divisible by 3
        elif a % 3 == 0:
            print("Fizz")
        # Checking if the number is divisible by 5
        elif a % 5 == 0:
            print("Buzz")
        # If none of the above conditions are applicable, print the number itself
        else:
            print(a)

# Executing the fizz_buzz function
fizz_or_buzz()
```

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Ex-6.py
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz
26
Fizz
28
29
FizzBuzz
31
32
```

And continuing in the same format
till hundred

Chap-1 Ex-7

```python
# Chapter 1 Exercise 7: Even Numbers

1 usage
def print_even_numbers():
    # Using for loop to print even numbers from 1 to 100
    for a in range(1, 101):
        # Checking if the number is odd, and if it is, skip to the next continuation
        if a % 2 != 0:
            continue
        # Printing the even number
        print(a)

# Calling the function to print even numbers
print_even_numbers()
```

output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap=1_Ex-7.py
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
```

Continuing till hundred

Chap-1 Ex-8

Code with output

```python
# Chapter 1 Exercise 8: Print Pattern

# Defining a function to display a pattern with ascending numbers in each row
1 usage
def displaying_pattern(rows):
    # Iterating in each row
    for a in range(1, rows + 1):
        # Iterating through each column in the current row
        for b in range(1, a + 1):
            # Printing the current number with a space at the end
            print(b, end=" ")
        # Move to the next line after printing all numbers in the current row
        print()

# Executing the display_pattern function
displaying_pattern(5)
```

Chap-1_Ex-8 × Chap-5_Ex-4 ×

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Ex-8.py
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

Process finished with exit code 0
```

Chap-1 Ex-9

```python
# Chapter 1 Exercise 9: Integer List

# Creating an int list with 10 values
my_list = [5, 10, 3, 8, 1, 7, 2, 9, 4, 6]

# Output the list using a for loop
print("Original List:")
for num in my_list:
    print(num, end=" ")

# Output the highest and lowest value
print("\nHighest Value:", max(my_list))
print("Lowest Value:", min(my_list))

# Sort the elements in ascending order
my_list.sort()
print("\nAscending Order:", my_list)

# Sort the elements in descending order
my_list.sort(reverse=True)
print("Descending Order:", my_list)

# Append two elements
my_list.append(11)
my_list.append(0)

# Print the list after appending
print("\nList after Appending:")
for num in my_list:
    print(num, end=" ")
```

output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Ex-9.py
Original List:
5 10 3 8 1 7 2 9 4 6
Highest Value: 10
Lowest Value: 1

Ascending Order: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Descending Order: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

List after Appending:
10 9 8 7 6 5 4 3 2 1 11 0
Process finished with exit code 0
```

Chap-1 Ex-10

Code and Output

```python
# Chapter 1 Exercise 10 : Film Dictionary

# Creating a film dictionary
film = {
    "Title": "Inception",
    "Director": "Christopher Nolan",
    "Year": 2010,
    "Genre": "Sci-Fi",
    "Rating": 8.8
}

# Displaying film details using a loop
print("Film Details:")
for key, value in film.items():
    print(f"{key}: {value}")
```

Chap-1_Ex-10 × Chap-5_Ex-4 ×

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Ex-10.py
Film Details:
Title: Inception
Director: Christopher Nolan
Year: 2010
Genre: Sci-Fi
Rating: 8.8

Process finished with exit code 0
```

Chap-1 Ex-11

```python
# Chapter 1 Exercise 11: Year Tuples

# Create a tuple with values
year = (2017, 2003, 2011, 2005, 1987, 2009, 2020, 2018, 2009)

# Accessing the value at index -3
value_at_minus_3 = year[-3]
print("Value at index -3:", value_at_minus_3)

# Reversing the tuple and print the original tuple and reversed tuple
reversed_year = tuple(reversed(year))
print("\nOriginal Tuple:", year)
print("Reversed Tuple:", reversed_year)

# Counting the number of times 2009 is in the tuple (use count() method)
count_2009 = year.count(2009)
print("\nNumber of times 2009 is in the tuple:", count_2009)

# Getting the index value of 2018 using index method
index_of_2018 = year.index(2018)
print("\nIndex of 2018:", index_of_2018)

# Finding the length of the given tuple using len() method
tuple_length = len(year)
print("\nLength of the tuple:", tuple_length)
```

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Ex-11.py
Value at index -3: 2020

Original Tuple: (2017, 2003, 2011, 2005, 1987, 2009, 2020, 2018, 2009)
Reversed Tuple: (2009, 2018, 2020, 2009, 1987, 2005, 2011, 2003, 2017)

Number of times 2009 is in the tuple: 2

Index of 2018: 7

Length of the tuple: 9

Process finished with exit code 0
```

Chap-1 Ex-12

```python
# Chapter 1 Exercise 12: Area Function

# Importing math library for math calculations
import math

# Defining function to calculate the area of a square
1 usage
def calculating_square_area():
    length_of_side = float(input("Enter the side length of the square: "))
    area = length_of_side ** 2
    print(f"The area of the square is: {area}")

# Defining function to calculate the area of a circle
1 usage
def calculating_circle_area():
    radius = float(input("Enter the radius of the circle: "))
    area = math.pi * (radius ** 2)
    print(f"The area of the circle is: {area:.2f}")

# Defining function to calculate the area of a triangle
1 usage
def calculating_triangle_area():
    base = float(input("Enter the base length of the triangle: "))
    height = float(input("Enter the height of the triangle: "))
    area = 0.5 * base * height
    print(f"The area of the triangle is: {area}")
```

```python
# Displaying the menu
while True:
    # Printing the menu options
    print("\nMenu:")
    print("1: Calculate the area of a square")
    print("2: Calculate the area of a circle")
    print("3: Calculate the area of a triangle")
    print("4: Exit")

    # Taking user input for menu choice
    choice = input("Enter your choice (1-4): ")

    # Checking the user's choice and calling the corresponding function
    if choice == "1":
        calculating_square_area()  # Calling function to calculate the area of a square
    elif choice == "2":
        calculating_circle_area()  # Calling function to calculate the area of a circle
    elif choice == "3":
        calculating_triangle_area()  # Calling function to calculate the area of a triangle
    elif choice == "4":
        print("Exiting the program. Goodbye!")
        break  # Exiting the loop and ending the program
    else:
        print("Invalid choice. Please enter a number between 1 and 4.")  # Informing the user about an invalid choice
```

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Ex-12.py

Menu:
1: Calculate the area of a square
2: Calculate the area of a circle
3: Calculate the area of a triangle
4: Exit
Enter your choice (1-4): 2
Enter the radius of the circle: 14
The area of the circle is: 615.75

Menu:
1: Calculate the area of a square
2: Calculate the area of a circle
3: Calculate the area of a triangle
4: Exit
Enter your choice (1-4): 4
Exiting the program. Goodbye!

Process finished with exit code 0
```

Chap-1 Bonus A

Code and Output

```python
# Chapter 1 Bonus A: Multiplication Tables

# Outer loop for each table from 1 to 10
for a in range(1, 11):
    print(f"\nMultiplication table of : {a}")

    # Inner loop for multiplying numbers from 1 to 10
    for b in range(1, 11):
        result = a * b
        print(f"{a} x {b} = {result}")
```

for a in range(1, 11) › for b in range(1, 11)

Chap-1_Bonus-A × Chap-5_Ex-4 ×

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Bonus-A.py

Multiplication table of : 1
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

Multiplication table of : 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
```

Continuing in the same format till the table of 10

Chap-1 Bonus B

```python
# Chapter 1 Bonus B: Locations List

# Given list
locations = ['dubai', 'paris', 'switzerland', 'London', 'amsterdam', 'New York']

# Print the original list and find its length
print("Original List:", locations)
print("Length of the List:", len(locations))

# Use sorted() to print the list in alphabetical order without modifying the actual list
sorted_alphabetical = sorted(locations)
print("\nSorted in Alphabetical Order:", sorted_alphabetical)

# Print the original list to show that it's still in its original order
print("Original List (unchanged):", locations)

# Use sorted() to print the list in reverse alphabetical order without changing the order of the original list
sorted_reverse = sorted(locations, reverse=True)
print("\nSorted in Reverse Alphabetical Order:", sorted_reverse)

# Print the original list to show that it's still in its original order
print("Original List (unchanged):", locations)

# Use sorted() to print the list in reverse alphabetical order without changing the order of the original list
sorted_reverse = sorted(locations, reverse=True)
print("\nSorted in Reverse Alphabetical Order:", sorted_reverse)

# Print the original list to show that it's still in its original order
print("Original List (unchanged):", locations)

# Use reverse() to change the order of the list
locations.reverse()
print("\nReversed List:", locations)

# Use sort() to change the list so it's stored in alphabetical order
locations.sort()
print("\nList Sorted in Alphabetical Order:", locations)

# Use sort() to change the list so it's stored in reverse alphabetical order
locations.sort(reverse=True)
print("\nList Sorted in Reverse Alphabetical Order:", locations)
```

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Bonus-B.py
Original List: ['dubai', 'paris', 'switzerland', 'London', 'amsterdam', 'New York']
Length of the List: 6

Sorted in Alphabetical Order: ['London', 'New York', 'amsterdam', 'dubai', 'paris', 'switzerland']
Original List (unchanged): ['dubai', 'paris', 'switzerland', 'London', 'amsterdam', 'New York']

Sorted in Reverse Alphabetical Order: ['switzerland', 'paris', 'dubai', 'amsterdam', 'New York', 'London']
Original List (unchanged): ['dubai', 'paris', 'switzerland', 'London', 'amsterdam', 'New York']

Reversed List: ['New York', 'amsterdam', 'London', 'switzerland', 'paris', 'dubai']

List Sorted in Alphabetical Order: ['London', 'New York', 'amsterdam', 'dubai', 'paris', 'switzerland']

List Sorted in Reverse Alphabetical Order: ['switzerland', 'paris', 'dubai', 'amsterdam', 'New York', 'London']

Process finished with exit code 0
```

Chap 1 Bonus C

```
# Chapter 1 Bonus C: Calculators Function

# Defining function for adding two numbers
1 usage
def adding(a, b):
    return a + b

# Defining function for subtracting two numbers
1 usage
def subtracting(a, b):
    return a - b

# Defining function for multiplying two numbers
1 usage
def multiplying(a, b):
    return a * b

# Defining function for dividing two numbers
1 usage
def dividing(a, b):
    if b != 0:
        return a / b
    else:
        return "Unable to divide by zero."

# Defining function to calculate modulus of two numbers
1 usage
def modulus(a, b):
    if b != 0:
        return a % b
    else:
        return "Cannot calculate modulus with zero divisor."

# Main loop for calculator functionality
while True:
    # Display calculator menu
    print("\nCalculator Menu:")
    print("a. Add")
    print("b. Subtract")
    print("c. Multiply")
    print("d. Divide")
    print("e. Modulus")

    # Get user's choice
    choice = input("Enter your choice (a-e): ")

    # Input values for the calculation
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))

    # Perform calculation based on user's choice
    if choice == "a":
        result = adding(num1, num2)
    elif choice == "b":
        result = subtracting(num1, num2)
    elif choice == "c":
        result = multiplying(num1, num2)
    elif choice == "d":
        result = dividing(num1, num2)
    elif choice == "e":
```

```python
# Perform calculation based on user's choice
if choice == "a":
    result = adding(num1, num2)
elif choice == "b":
    result = subtracting(num1, num2)
elif choice == "c":
    result = multiplying(num1, num2)
elif choice == "d":
    result = dividing(num1, num2)
elif choice == "e":
    result = modulus(num1, num2)
else:
    print("Invalid choice. Please enter a number between a to e.")
    continue

# Display the result
print(f"\nResult: {result}")

# Asking the user if they are looking to perform another calculation
more_cal = input("Do you want to perform another calculation? (yes/no): ").lower()
# If entered 'yes' the calculation will continue
if more_cal == 'yes':
    continue
# If entered 'no' the calculation will stop
elif more_cal == 'no':
    print("Exiting the calculator. Goodbye!")
    break
```

Output

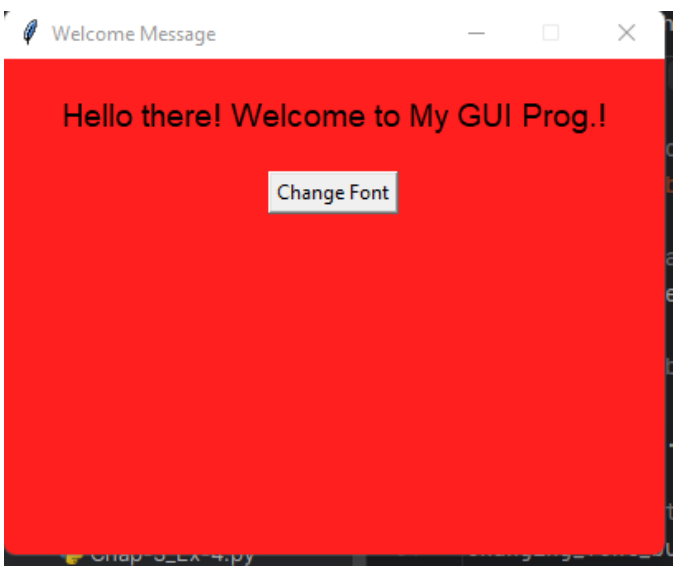
```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-1_Bonus-C.py

Calculator Menu:
a. Add
b. Subtract
c. Multiply
d. Divide
e. Modulus
Enter your choice (a-e): b
Enter the first number: 12
Enter the second number: 5

Result: 7.0
Do you want to perform another calculation? (yes/no): no
Exiting the calculator. Goodbye!

Process finished with exit code 0
```

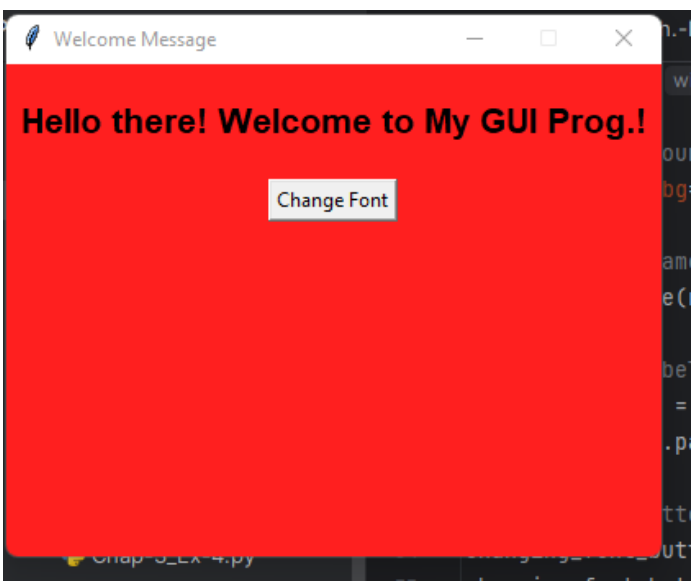
Chapter 2
Graphical User Interface

Chap-2 Ex-1

```python
# Chapter 2 Exercise 1: Welcome

# Importing tkinter library
import tkinter as tk
from tkinter import font

# Defining function for changing the fonts
1 usage
def changing_font():
    font_new = font.Font(label_for_frame, label_for_frame.cget("font"))
    font_new.config(family="Arial", weight="bold", size=16)
    label_for_frame.configure(font=font_new)

# Creating the main window
root = tk.Tk()
root.title("Welcome Message")

# Setting the default window size
root.geometry("400x300")

# Disabling the resizing of the window
root.resizable( width: False,  height: False)
```

Output

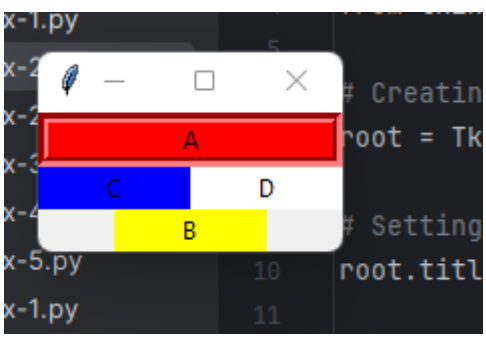




Chap-2 Ex-2 a

```python
# Chapter 2 Exercise 2a: Using pack

# using the tkinter library
from tkinter import *

# Creating the main window
root = Tk()

# Setting the title of the window
root.title("Four Labels Example")

# Creating Label 1 with style
label1 = Label(root, text="A", bg="red", width=10, bd=5, relief="groove")
label1.pack(side=TOP, fill=X, expand=1)  # Using pack to display and arrange Label 1 at the top

# Creating Label 2 with style
label2 = Label(root, text="B", bg="yellow", width=10)
label2.pack(side=BOTTOM)  # Using pack to display and arrange Label 2 at the bottom

# Creating Label 3 with style
label3 = Label(root, text="C", bg="blue", width=10)
label3.pack(side=LEFT, fill=Y, expand=1)  # Using pack to display and arrange Label 3 on the left

# Creating Label 4 with style
label4 = Label(root, text="D", bg="white", width=10)
label4.pack(side=RIGHT)  #  Using pack to display and arrange Label 4 on the right

# Starting the Tkinter window
root.mainloop()

# End marker
```

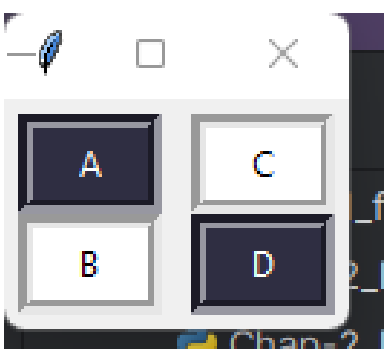
Output



Chap-2 Ex-2 b

```python
# Chapter 2 Exercise 2 b: Square Grid

# Importing tkinter library
import tkinter as tk

# Defining a function for creating square grids
1 usage
def creating_square_grid():
    root = tk.Tk()
    root.title("Square Grid with Labels")

    # Creating frames with a border of 5
    left_frame = tk.Frame(root, bd=5)
    right_frame = tk.Frame(root, bd=5)

    # Creating labels for the frames
    label_a = tk.Label(left_frame, text="A", padx=10, pady=5, bd=5, relief=tk.GROOVE, bg="#2f2e42", fg="white")
    label_b = tk.Label(left_frame, text="B", padx=10, pady=5, bd=5, relief=tk.GROOVE, bg="white")
    label_c = tk.Label(right_frame, text="C", padx=10, pady=5, bd=5, relief=tk.GROOVE, bg="white")
    label_d = tk.Label(right_frame, text="D", padx=10, pady=5, bd=5, relief=tk.GROOVE, bg="#2f2e42", fg="white")

    # Packing the labels with expand and fill options
    label_a.pack(side=tk.TOP, fill=tk.BOTH, expand=1)
    label_b.pack(side=tk.BOTTOM, fill=tk.BOTH, expand=1)
    label_c.pack(side=tk.TOP, fill=tk.BOTH, expand=1)
    label_d.pack(side=tk.BOTTOM, fill=tk.BOTH, expand=1)

    # Packing the frames with a border of 5, expand, and fill options
    left_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=1)
    right_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=1)

    # Running the mainloop
    root.mainloop()

# Call the function to create the square grid
creating_square_grid()
```

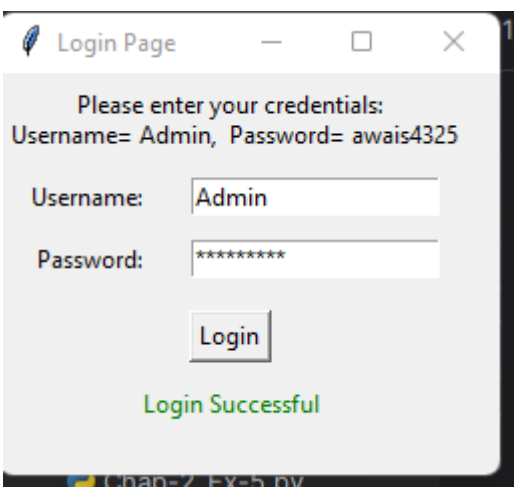
Output



Chap-2 Ex-3

```python
import tkinter as tk

1 usage
def login_page():
    user_name = entry_user_name.get()
    password = entry_password.get()

    # Basic authentication (replace with your actual authentication logic)
    if user_name == "Admin" and password == "awais4325":
        result_label.config(text="Login Successful", fg="green")
    else:
        result_label.config(text="Login Failed", fg="red")

root = tk.Tk()
root.title("Login Page")

# Set the initial size of the window
root.geometry("250x200")

# Adding descriptive labels above the username and password fields
label_instruction = tk.Label(root, text="Please enter your credentials:\n Username= Admin,  Password= awais4325")
label_user_name = tk.Label(root, text="Username:")
label_password = tk.Label(root, text="Password:")
entry_user_name = tk.Entry(root)
entry_password = tk.Entry(root, show="*")
login_button = tk.Button(root, text="Login", command=login_page)
result_label = tk.Label(root, text="", fg="black")

# Placing widgets in the grid
label_instruction.grid(row=0, column=0, columnspan=2, pady=5)
label_user_name.grid(row=1, column=0, padx=10, pady=5, sticky="e")
entry_user_name.grid(row=1, column=1, padx=10, pady=5)
label_password.grid(row=2, column=0, padx=10, pady=5, sticky="e")
entry_password.grid(row=2, column=1, padx=10, pady=5)
login_button.grid(row=3, column=0, columnspan=2, pady=10)
result_label.grid(row=4, column=0, columnspan=2)

# Running the mainloop
root.mainloop()
```

Output

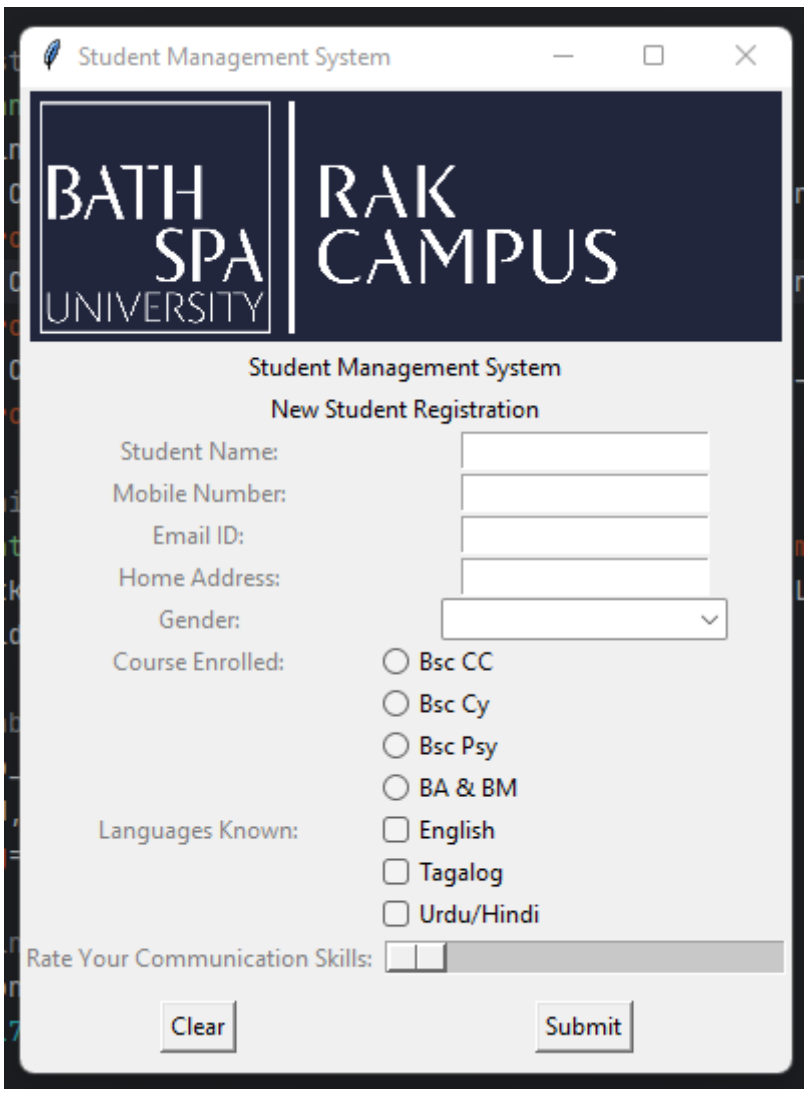


Chap-2 Ex-4

```python
# Chapter 2 Exercise 4: Registration page

# Importing tkinter library
import ...

# Defining function for data entry
1 usage
def submit_form():
    # Retrieving and printing the entered data
    print("Student Name:", entry_name.get())
    print("Mobile Number:", entry_mobile.get())
    print("Email ID:", entry_email.get())
    print("Home Address:", entry_address.get())
    print("Gender:", var_gender.get())
    print("Course Enrolled:", var_course.get())
    print("Languages Known:", var_languages.get())
    print("Communication Skills:", scale_communication.get())

# Defining the function for data clearing
1 usage
def clear_form():
    # Clear all entry fields and selections
    entry_name.delete( first: 0, tk.END)
    entry_mobile.delete( first: 0, tk.END)
    entry_email.delete( first: 0, tk.END)
    entry_address.delete( first: 0, tk.END)
    gender_choices.set("")  # Clear the combobox
    var_course.set("")  # Clear the radio buttons
    var_languages.set("")  # Clear the checkboxes
    scale_communication.set(0)  # Reset the scale

# Creating the main window
root = tk.Tk()
root.title("Student Management System")

# Codes for the university Picture
img = tk.PhotoImage(file='bath spa.png')
# Adjusting width and height using subsample
img = img.subsample( x: 4,  y: 4)  # Adjusting the values to control the size
label_picture = tk.Label(root, image=img)
label_picture.grid(row=0, column=0, columnspan=2)

# Text labels and entry widgets
tk.Label(root, text="Student Management System").grid(row=1, column=0, columnspan=2)
tk.Label(root, text="New Student Registration").grid(row=2, column=0, columnspan=2)

# Creating labels and entry widgets for student information

# Student Name
tk.Label(root, text="Student Name:").grid(row=3, column=0)
entry_name = tk.Entry(root)
entry_name.grid(row=3, column=1)

# Mobile Number
tk.Label(root, text="Mobile Number:").grid(row=4, column=0)
entry_mobile = tk.Entry(root)
entry_mobile.grid(row=4, column=1)

# Email ID
tk.Label(root, text="Email ID:").grid(row=5, column=0)
entry_email = tk.Entry(root)
```

```
# Email ID
tk.Label(root, text="Email ID:").grid(row=5, column=0)
entry_email = tk.Entry(root)
entry_email.grid(row=5, column=1)

# Home Address
tk.Label(root, text="Home Address:").grid(row=6, column=0)
entry_address = tk.Entry(root)
entry_address.grid(row=6, column=1)

# Gender
tk.Label(root, text="Gender:").grid(row=7, column=0)
var_gender = tk.StringVar()
gender_choices = ttk.Combobox(root, textvariable=var_gender, values=["Male", "Female"])
gender_choices.grid(row=7, column=1)

# Course Enrolled with the styling of the choices
tk.Label(root, text="Course Enrolled:").grid(row=8, column=0)
var_course = tk.StringVar()
course_choices = ttk.Radiobutton(root, text="Bsc CC", variable=var_course, value="Bsc CC")
course_choices.grid(row=8, column=1, sticky="w")
course_choices = ttk.Radiobutton(root, text="Bsc Cy", variable=var_course, value="Bsc Cy")
course_choices.grid(row=9, column=1, sticky="w")
course_choices = ttk.Radiobutton(root, text="Bsc Psy", variable=var_course, value="Bsc Psy")
course_choices.grid(row=10, column=1, sticky="w")
course_choices = ttk.Radiobutton(root, text="BA & BM", variable=var_course, value="BA & BM")
course_choices.grid(row=11, column=1, sticky="w")


# Languages Known and styling the choices
tk.Label(root, text="Languages Known:").grid(row=12, column=0)
var_languages = tk.StringVar()
language_choices = ttk.Checkbutton(root, text="English", variable=var_languages, onvalue="English", offvalue="")
language_choices.grid(row=12, column=1, sticky="w")
language_choices = ttk.Checkbutton(root, text="Tagalog", variable=var_languages, onvalue="Tagalog", offvalue="")
language_choices.grid(row=13, column=1, sticky="w")
language_choices = ttk.Checkbutton(root, text="Urdu/Hindi", variable=var_languages, onvalue="Urdu/Hindi", offvalue="")
language_choices.grid(row=14, column=1, sticky="w")

# label for Your Communication Skills
tk.Label(root, text="Rate Your Communication Skills:").grid(row=15, column=0)
scale_communication = tk.Scale(root, from_=0, to=10, orient=tk.HORIZONTAL, showvalue=0, length=200)
scale_communication.grid(row=15, column=1)

# Setting date entry labels to grey
for child in root.winfo_children():
    if isinstance(child, tk.Label) and child.cget("text").endswith(":"):
        child.config(fg="grey")

# Clearing and Submitting buttons
clear_button = tk.Button(root, text="Clear", command=clear_form)
clear_button.grid(row=17, column=0, pady=10)

submit_button = tk.Button(root, text="Submit", command=submit_form)
submit_button.grid(row=17, column=1, pady=10)

# Running the mainloop
root.mainloop()
```

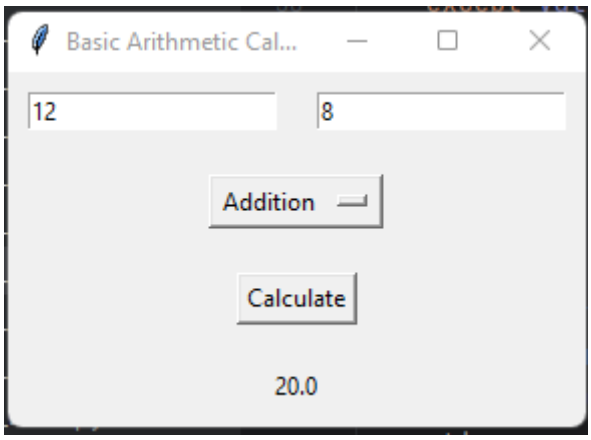
Output


Student Management System
BATH SPA UNIVERSITY
RAK CAMPUS
Student Management System
New Student Registration
Student Name:
Mobile Number:
Email ID:
Home Address:
Gender:
Course Enrolled:
Bsc CC
Bsc Cy
Bsc Psy
BA & BM
Languages Known:
English
Tagalog
Urdu/Hindi
Rate Your Communication Skills:
Clear
Submit

Chap-2 Ex-5

```python
# Chapter 2 Exercise 5: Calculator

# Importing the tkinter library
import tkinter as tk

# Defining the function for performing operations
1 usage
def arithmetic_operations():
    # Using Try block to handle potential errors during calculation
    try:
        # Retrieving numerical inputs and selected operator
        first_number = float(enter_first_number.get())
        second_number = float(enter_second_number.get())
        operator = operation_var.get()

        # Performing the selected operation and updating the result
        if operator == "Addition":
            result.set(first_number + second_number)
        elif operator == "Subtraction":
            result.set(first_number - second_number)
        elif operator == "Multiplication":
            result.set(first_number * second_number)
        elif operator == "Division":
            result.set(first_number / second_number)
        elif operator == "Modulo":
            result.set(first_number % second_number)
        else:
            result.set("Invalid Operation")

    # Handling the ValueError that may occur if non-numeric input is provided
    except ValueError:
        result.set("Invalid Input")

# Creating the main window
root = tk.Tk()
root.title("Basic Arithmetic Calculator")

# Using entry function for user input, operation, and result
enter_first_number = tk.Entry(root, width=20)
enter_second_number = tk.Entry(root, width=20)
operation_var = tk.StringVar(value="Addition")
result = tk.StringVar()

# Styling the widgets
tk.OptionMenu(root, operation_var, value: "Addition", *values: "Subtraction", "Multiplication", "Division", "Modulo").grid(row=1, column=0, columnspan=2,
tk.Button(root, text="Calculate", command=arithmetic_operations).grid(row=2, column=0, columnspan=2, padx=10, pady=10)
tk.Label(root, textvariable=result).grid(row=3, column=0, columnspan=2, padx=10, pady=10)

# Arranging the entry widgets
enter_first_number.grid(row=0, column=0, padx=10, pady=10)
enter_second_number.grid(row=0, column=1, padx=10, pady=10)

# Running the mainloop
root.mainloop()
```

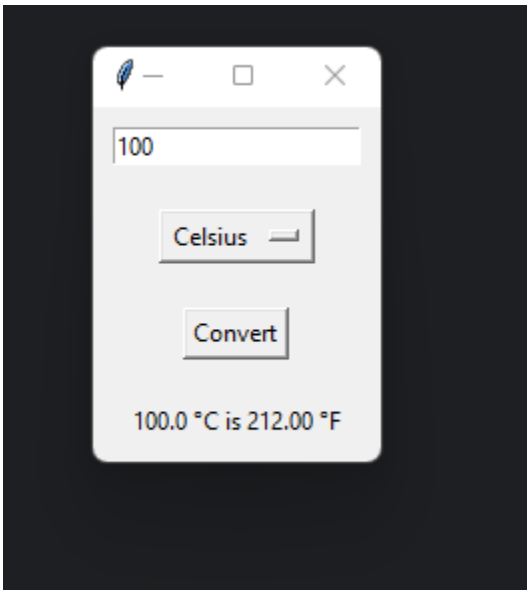
Output



Chap 2 Bonus A

```python
# Chapter 2 Bonus A: Temperature Converter

# Importing tkinter library
import tkinter as tk

# Defining function for converting temperature
1 usage
def temperature_converter():
    try:
        # Trying to convert input temperature to a float and getting the user selected temperature unit
        temp = float(entered_temperature.get())  # Getting temperature from the entry widget and convert it to a float
        temp_type = temperature_var.get()  # Getting selected temperature unit

        # Checking the selected unit and perform the corresponding temperature conversion
        if temp_type == "Celsius":
            # Converting Celsius to Fahrenheit and setting the result string with 2 decimal places
            result.set(f"{temp} °C is {((temp * 9 / 5) + 32):.2f} °F")
        elif temp_type == "Fahrenheit":
            # Converting Fahrenheit to Celsius and setting the result string with 2 decimal places
            result.set(f"{temp} °F is {((temp - 32) * 5 / 9):.2f} °C")

    except ValueError:
        # Handling the case where the input cannot be converted to a float and output "invalid input"
        result.set("Invalid Input")

# Creating the main window
root = tk.Tk()
root.title("Temperature Converter")

# Variables for user input, unit, and result
entered_temperature = tk.Entry(root, width=20)
temperature_var = tk.StringVar(value="Celsius")
result = tk.StringVar()

# Styling and writing Widgets
tk.OptionMenu(root, temperature_var, value: "Celsius", *values: "Fahrenheit").grid(row=1, column=0, columnspan=2, padx=10, pady=10)
tk.Button(root, text="Convert", command=temperature_converter).grid(row=2, column=0, columnspan=2, padx=10, pady=10)
tk.Label(root, textvariable=result).grid(row=3, column=0, columnspan=2, padx=10, pady=10)

# Arrange entry widget
entered_temperature.grid(row=0, column=0, padx=10, pady=10)

# Running the main loop
root.mainloop()
```

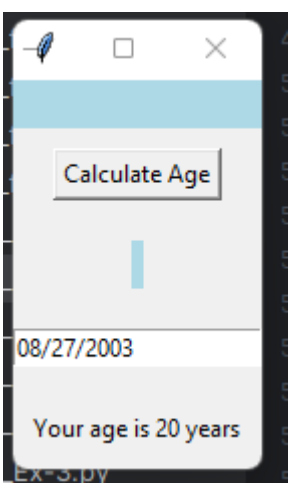
Output



Chap-2 Bonus B

```python
# Chapter 2 Bonus B: Age Calculator

# Importing date and time from datetime library
import ...

# Defining function for handling every click event
1 usage
def entry_click(event):
    if enter_date.get() == 'MM/DD/YYYY':
        enter_date.delete( first: 0, last: "end")  # Clearing the default text
        enter_date.config(fg='black')  # Changing text color to black

# Defining function for calculating the age
1 usage
def age_calculation():
    try:
        # Getting the user input for date of birth
        dob_str = enter_date.get()

        # Checking if the default text is entered
        if dob_str == 'MM/DD/YYYY':
            raise ValueError("Invalid Input")

        d_o_b = datetime.strptime(dob_str, _format: "%m/%d/%Y")

        # Codes for calculating the age
        today = datetime.today()
        age = today.year - d_o_b.year - ((today.month, today.day) < (d_o_b.month, d_o_b.day))

        # Displaying the final result
        result.set(f"Your age is {age} years")
        text_label.config(text="")  # Remove the text "How are you?"


    # Displaying the error message if data not entered per the specification
    except ValueError:
        result.set("Invalid Input")
        text_label.config(text="")


# Creating the the main window
root = tk.Tk()
root.title("Age Calculator")

# Variables for input from the user and result
enter_date = tk.Entry(root, width=20, fg='grey')
enter_date.insert( index: 0, string: 'MM/DD/YYYY')  # Setting default text
enter_date.bind("<FocusIn>", entry_click)  # Binding the click event
result = tk.StringVar()

# Creating a label to cover the entire background
background_label = tk.Label(
    root,
    text="",  # Empty text
    font=("Helvetica", 12),
    bg="lightblue",  # Setting background color
)
background_label.pack(expand=True, fill='both')  # Expand to fill the available space

# styling the Widgets
tk.Button(root, text="Calculate Age", command=age_calculation).pack(pady=10)
text_label = tk.Label(
    root,
    text="Please! Enter Your Age in the given Format.\n 'MM/DD/YYYY'",
    font=("Helvetica", 12),
    bg="lightblue",
)
text_label.pack(pady=10)
enter_date.pack(pady=10)
tk.Label(root, textvariable=result).pack(pady=10)

# Running the main loop
root.mainloop()
```

Output



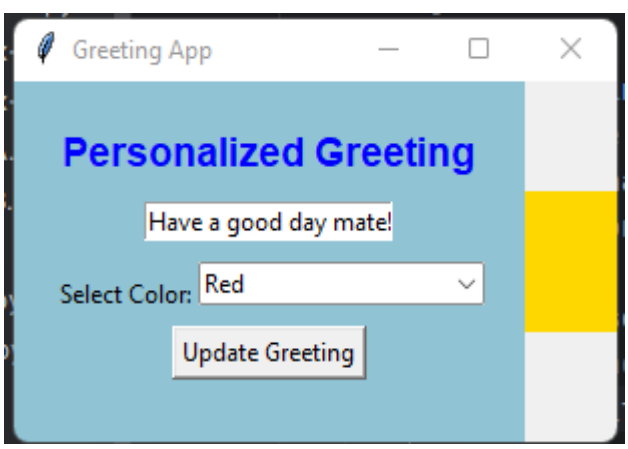
Chapter 3

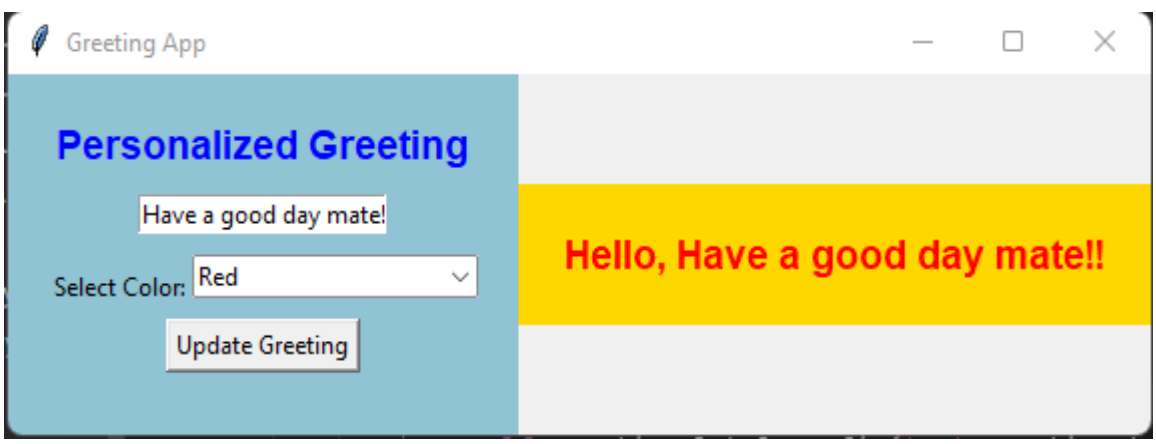
Graphical User Interface(cont.)

Chap-3 Ex-1

```python
# Chapter 3 Exercise 1: Greeting App

# Importing tk inter library and modules
import tkinter as tk
from tkinter import ttk

# making a class for Greetings
1 usage
class GreetingApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Greeting App")

        # Setting the background colors
        input_frame_color = "#90C3D4"  # Light blue
        display_frame_color = "#FFD700"  # Gold

        # Creating the InputFrame
        self.input_frame = tk.Frame(root, bg=input_frame_color, padx=20, pady=20)
        self.input_frame.grid(row=0, column=0)

        # Titling the label in blue
        title_label = tk.Label(self.input_frame, text="Personalized Greeting", font=("Helvetica", 14, "bold"), fg="blue", bg=input_frame_color)
        title_label.grid(row=0, column=0, columnspan=2, pady=(0, 10))

        # Making an Entry field for the user's name
        self.name_entry = tk.Entry(self.input_frame)
        self.name_entry.grid(row=1, column=0, columnspan=2, pady=(0, 10))

        # Making a dropdown menu for selecting color
        color_label = tk.Label(self.input_frame, text="Select Color:", bg=input_frame_color)
```

Output

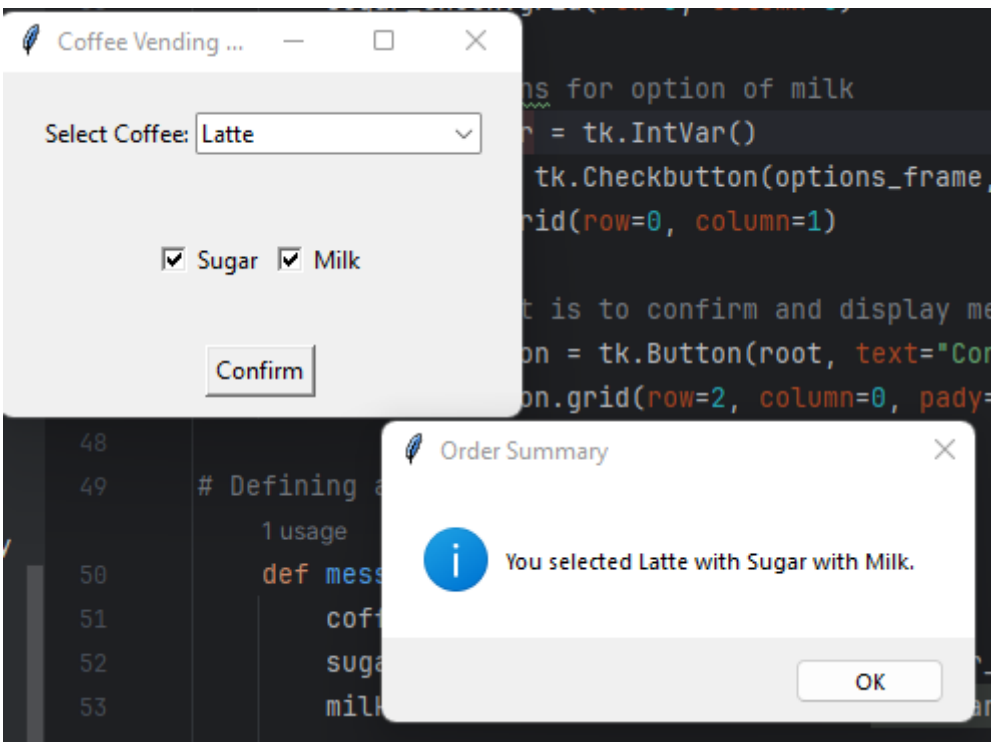




Chap-3 Ex-2

```python
# Chapter 3 Exercise 2: Coffee Vending Machine

# Importing tkinter library and modules
import tkinter as tk
from tkinter import ttk, messagebox

# Making a class for the coffee machine
1 usage
class Coffee_Vending_Machine:
    def __init__(self, root):
        self.root = root
        self.root.title("Coffee Vending Machine")

        # Displaying the message for options that would be given for the Coffee
        self.coffee_var = tk.StringVar()
        self.coffee_var.set("Select type of Coffee")

        # Adding the coffee options
        coffee_options = ["Espresso", "Cappuccino", "Latte", "Americano"]

        # Creating the Coffee Frame
        coffee_frame = tk.Frame(root, padx=20, pady=20)
        coffee_frame.grid(row=0, column=0)

        # Labeling and Dropdown for Coffee selection
        coffee_label_text = tk.Label(coffee_frame, text="Select Coffee:")
        coffee_label_text.grid(row=0, column=0)

        coffee_dropdown = ttk.Combobox(coffee_frame, textvariable=self.coffee_var, values=coffee_options)
        coffee_dropdown.grid(row=0, column=1)

        # Creating Options Frame
        options_frame.grid(row=1, column=0)

        # Checkbuttons for option of sugar
        self.sugar_var = tk.IntVar()
        sugar_check = tk.Checkbutton(options_frame, text="Sugar", variable=self.sugar_var)
        sugar_check.grid(row=0, column=0)

        # Checkbuttons for option of milk
        self.milk_var = tk.IntVar()
        milk_check = tk.Checkbutton(options_frame, text="Milk", variable=self.milk_var)
        milk_check.grid(row=0, column=1)

        # Button that is to confirm and display message
        confirm_button = tk.Button(root, text="Confirm", command=self.message_displayed)
        confirm_button.grid(row=2, column=0, pady=10)

# Defining a function for displaying the message
    1 usage
    def message_displayed(self):
        coffee_choice = self.coffee_var.get()
        sugar_choice = "with Sugar" if self.sugar_var.get() else "without Sugar"
        milk_choice = "with Milk" if self.milk_var.get() else "without Milk"

        message = f"You selected {coffee_choice} {sugar_choice} {milk_choice}."

        # Displaying the message using messagebox
        messagebox.showinfo( title: "Order Summary", message)

# And making sure it runs properly
if __name__ == "__main__":
    root = tk.Tk()
    app = Coffee_Vending_Machine(root)
    root.mainloop() # With closing the mainloop
```

Output

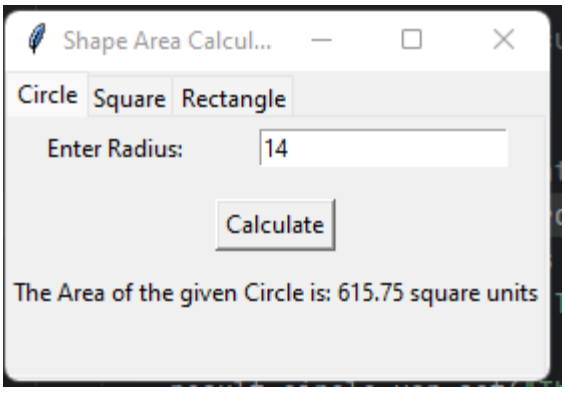

Coffee Vending ...
Select Coffee:
Latte
Sugar
Milk
Confirm
Order Summary
You selected Latte with Sugar with Milk.
OK

Chap-3 Ex-3

```
# Chapter 3 Exercise 3: Area Function

# Importing tkinter library and its module
import ...

# Defining function for the calculation of area of circle
1 usage
def calculate_circle_area():
    try: # Codes for the calculation of area of circle
        radius = float(entry_circle_radius.get())
        area = math.pi * (radius ** 2)
        result_circle_var.set(f"The Area of the given Circle is: {area:.2f} square units")
    except ValueError:
        result_circle_var.set("The Input is Invalid. Please enter a valid number.")

# Defining function for the calculation of area of square
1 usage
def calculate_square_area():
    try: # Codes for the calculation of area of square
        side_length = float(entry_square_side.get())
        area = side_length ** 2
        result_square_var.set(f"The Area of the given Square ia: {area:.2f} square units")
    except ValueError:
        result_square_var.set("The Input is Invalid. Please enter a valid number.")

# Defining function for the calculation of area of rectangle
1 usage
def calculate_rectangle_area():
    try: # Codes for the calculation of area of rectangle
        length = float(entry_rectangle_length.get())
        width = float(entry_rectangle_width.get())
        area = length * width
        result_rectangle_var.set(f"The Area of the given Rectangle is: {area:.2f} square units")
    except ValueError:
        result_rectangle_var.set("The Input is Invalid. Please enter valid numbers.")

# Creating the main window
root = tk.Tk()
root.title("Shape Area Calculator")

# Creating a notebook
notebook = ttk.Notebook(root)

# Creating the tabs for Circle, Square, and Rectangle
tab_circle = ttk.Frame(notebook)
tab_square = ttk.Frame(notebook)
tab_rectangle = ttk.Frame(notebook)

# Notebook output for Circle
notebook.add(tab_circle, text="Circle")
# Notebook output for square
notebook.add(tab_square, text="Square")
# Notebook output for rectangle
notebook.add(tab_rectangle, text="Rectangle")

notebook.pack(fill="both", expand=True)
# The Tab for circle
# Labels and entry for circle radius
label_circle_radius = tk.Label(tab_circle, text="Enter Radius:")
entry_circle_radius = tk.Entry(tab_circle)
button_calculate_circle = tk.Button(tab_circle, text="Calculate", command=calculate_circle_area)
result_circle_var = tk.StringVar()
label_result_circle = tk.Label(tab_circle, textvariable=result_circle_var)

# Grid layout for circle labels, entry, button, and result
label_circle_radius.grid(row=0, column=0, padx=10, pady=5)
```

```
entry_circle_radius.grid(row=0, column=1, padx=10, pady=5)
button_calculate_circle.grid(row=1, column=0, columnspan=2, pady=10)
label_result_circle.grid(row=2, column=0, columnspan=2)

# The Tab for square
# Labels and entry for square side length
label_square_side = tk.Label(tab_square, text="Enter Side Length:")
entry_square_side = tk.Entry(tab_square)
button_calculate_square = tk.Button(tab_square, text="Calculate", command=calculate_square_area)
result_square_var = tk.StringVar()
label_result_square = tk.Label(tab_square, textvariable=result_square_var)

# Grid layout for square labels, entry, button, and result
label_square_side.grid(row=0, column=0, padx=10, pady=5)
entry_square_side.grid(row=0, column=1, padx=10, pady=5)
button_calculate_square.grid(row=1, column=0, columnspan=2, pady=10)
label_result_square.grid(row=2, column=0, columnspan=2)

# The Tab for rectangle
# Labels and entry for rectangle length and width
label_rectangle_length = tk.Label(tab_rectangle, text="Enter Length:")
label_rectangle_width = tk.Label(tab_rectangle, text="Enter Width:")
entry_rectangle_length = tk.Entry(tab_rectangle)
entry_rectangle_width = tk.Entry(tab_rectangle)
button_calculate_rectangle = tk.Button(tab_rectangle, text="Calculate", command=calculate_rectangle_area)
result_rectangle_var = tk.StringVar()
label_result_rectangle = tk.Label(tab_rectangle, textvariable=result_rectangle_var)

# Grid layout for rectangle labels, entry, button, and result
label_rectangle_length.grid(row=0, column=0, padx=10, pady=5)
entry_rectangle_length.grid(row=0, column=1, padx=10, pady=5)
label_rectangle_width.grid(row=1, column=0, padx=10, pady=5)
entry_rectangle_width.grid(row=1, column=1, padx=10, pady=5)
button_calculate_rectangle.grid(row=2, column=0, columnspan=2, pady=10)
label_result_rectangle.grid(row=3, column=0, columnspan=2)
# Run the Tkinter event loop
root.mainloop()
```

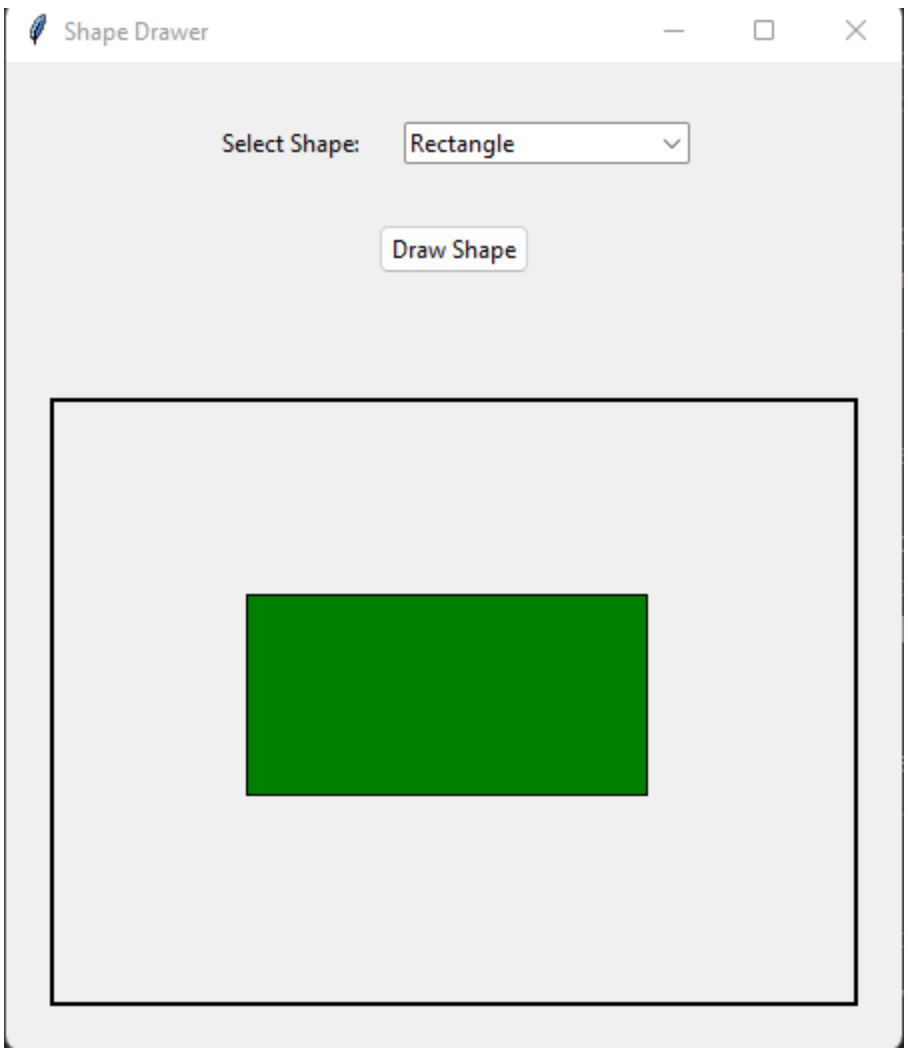
Output



Chap-3 Ex-4

```python
# Chapter 3 Exercise 4: Draw Shape

# Importing tkinter library and modules
import ...

# Defining a function for drawing the shape
1 usage
def draw_shape():
    canvas.delete("all")  # Clearing the canvas
    # Get the selected shape from the dropdown menu
    shape_selected = shape_var.get()

    # Drawing the selected shape on the canvas
    if shape_selected == "Oval":
        # Drawing an oval with specified coordinates, outline, and fill color
        canvas.create_oval(100, 100, 300, 200, outline="black", fill="red")
    elif shape_selected == "Rectangle":
        # Drawing a rectangle with specified coordinates, outline, and fill color
        canvas.create_rectangle(100, 100, 300, 200, outline="black", fill="green")
    elif shape_selected == "Square":
        # Drawing a square with specified coordinates, outline, and fill color
        canvas.create_rectangle(100, 100, 200, 200, outline="black", fill="blue")
    elif shape_selected == "Triangle":
        # Drawing a triangle with specified coordinates, outline, and fill color
        canvas.create_polygon(100, 200, 200, 100, 300, 200, outline="black", fill="yellow")

# Creating the main window
root = tk.Tk()
root.title("Shape Drawer")

# Creating a frame to hold the controls
frame = ttk.Frame(root, padding=20)
frame.grid(row=0, column=0)

# Creating a Label and Combobox for shape selection
label_shape = ttk.Label(frame, text="Select Shape:")
label_shape.grid(row=0, column=0, padx=10, pady=10)

# Setting the shapes
shapes = ["Oval", "Rectangle", "Square", "Triangle"]
shape_var = tk.StringVar()
shape_combobox = ttk.Combobox(frame, textvariable=shape_var, values=shapes)
shape_combobox.grid(row=0, column=1, padx=10, pady=10)
shape_combobox.set(shapes[0])  # Set the default shape

# Creating a button to draw the selected shape
draw_button = ttk.Button(frame, text="Draw Shape", command=draw_shape)
draw_button.grid(row=1, column=0, columnspan=2, pady=20)

# Creating a canvas to draw the shapes
canvas = tk.Canvas(root, width=400, height=300, borderwidth=2, relief="solid")
canvas.grid(row=1, column=0, padx=20, pady=20)

# Running the Tkinter event loop
root.mainloop()
```

Output



Chap-3 Bonus

```
# Chap 3 Bonus Ex: Burger Shack Vendor


# Defining Function for displaying the menu
1 usage
def display_menu():
    # Displaying the welcome message and the menu options
    print("Welcome to Burger Shack!")
    print("1. Burger Types: Beef, Chicken, Vegetarian")
    print("2. Toppings: Cheese, Peanut Butter, Avocado")
    print("3. Condiments: Ketchup, Mayonnaise, BBQ Sauce")
    print("4. Sides: Fries, Drink")

# Defining function for placing order
1 usage
def place_order():
    # Initializing an empty order dictionary
    order = {
        "burger_type": None,
        "toppings": [],
        "condiments": [],
        "sides": []
    }

    try:
        # Choosing burger type
        burger_choice = int(input("Select Burger Type (1-3): "))
        burger_types = ["Beef", "Chicken", "Vegetarian"]
        order["burger_type"] = burger_types[burger_choice - 1]

        # Adding toppings
        print("\nChoose Toppings (Enter 0 to finish):")
        toppings = ["Cheese", "Peanut Butter", "Avocado"]
        while True:
            topping_choice = int(input("   Topping (1-3): "))
            if topping_choice == 0:
                break
            order["toppings"].append(toppings[topping_choice - 1])

        # Adding condiments
        print("\nChoose Condiments (Enter 0 to finish):")
        condiments = ["Ketchup", "Mayonnaise", "BBQ Sauce"]
        while True:
            condiment_choice = int(input("   Condiment (1-3): "))
            if condiment_choice == 0:
                break
            order["condiments"].append(condiments[condiment_choice - 1])

        # Adding sides
        print("\nChoose Sides (Enter 0 to finish):")
        sides = ["Fries", "Drink"]
        while True:
            side_choice = int(input("   Side (1-2): "))
            if side_choice == 0:
                break
            order["sides"].append(sides[side_choice - 1])
```

```python
def calculate_total(order):
    # Assuming the fixed costs for items
    burger_cost = 5.00
    topping_cost = 1.50
    condiment_cost = 1.00
    side_cost = 2.50

    # Calculating the total cost based on the order
    total_cost = burger_cost + len(order["toppings"]) * topping_cost + len(order["condiments"]) * condiment_cost \
                 + len(order["sides"]) * side_cost

    return total_cost


1 usage
def process_payment(total_cost):
    try:
        # Getting the amount paid from the user
        amount_paid = float(input(f"\nTotal Cost: {total_cost:.2f} AED\nEnter Amount Paid: "))

        # Checking if the payment is sufficient and provide change if needed
        if amount_paid >= total_cost:
            change = amount_paid - total_cost
            print(f"Payment successful! Change: {change:.2f} AED")
        else:
            print("Insufficient payment. Please provide enough funds.")
    except ValueError:
        print("Invalid input. Please enter a valid amount.")


def main():
    # Executing the main ordering process
    display_menu()
    order = place_order()
    total_cost = calculate_total(order)
    process_payment(total_cost)


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

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe "C:\Users\work\PycharmProjects\pythonProject1\Chap-3_Bonus Exercise.py"
Welcome to Burger Shack!
1. Burger Types: Beef, Chicken, Vegetarian
2. Toppings: Cheese, Peanut Butter, Avocado
3. Condiments: Ketchup, Mayonnaise, BBQ Sauce
4. Sides: Fries, Drink
Select Burger Type (1-3): 1

Choose Toppings (Enter 0 to finish):
   Topping (1-3): 1
   Topping (1-3): 2
   Topping (1-3): 0

Choose Condiments (Enter 0 to finish):
   Condiment (1-3): 3
   Condiment (1-3): 2
   Condiment (1-3): 0

Choose Sides (Enter 0 to finish):
   Side (1-2): 2
   Side (1-2): 0

Total Cost: 12.50 AED
Enter Amount Paid:
```

Chapter 4

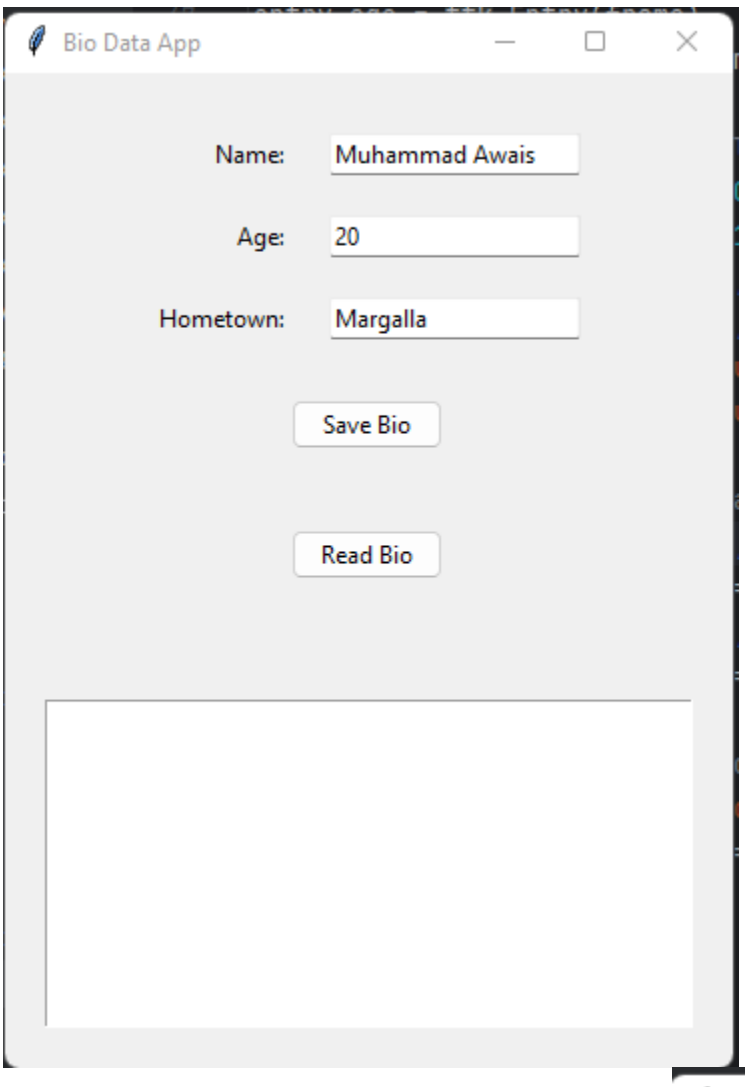
File Handling and Regular Expressions

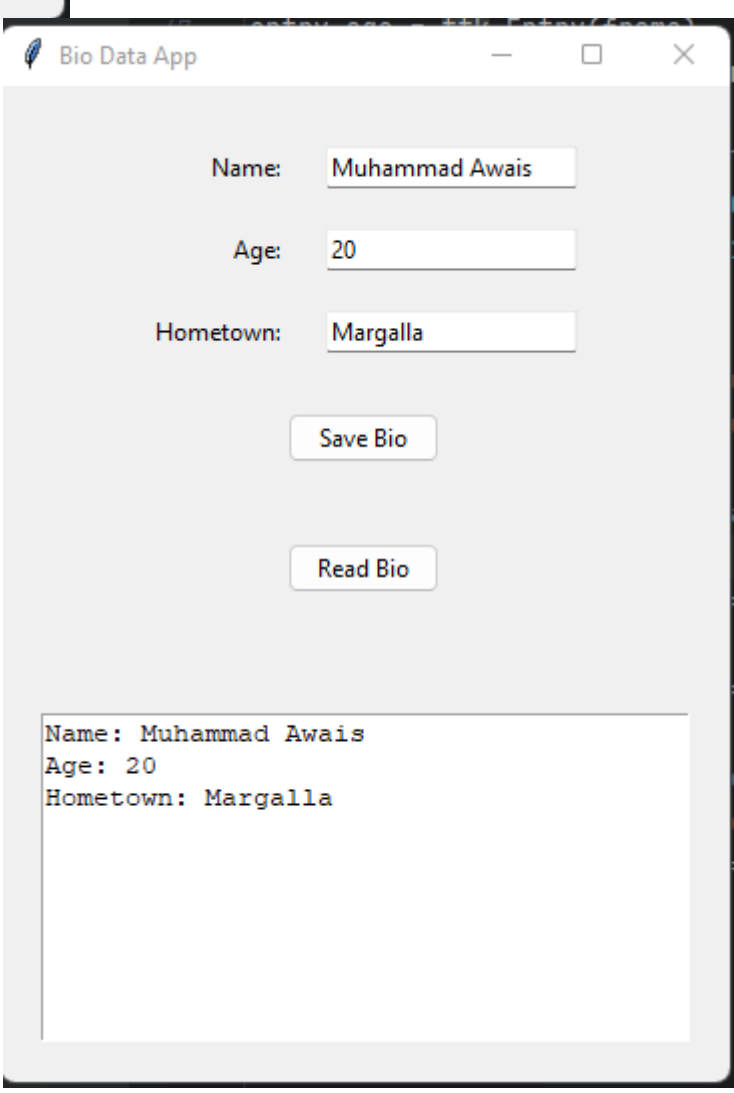
Chap-4 Ex-1

```
# Chapter 4 Exercise 1: User information

# Importing tkinter library and its modules
import ...

# Defining a function for saving the data
1 usage
def save_info():
    # Geting user input
    name = entry_name.get()
    age = entry_age.get()
    hometown = entry_hometown.get()

    # Write data to the file
    file_path = "bio.txt"
    with open(file_path, "w") as file:
        file.write(f"Name: {name}\n")
        file.write(f"Age: {age}\n")
        file.write(f"Hometown: {hometown}\n")

# Defining function for bio data input
1 usage
def read_info():
    # Reading data from the file
    file_path = "bio.txt"
    try:
        with open(file_path, "r") as file:
            bio_data = file.read()
        text_output.delete( index1: 1.0, tk.END)  # Clearing the previous output
        text_output.insert(tk.END, bio_data)
    except FileNotFoundError: # If data not entered show the error
        text_output.delete( index1: 1.0, tk.END)
        text_output.insert(tk.END,  chars: "Entered Bio file not found. Please save your bio first.")
```

```
# Creating the main window
root = tk.Tk()
root.title("Bio Data App")

# Creating a frame to hold the controls
frame = ttk.Frame(root, padding=20)
frame.grid(row=0, column=0)

# Creating the entry widgets for user input
label_name = ttk.Label(frame, text="Name:")
label_age = ttk.Label(frame, text="Age:")
label_hometown = ttk.Label(frame, text="Hometown:")
entry_name = ttk.Entry(frame)
entry_age = ttk.Entry(frame)
entry_hometown = ttk.Entry(frame)

# Setting the grid layout for labels and entry widgets
label_name.grid(row=0, column=0, padx=10, pady=10, sticky="e")
entry_name.grid(row=0, column=1, padx=10, pady=10)
label_age.grid(row=1, column=0, padx=10, pady=10, sticky="e")
entry_age.grid(row=1, column=1, padx=10, pady=10)
label_hometown.grid(row=2, column=0, padx=10, pady=10, sticky="e")
entry_hometown.grid(row=2, column=1, padx=10, pady=10)

# Creating buttons for saving and reading bio
button_save = ttk.Button(frame, text="Save Bio", command=save_info)
button_save.grid(row=3, column=0, columnspan=2, pady=20)
button_read = ttk.Button(frame, text="Read Bio", command=read_info)
button_read.grid(row=4, column=0, columnspan=2, pady=20)

# Creating the text widget to display bio data
text_output = tk.Text(root, width=40, height=10)
text_output.grid(row=1, column=0, padx=20, pady=20)

# Running the Tkinter loop
root.mainloop()
```

Output


Bio Data App
Name:
Muhammad Awais
Age:
20
Hometown:
Margalla
Save Bio
Read Bio

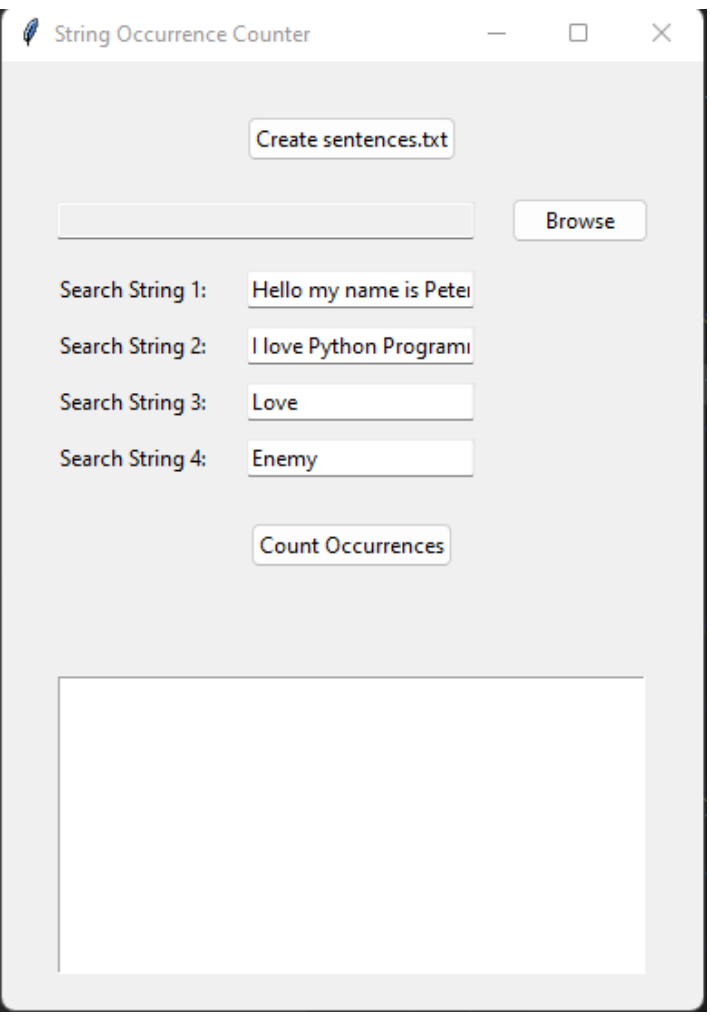

Bio Data App
Name:
Muhammad Awais
Age:
20
Hometown:
Margalla
Save Bio
Read Bio
Name: Muhammad Awais
Age: 20
Hometown: Margalla

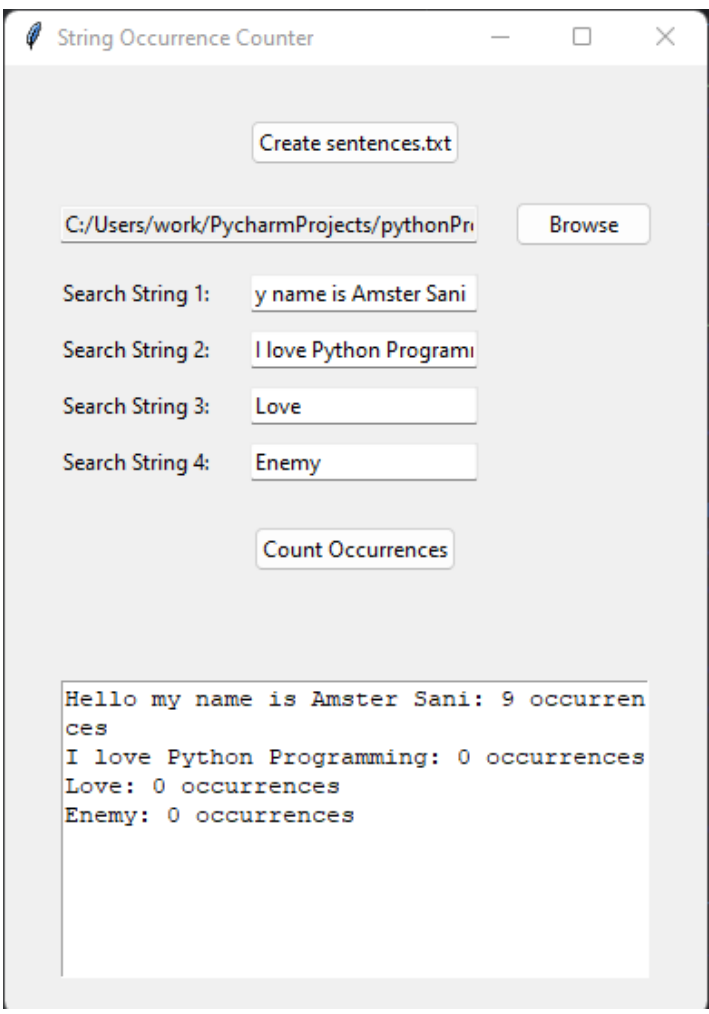
Chap-4 Ex-2

```
# Chapter 4 Exercise 2: CountString


# Importing tkinter library and its modules
import ...

# Defining a function for the sentences files
1 usage
def create_sentences_file():
    # Creating a file named sentences.txt with sample content
    file_path = "sentences.txt"
    with open(file_path, "w") as file:
        file.write("Hello my name is Peter Parker\n")
        file.write("I love Python Programming\n")
        file.write("Love\n")
        file.write("Enemy\n")
    file_var.set(file_path)

# Defining a function for the counting of count occurence
1 usage
def count_occurrences():
    # Getting the selected file
    file_path = file_var.get()

    # Reading the content of the file
    try:
        with open(file_path, "r") as file:
            content = file.read()

        # Getting the search strings
        search_strings = [entry.get() for entry in entry_list]

        # Counting occurrences for each search string
        result = {search_string: content.count(search_string) for search_string in search_strings}

        # Displaying the results
        text_output.delete( index1: 1.0, tk.END)
        for search_string, count in result.items():
            text_output.insert(tk.END, chars: f"{search_string}: {count} occurrences\n")
    # Showing error if file not found
    except FileNotFoundError:
        text_output.delete( index1: 1.0, tk.END)
        text_output.insert(tk.END, chars: "File not found. Please create sentences.txt first.")

# Creating the main window
root = tk.Tk()
root.title("String Occurrence Counter")

# Creating a frame to hold the controls
frame = ttk.Frame(root, padding=20)
frame.grid(row=0, column=0)

# Creating a button to create the sentences.txt file
create_button = ttk.Button(frame, text="Create sentences.txt", command=create_sentences_file)
create_button.grid(row=0, column=0, columnspan=3, pady=10)

# Creating an entry for file path
file_var = tk.StringVar()
file_entry = ttk.Entry(frame, textvariable=file_var, state="readonly")
file_entry.grid(row=1, column=0, columnspan=2, padx=10, pady=10, sticky="ew")

# Creating a button to browse for a file
browse_button = ttk.Button(frame, text="Browse", command=lambda: browse_file("sentences.txt"))
browse_button.grid(row=1, column=2, padx=10, pady=10)

# Creating entry widgets for search strings
```

```python
# Creating entry widgets for search strings
entry_list = []
search_strings = ["Hello my name is Peter Parker", "I love Python Programming", "Love", "Enemy"]
for i, search_string in enumerate(search_strings):
    ttk.Label(frame, text=f"Search String {i+1}:").grid(row=i+2, column=0, padx=10, pady=5, sticky="e")
    entry = ttk.Entry(frame)
    entry.insert(tk.END, search_string)
    entry.grid(row=i+2, column=1, padx=10, pady=5)
    entry_list.append(entry)

# Creating a button to count occurrences
count_button = ttk.Button(frame, text="Count Occurrences", command=count_occurrences)
count_button.grid(row=len(search_strings)+2, column=0, columnspan=3, pady=20)

# Creating a text widget to display results
text_output = tk.Text(root, width=40, height=10)
text_output.grid(row=1, column=0, padx=20, pady=20)

# Defining a function for browsing for a file
1 usage
def browse_file(default_filename):
    file_path = filedialog.askopenfilename(filetypes=[("Text Files", "*.txt")], initialfile=default_filename)
    file_var.set(file_path)

# Running the Tkinter event loop
root.mainloop()
```

Output





Chap-4 Ex-3

Code and Output

```python
# Chapter 4 Exercise 3: Reading to a List

# Opening the file in read mode
with open('numbers.txt', 'r') as file:
    # Reading all lines from the file and converting each line to an integer
    numbers = [int(line.strip()) for line in file]

# Outputting the values in integer format
for number in numbers:
    print(number)
```

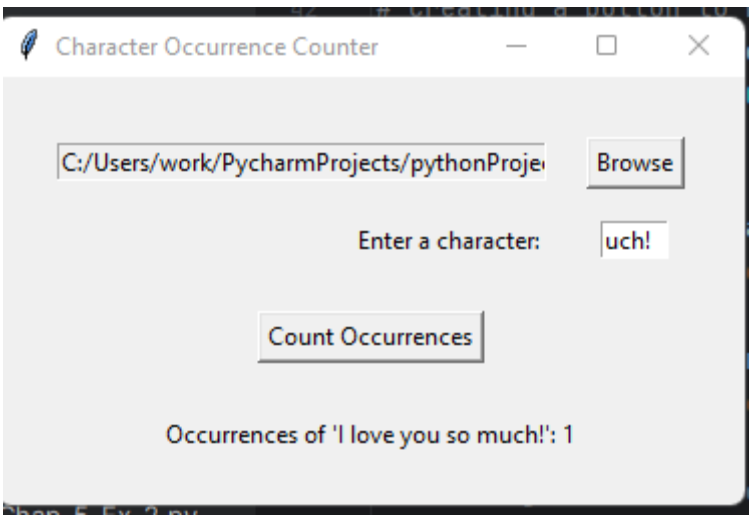
Chap-4_Ex-3

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-4_Ex-3.
58
19
4
46
52
33
36
96
15
97
37
42
88
83
62
91
79
25
1
10
```

Chap-4 Ex-4

```
# Chapter 4 Exercise 4: letter count

# Importing tkinter library and its modules
import ...

# Defining function for the amount of number occurrence
1 usage
def count_occurrences():
    # Getting the selected file
    file_path = file_var.get()

    # Reading the content of the file
    try:
        with open(file_path, "r") as file:
            content = file.read()

        # Getting the character entered by the user
        char_to_count = char_entry.get()

        # Counting occurrences of the character
        char_count = content.count(char_to_count)

        # Displaying the result
        result_label.config(text=f"Occurrences of '{char_to_count}': {char_count}")

    except FileNotFoundError:
        result_label.config(text="File not found. Please select a valid file.")

# Creating the main window
root = tk.Tk()
root.title("Character Occurrence Counter")

# Creating a frame to hold the controls
frame = tk.Frame(root, padx=20, pady=20)
frame.pack()

# Creating an entry for file path
file_var = tk.StringVar()
file_entry = tk.Entry(frame, textvariable=file_var, state="readonly", width=40)
file_entry.grid(row=0, column=0, padx=10, pady=10)

# Creating a button to browse for a file
browse_button = tk.Button(frame, text="Browse", command=lambda: browse_file("sentences.txt"))
browse_button.grid(row=0, column=1, padx=10, pady=10)

# Creating an entry for the character
char_entry_label = tk.Label(frame, text="Enter a character:")
char_entry_label.grid(row=1, column=0, padx=10, pady=5, sticky="e")

char_entry = tk.Entry(frame, width=5)
char_entry.grid(row=1, column=1, padx=10, pady=5)

# Creating a button to count occurrences
count_button = tk.Button(frame, text="Count Occurrences", command=count_occurrences)
count_button.grid(row=2, column=0, columnspan=2, pady=20)

# Creating a label to display the result
result_label = tk.Label(frame, text="")
result_label.grid(row=3, column=0, columnspan=2, pady=5)

# Defining a function to browse for a file
1 usage
def browse_file(default_filename):
    file_path = filedialog.askopenfilename(filetypes=[("Text Files", "*.txt")], initialfile=default_filename)
    file_var.set(file_path)

# Defining a function to browse for a file
1 usage
def browse_file(default_filename):
    file_path = filedialog.askopenfilename(filetypes=[("Text Files", "*.txt")], initialfile=default_filename)
    file_var.set(file_path)

# Running the Tkinter event loop
root.mainloop()
```

Output

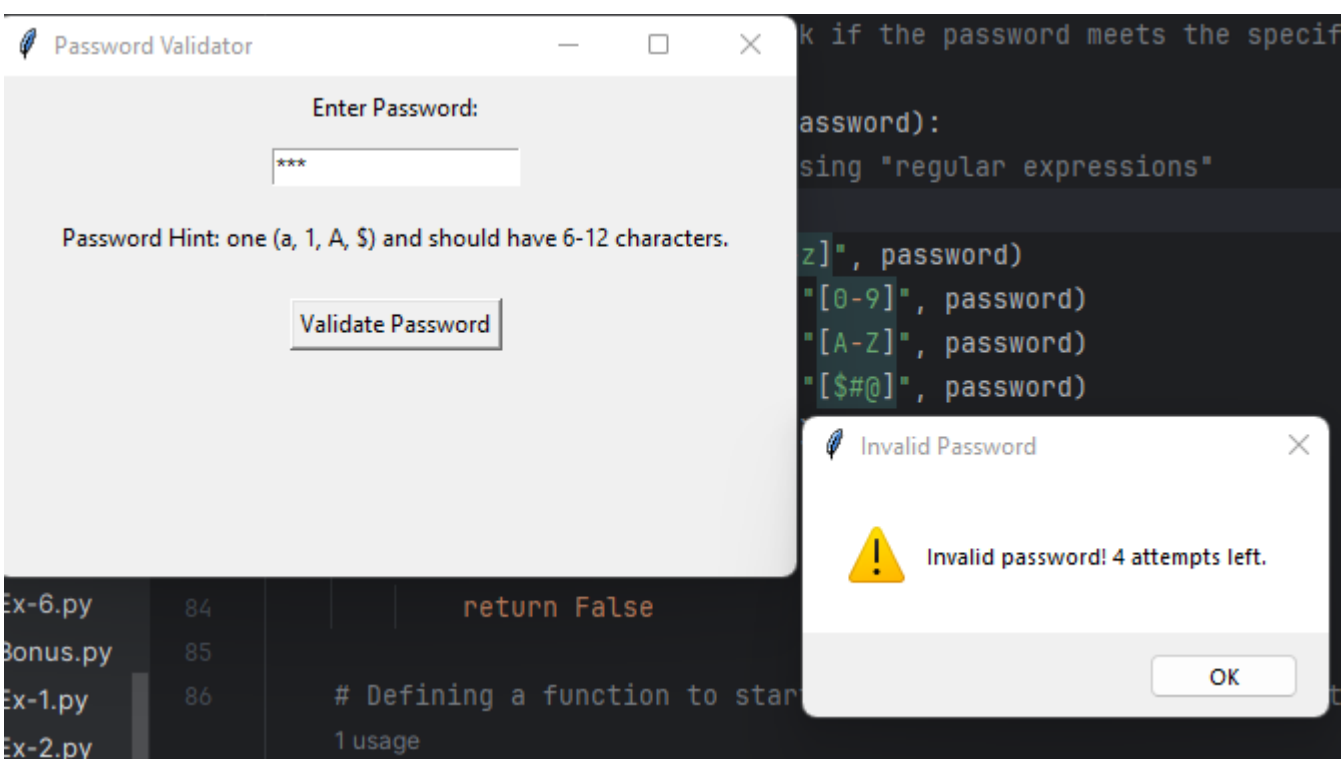

Character Occurrence Counter
C:/Users/work/PycharmProjects/pythonProje
Browse
Enter a character:
uch!
Count Occurrences
Occurrences of 'I love you so much!': 1

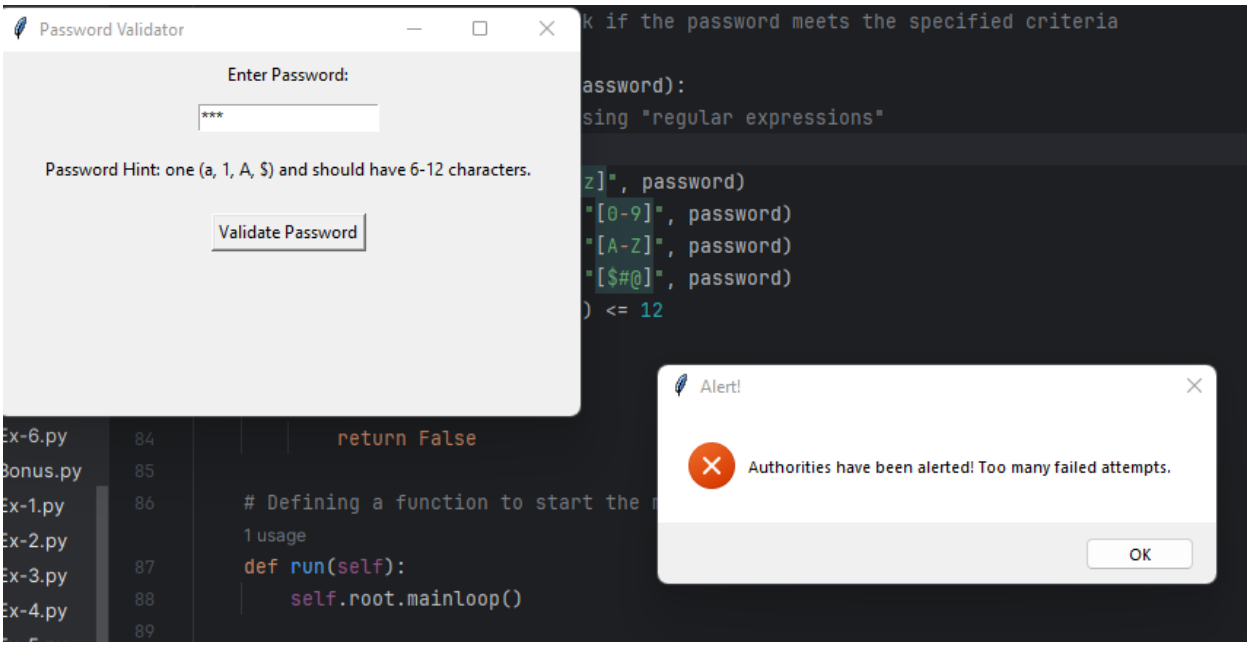
Chap-4 Ex-5

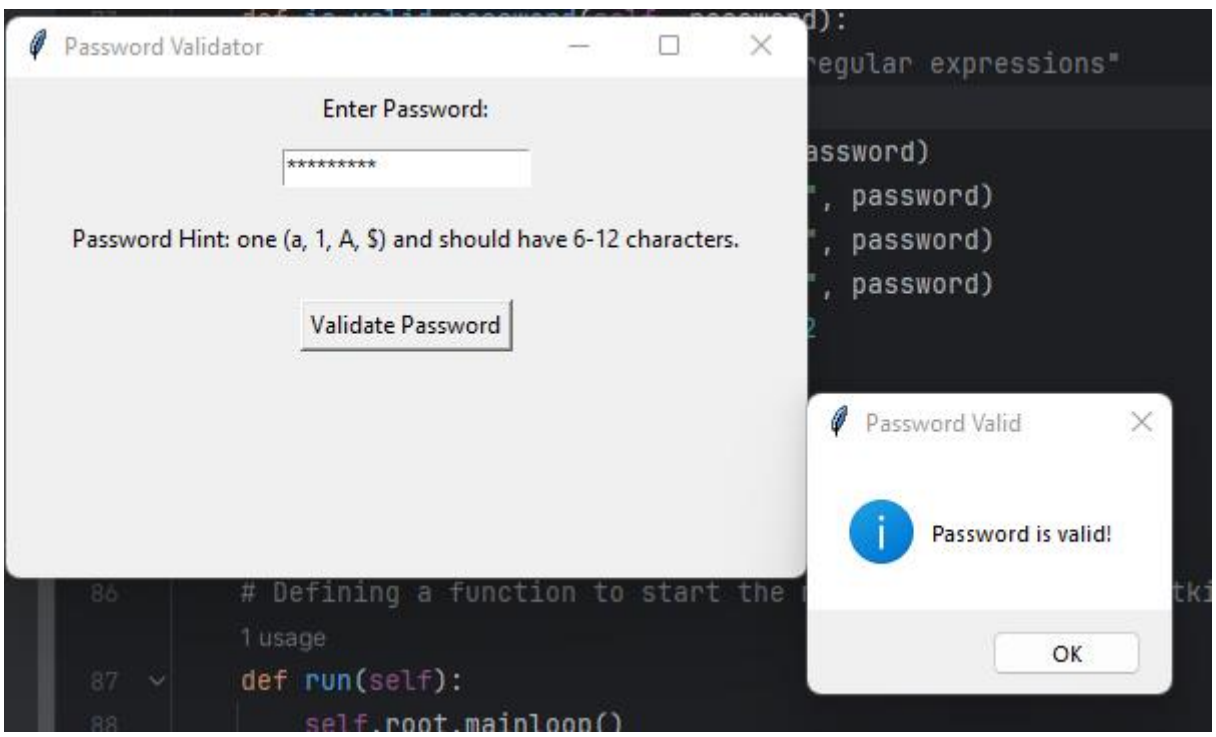
```python
# Chapter 4 Exercise 5: Password Check

# Importing tkinter library and its modules
import ...

# Making a class named 'PasswordValidatorApp'
1 usage
class PasswordValidatorApp:
    # Using Constructor method, and initializing the class instance
    def __init__(self):
        # Initializing the variable 'attempts_left' to 5
        self.attempts_left = 5

        # Creating the main tkinter window
        self.root = tk.Tk()

        # Setting the title of the window to "Password Validator"
        self.root.title("Password Validator")

        # Setting the initial dimensions of the window to 400x250 pixels
        self.root.geometry("400x250")

        # Calling the 'create_widgets' method to create GUI elements
        self.create_widgets()

    # Defining function for creating GUI elements
    1 usage
    def create_widgets(self):
        # Creating a label with the text "Enter Password:" and pack it with some vertical padding
        tk.Label(self.root, text="Enter Password:").pack(pady=5)

        # Creating an entry widget for password input and setting it to show '*' for each character
        self.password_entry.pack(pady=5)

        # Creating a label with a password hint and pack it with some vertical padding
        tk.Label(self.root, text="Password Hint: one (a, 1, A, $) and should have 6-12 characters.").pack(pady=10)

        # Creating a button for validating the password with a command to call 'validate_password'
        validate_button = tk.Button(self.root, text="Validate Password", command=self.validate_password)
        validate_button.pack(pady=10)

    # Defining a function to validate the entered password
    1 usage
    def validate_password(self):
        # Getting the entered password from the entry widget
        password = self.password_entry.get()

        # Checking if the password is valid using the 'is_valid_password' method
        if self.is_valid_password(password):
            # Showing an information message box if the password is valid
            messagebox.showinfo( title: "Password Valid",  message: "Password is valid!")

            # Closing the application window if the password is valid
            self.root.destroy()
        else:
            # Decreasing the 'attempts_left' variable
            self.attempts_left -= 1

            # Checking if there are any attempts left
            if self.attempts_left > 0:
                # Showing a warning message box with the number of attempts left
                messagebox.showwarning( title: "Invalid Password",  message: f"Invalid password! {self.attempts_left} attempts left.")
            else:
                # Showing an error message box and destroy the application window if attempts are exhausted
                messagebox.showerror( title: "Alert!",  message: "Authorities have been alerted! Too many failed attempts.")
```

```
                messagebox.showerror( title: "Alert!",  message: "Authorities have been alerted! Too many failed attempts.")
                self.root.destroy()

    # Defining a function to check if the password meets the specified criteria
    1 usage
    def is_valid_password(self, password):
        # The Password criteria using "regular expressions"
        if (
            re.search( pattern: "[a-z]", password)
            and re.search( pattern: "[0-9]", password)
            and re.search( pattern: "[A-Z]", password)
            and re.search( pattern: "[$#@]", password)
            and 6 <= len(password) <= 12
        ):
            return True
        else:
            return False

    # Defining a function to start the main event loop of the tkinter window
    1 usage
    def run(self):
        self.root.mainloop()

# Instantiating the PasswordValidatorApp class
app = PasswordValidatorApp()

# Running the app
app.run()
```

Output




Password Validator
Enter Password:
***
Password Hint: one (a, 1, A, $) and should have 6-12 characters.
Validate Password
Alert!
Authorities have been alerted! Too many failed attempts.
OK
return False
Defining a function to start the
1 usage
def run(self):
self.root.mainloop()

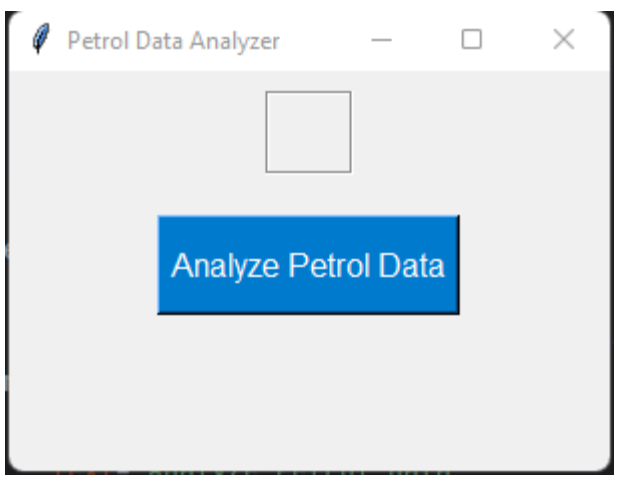

Password Validator
Enter Password:
****
Password Hint: one (a, 1, A, $) and should have 6-12 characters.
Validate Password
Password Valid
Password is valid!
OK
Defining a function to start the
1 usage
def run(self):

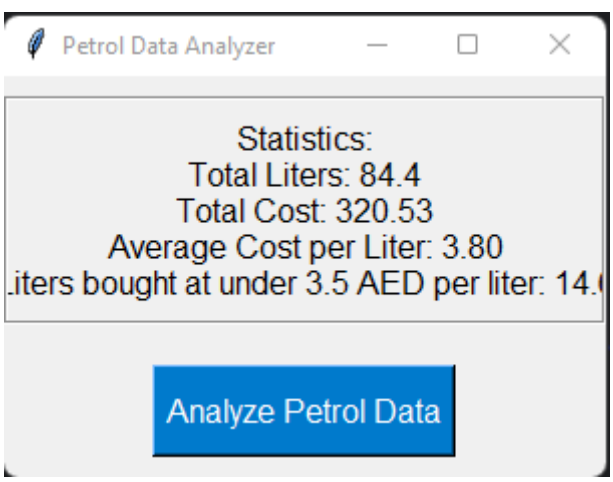
Chap-4 Bonus

```python
# Chapter 4 Bonus Exercise: Petrol Price

# Importing tkinter library
from tkinter import *

# Defining a function to analyze petrol data
1 usage
def analyze_petrol_data():
    try:
        # The File path for petrol data
        petrol_data_file_path = 'petrolPrice.txt'

        # Initializing the variables for calculations
        total_liters = 0
        total_cost = 0
        under_3_5_liters = 0

        # Reading the data from the file and processing each line
        with open(petrol_data_file_path, 'r') as file:
            # Skipping the header line
            next(file)

            for line in file:
                # Splitting the line into columns
                columns = line.strip().split('\t')

                # Extracting liters and cost from the columns
                liters, cost = float(columns[0]), float(columns[1])

                # Adding to totals
                total_liters += liters
                total_cost += cost

                # Checking if the cost per liter is under 3.5 AED
                if liters > 0 and cost / liters < 3.5:
                    under_3_5_liters += liters

        # Calculating average cost per liter
        if total_liters > 0:
            average_cost_per_liter = total_cost / total_liters
        else:
            average_cost_per_liter = 0

        # Displaying statistics
        result_text = f"Statistics:\nTotal Liters: {total_liters}\nTotal Cost: {total_cost}\n"
        result_text += f"Average Cost per Liter: {average_cost_per_liter:.2f}\n"
        result_text += f"Liters bought at under 3.5 AED per liter: {under_3_5_liters}"
        result_label.config(text=result_text)

    except FileNotFoundError:
        result_label.config(text="File not found. Please make sure the file exists.")


# Creating the main window
root = Tk()

# Creating a title for the page
root.title("Petrol Data Analyzer")

# Setting the window size
root.geometry('300x200')

# Using Label for displaying the result with updated design
result_label = Label(
    root,
```

```python
# Using Label for displaying the result with updated design
result_label = Label(
    root,
    text="",
    font=("Arial", 12),
    bg="#f0f0f0",  # Background color
    pady=10,
    padx=20,
    borderwidth=2,
    relief="groove"
)
result_label.pack(pady=10)

# The Button for analyzing the petrol data with a blue color
analyze_button = Button(
    root,
    text="Analyze Petrol Data",
    command=analyze_petrol_data,
    bg="#007acc",  # Blue color
    fg="white",
    font=("Arial", 12),
    pady=10
)
analyze_button.pack(pady=10)

# Running the main event loop
root.mainloop()
```

Output





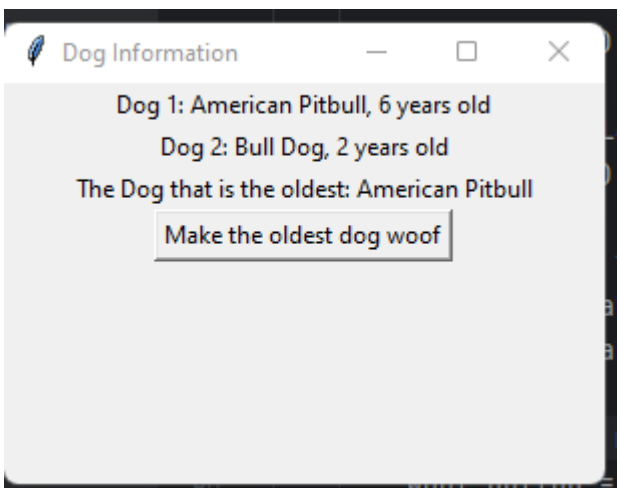
Chapter 5

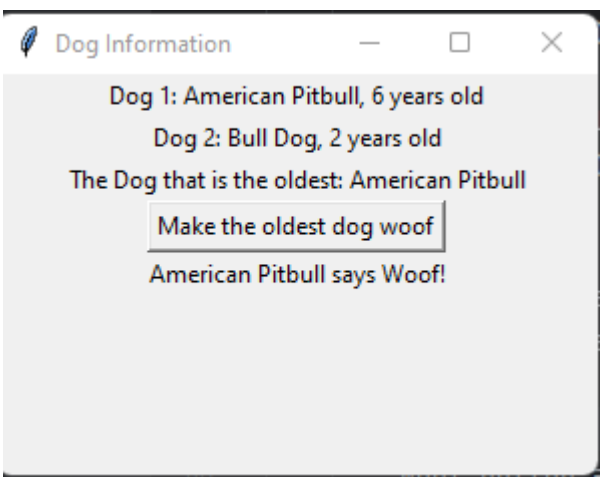
Object Oriented Programming

Chap-5 Ex-1

```python
# Chapter 5 Exercise 1: Woof Woof

# Importing tkinter library
import tkinter as tk

# Making a class called Dog
2 usages
class Dog:
    def __init__(self, name, age):
        # Initializing the Dog object with name and age attributes
        self.name = name
        self.age = age

    1 usage
    def woof(self):
        # Method to make the dog say woof
        return f"{self.name} says Woof!"

# Creating two dog objects
dog1 = Dog( name: "American Pitbull",  age: 6)
dog2 = Dog( name: "Bull Dog",  age: 2)

# Determining the oldest dog
oldest_dog = dog1 if dog1.age > dog2.age else dog2

# Making class for Tkinter GUI
1 usage
class DogGUI(tk.Tk):
    def __init__(self):
        # Initializing the DogGUI window
        super().__init__()
        self.title("Dog Information")
        self.geometry("300x200")
        self.display_dog_info()

    1 usage
    def display_dog_info(self):
        # Displaying the information about each dog and the oldest dog
        label1 = tk.Label(self, text=f"Dog 1: {dog1.name}, {dog1.age} years old")
        label1.pack()

        label2 = tk.Label(self, text=f"Dog 2: {dog2.name}, {dog2.age} years old")
        label2.pack()

        # Displaying the information about the dog that is oldest
        oldest_dog_label = tk.Label(self, text=f"The Dog that is the oldest: {oldest_dog.name}")
        oldest_dog_label.pack()

        # Button for making the oldest dog woof
        woof_button = tk.Button(self, text="Make the oldest dog woof", command=self.woof)
        woof_button.pack()

# Defining function for woof
    1 usage
    def woof(self):
        # Codee to make the oldest dog woof
        result = oldest_dog.woof()
        woof_label = tk.Label(self, text=result)
        woof_label.pack()

# Instantiate the DogGUI class
app = DogGUI()
app.mainloop()
```

Output


Dog Information
Dog 1: American Pitbull, 6 years old
Dog 2: Bull Dog, 2 years old
The Dog that is the oldest: American Pitbull
Make the oldest dog woof

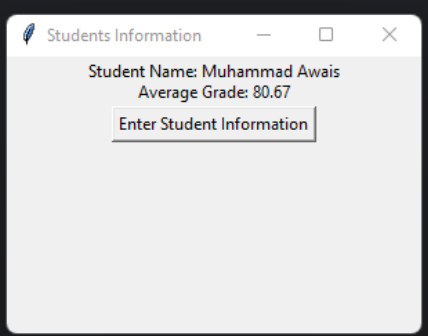

Dog Information
Dog 1: American Pitbull, 6 years old
Dog 2: Bull Dog, 2 years old
The Dog that is the oldest: American Pitbull
Make the oldest dog woof
American Pitbull says Woof!

Chap-5 Ex-2

```python
# Chapter 5 Exercise 2: Student Class

# Importing the tkinter library
import tkinter as tk

# Making a class called students
1 usage
class Students:
    def __init__(self, name, mark1, mark2, mark3):
        # Making a constructor to initialize student attributes
        self.name = name
        self.mark1 = mark1
        self.mark2 = mark2
        self.mark3 = mark3

# Defining function to calculate the grade
    1 usage
    def calc_Grade(self):
        # Codes for calculating the average grade
        return (self.mark1 + self.mark2 + self.mark3) / 3
# Defining function for displaying the information
    def display(self):
        # Codes for displaying student information
        average = self.calc_Grade()
        return f"Student Name: {self.name}\nAverage Grade: {average:.2f}"

# Making a clas for students GUI
1 usage
class StudentsGUI(tk.Tk):
    def __init__(self):
        # Using GUI constructor for styling
        super().__init__()
        self.title("Students Information")
        self.geometry("300x200")
        self.create_student_objects()

# Defining a function for creating student objects
    1 usage
    def create_student_objects(self):
        # Method to create GUI components
        self.label = tk.Label(self, text="The Information about the student will be displayed here.")
        self.label.pack()

       # Using input to store student information
        input_button = tk.Button(self, text="Enter Student Information", command=self.get_student_info)
        input_button.pack()

    1 usage
    def get_student_info(self):
        # Codes to get student information and display it
        try:
            name = input("Enter Student Name: ")
            mark1 = int(input("Enter Mark 1: "))
            mark2 = int(input("Enter Mark 2: "))
            mark3 = int(input("Enter Mark 3: "))

            student = Students(name, mark1, mark2, mark3)

            self.label.config(text=student.display())
        except ValueError:
            print("Invalid input for marks. Please enter numerical values.")

# Running the StudentsGUI class
app = StudentsGUI()
app.mainloop()
```

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-5_Ex-2.py
Enter Student Name: Muhammad Awais
Enter Mark 1: 85
Enter Mark 2: 78
Enter Mark 3: 79
```



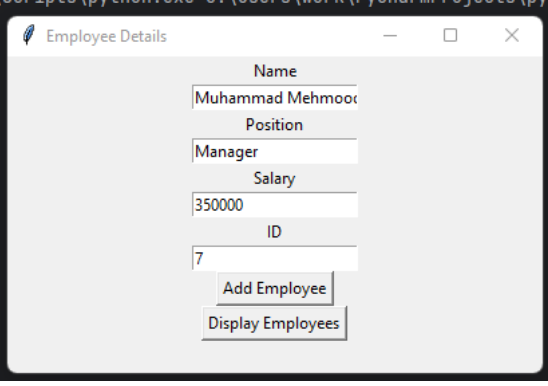
Chap-5 Ex-3

```python
# Chapter 5 Exercise 3: Employee Class

# Importing the tkinter library
import tkinter as tk

# Making a class called employee
1 usage
class Employee:
    def __init__(self):
        # Initializing the attributes for employee
        self.name = ""
        self.position = ""
        self.salary = 0.0
        self.id = ""

# Defining a function for data
    1 usage
    def setData(self, name, position, salary, emp_id):
        # Setting data for the employee
        self.name = name
        self.position = position
        self.salary = salary
        self.id = emp_id

# Defining a function for the the set fata of the employee
    2 usages (1 dynamic)
    def getData(self):
        # Returning formatted employee data as a string
        return f"{self.name}\t{self.position}\t{self.salary}\t{self.id}"

# Making a class for employee GUI and styling it as well
1 usage
class EmployeeGUI(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Employee Details")
        self.geometry("400x300")
        self.employees = []

        self.create_employee_data_fields()

        # Creating a button to add employee
        add_button = tk.Button(self, text="Add Employee", command=self.add_employee)
        add_button.pack()

        # Creating a button to display employees
        display_button = tk.Button(self, text="Display Employees", command=self.display_employees)
        display_button.pack()

# Defining function for employees work filed
    1 usage
    def create_employee_data_fields(self):
        # Creating labels and entry fields for employee data

        # Label and entry for employee name
        self.name_label = tk.Label(self, text="Name")
        self.name_label.pack()
        self.name_entry = tk.Entry(self)
        self.name_entry.pack()

        # Label and entry for employee position
        self.position_label = tk.Label(self, text="Position")
        self.position_label.pack()
        self.position_entry = tk.Entry(self)
        self.position_entry.pack()
```

```
        # Label and entry for employee salary
        self.salary_label = tk.Label(self, text="Salary")
        self.salary_label.pack()
        self.salary_entry = tk.Entry(self)
        self.salary_entry.pack()

        # Label and entry for employee ID
        self.id_label = tk.Label(self, text="ID")
        self.id_label.pack()
        self.id_entry = tk.Entry(self)
        self.id_entry.pack()

# Defining function for adding employee data to the list
    1 usage
    def add_employee(self):
        # Adding employee to the list

        # Checking if the maximum number of employees (5) has been reached
        if len(self.employees) >= 5:
            print("Maximum number of employees reached (5 employees).")
        else:
            # Retrieving employee information from entry fields
            name = self.name_entry.get()
            position = self.position_entry.get()
            salary = float(self.salary_entry.get())
            emp_id = self.id_entry.get()

            # Creating a new Employee instance and set its data
            employee = Employee()
            employee.setData(name, position, salary, emp_id)

            # Adding the employee to the list of employees
            self.employees.append(employee)

            # Printing a message indicating the employee has been added
            print("Employee Added:")
            print(employee.getData())

# Defining a function for displaying employee details
    1 usage
    def display_employees(self):
        # Displaying employee details

        # Checking if there are no employees in the list
        if not self.employees:
            print("No employees to display.")
        else:
            # Printing headers for employee details
            print("Employee Details:")
            print("Name\tPosition\tSalary\tID")

            # Iterating through the list of employees and print their details
            for employee in self.employees:
                print(employee.getData())

# Running the EmployeeGUI class
app = EmployeeGUI()
app.mainloop()
```

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-5_Ex-3.py
Employee Added:
Muhammad Awais  CEO 50000.0 10
Employee Added:
Muhammad Mehmood    Manager 350000.0    7
Employee Details:
Name    Position    Salary  ID
Muhammad Awais  CEO 50000.0 10
Muhammad Mehmood    Manager 350000.0    7
```



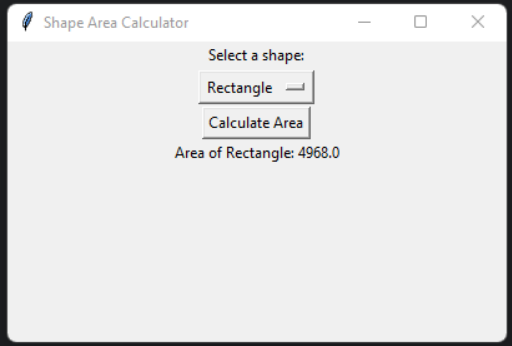
Chap-5 Ex-4

```python
# Chapter 5 Exercise 4: Shapes

# Importing tkinter library
import ...

#  Making a class for shape
3 usages
class Shape:
    # Defining function for storing the sides
    def __init__(self):
        self.sides = []  # Initializing a list to store the sides of the shape

# Defining function for the sides
    def input_sides(self):
        pass  # To implement it by subclasses, collecting the input for the sides

# Defining function for area
    def area(self):
        pass  # To implement it by subclasses, calculating and returning the area

#  Making a class for circle
1 usage
class Circle(Shape):

    # Defining function for the input of the sides
    1 usage
    def input_sides(self):
        radius = float(input("Enter the radius of the circle: "))
        self.sides = [radius]  # Storing the radius in the sides list

    # Defining function for the area
    1 usage
    def area(self):
        return pi * self.sides[0] ** 2  # Calculating and returning the area of the circle

#  Making a class for rectangle
1 usage
class Rectangle(Shape):

    # Defining function for input of the sides of rectangle
    1 usage
    def input_sides(self):
        length = float(input("Enter the length of the rectangle: "))
        width = float(input("Enter the width of the rectangle: "))
        self.sides = [length, width]  # Storing the length and width in the sides list

    # Defining function for input of area of rectangle
    1 usage
    def area(self):
        return self.sides[0] * self.sides[1]  # Calculating and returning the area of the rectangle

#  Making a class for triangle
1 usage
class Triangle(Shape):

    # Defining function for the sides input of triangle
    1 usage
    def input_sides(self):
        base = float(input("Enter the base of the triangle: "))
        height = float(input("Enter the height of the triangle: "))
        self.sides = [base, height]  # Storing the base and height in the sides list

    # Defining function for the area of triangle
    1 usage
    def area(self):
        return 0.5 * self.sides[0] * self.sides[1]  # Calculating and returning the area of the triangle
```

```python
#  Making a class for calculations
1 usage
class ShapeCalculatorGUI(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Shape Area Calculator")
        self.geometry("400x300")
        self.create_widgets()

    # Defining function for the widgets
    1 usage
    def create_widgets(self):
        self.shape_label = tk.Label(self, text="Select a shape:")
        self.shape_label.pack()

        self.shape_var = tk.StringVar()
        self.shape_var.set("Circle")  # Default shape

        # The Option Menu for selecting a shape
        self.shape_menu = tk.OptionMenu(self, self.shape_var, value: "Circle", *values: "Rectangle", "Triangle")
        self.shape_menu.pack()

        self.calculate_button = tk.Button(self, text="Calculate Area", command=self.calculate_area)
        self.calculate_button.pack()

        self.result_label = tk.Label(self, text="")
        self.result_label.pack()

    # Defining function to calculate the area
    1 usage
    def calculate_area(self):
        selected_shape = self.shape_var.get()
        current_shape = None  # Declaring the shape before the if statements

        # Using if statements to determine the type of shape based on user's selection
        if selected_shape == "Circle":
            current_shape = Circle()
        elif selected_shape == "Rectangle":
            current_shape = Rectangle()
        elif selected_shape == "Triangle":
            current_shape = Triangle()
        else:
            return

        # Collecting input for the selected shape
        current_shape.input_sides()

        # Calculating and displaying the area of the selected shape
        area_result = current_shape.area()
        self.result_label.config(text=f"Area of {selected_shape}: {area_result}")


# Running the ShapeCalculatorGUI class
app = ShapeCalculatorGUI()
app.mainloop()
```

Output

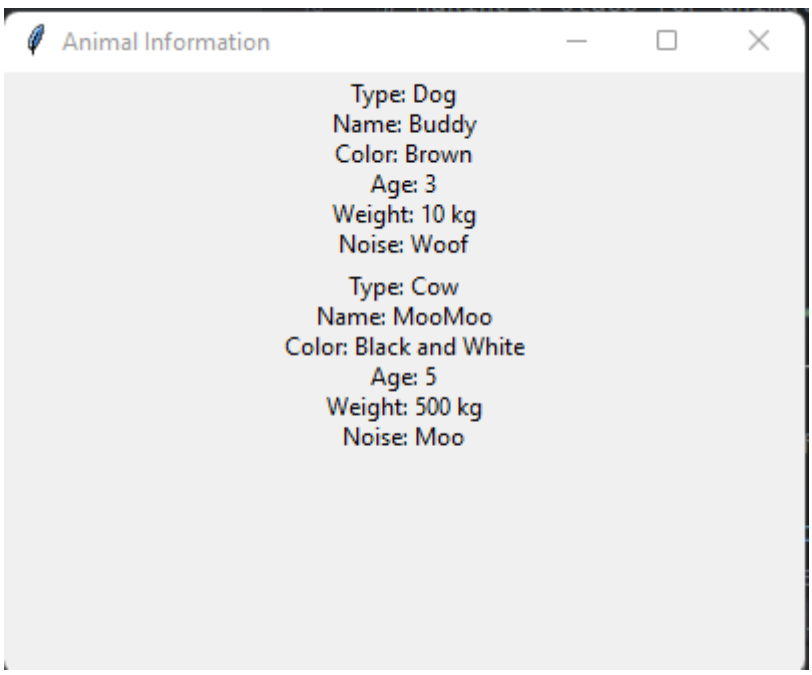
```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-5_Ex-4.py
Enter the length of the rectangle: 414
Enter the width of the rectangle: 12
```



Chap-5 Ex-5

```python
# Chapter 5 Exercise 5: Playing around in class

# Importing the tkinter library
import tkinter as tk

# Making a class called animal
2 usages
class Animal:
    def __init__(self, animal_type, name, color, age, weight, noise):
        # Initializing the Animal object with specified attributes
        self.animal_type = animal_type
        self.name = name
        self.color = color
        self.age = age
        self.weight = weight
        self.noise = noise

# Defining a function for making the animal say hello
    def sayHello(self):
        # Codes to make the animal say hello
        return f"{self.name} says hello!"

    # Defining a function for to make the animal make its noise
    def makeNoise(self):
        # Codes to make the animal produce its characteristic noise
        return f"{self.name} makes the noise: {self.noise}"

    # Defining a function for adding details to the animal
    2 usages
    def animalDetails(self):
        # Codes to get a string representation of all details of the animal
        details = (
            f"Type: {self.animal_type}\n"
            f"Name: {self.name}\n"
            f"Color: {self.color}\n"
            f"Age: {self.age}\n"
            f"Weight: {self.weight} kg\n"
            f"Noise: {self.noise}"
        )
        return details

# Making a class for animal GUI
1 usage
class AnimalGUI(tk.Tk):
    def __init__(self):
        # Initializing the AnimalGUI window and styling it
        super().__init__()
        self.title("Animal Information")
        self.geometry("400x300")
        self.create_animal_objects()

    # Defining a function for animal objects
    1 usage
    def create_animal_objects(self):
        # Creating instances of the Animal class
        dog = Animal( animal_type: "Dog",  name: "Buddy",  color: "Brown",  age: 3,  weight: 10,  noise: "Woof")
        cow = Animal( animal_type: "Cow",  name: "MooMoo",  color: "Black and White",  age: 5,  weight: 500,  noise: "Moo")

        # Creating labels to display information for animal dog
        dog_label = tk.Label(self, text=dog.animalDetails())
        dog_label.pack()
        # Creating labels to display information for animal cow
        cow_label = tk.Label(self, text=cow.animalDetails())
        cow_label.pack()


# Running the AnimalGUI class
app = AnimalGUI()
app.mainloop()
```

Output

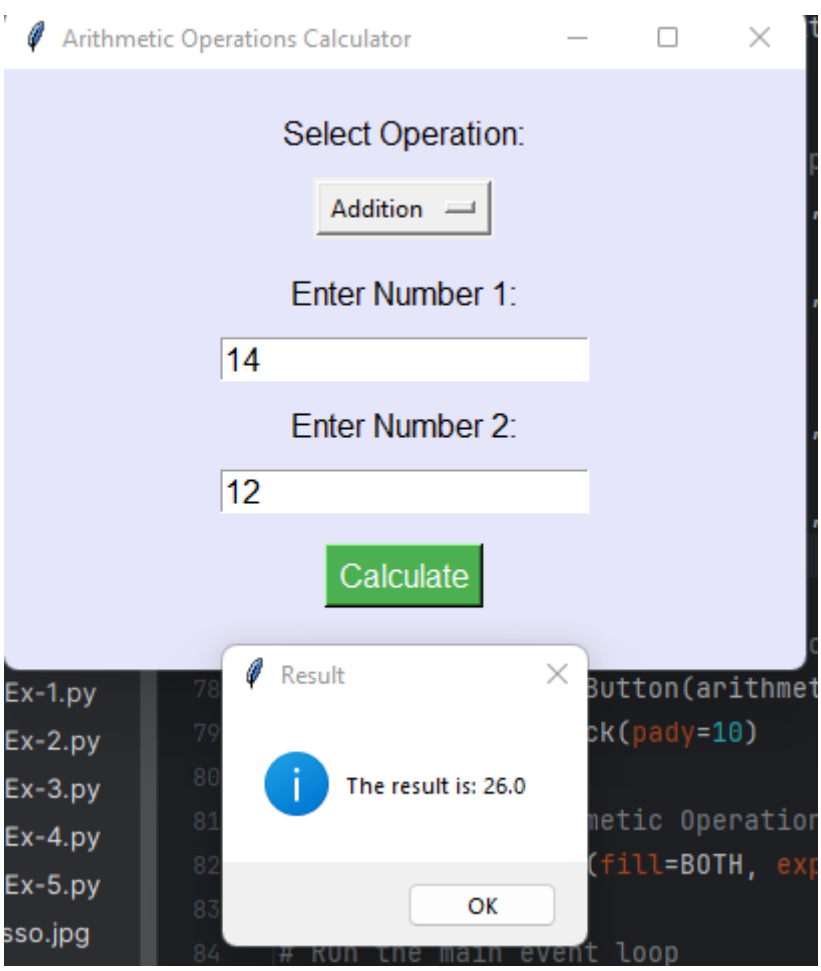

Animal Information
Type: Dog
Name: Buddy
Color: Brown
Age: 3
Weight: 10 kg
Noise: Woof
Type: Cow
Name: MooMoo
Color: Black and White
Age: 5
Weight: 500 kg
Noise: Moo

Chap-5 Ex-6

```python
# Chapter 5 Exercise 6: Arithmetic Operation

# Importing modules from the tkinter library
from tkinter import *
from tkinter import messagebox

# Making a Class for performing arithmetic operations
1 usage
class ArithmeticCalculator:
    1 usage
    def perform_operation(selected_operation, num1, num2):
        try:
            # To Perform the selected arithmetic operation
            if selected_operation == "Addition":
                return num1 + num2
            elif selected_operation == "Subtraction":
                return num1 - num2
            elif selected_operation == "Multiplication":
                return num1 * num2
            elif selected_operation == "Division":
                # Check for division by zero
                return num1 / num2 if num2 != 0 else messagebox.showerror( title: "Error",  message: "Cannot divide by zero.")
            elif selected_operation == "Exponentiation":
                return num1 ** num2
        except ValueError:
            # Handle invalid input values
            messagebox.showinfo( title: "Error",  message: "Please enter valid numeric values.")
            return

# Function to perform calculation when the "Calculate" button is clicked
1 usage
def calculate_result():
    # Getting the selected arithmetic operation
    selected_operation = operation_var.get()

    try:
        # Getting values from entry widgets
        result = ArithmeticCalculator.perform_operation(selected_operation, float(num1_entry.get()), float(num2_entry.get
        if result:
            # Displaying the result in a messagebox pop-up
            messagebox.showinfo( title: "Result",  message: f"The result is: {result}")
    except ValueError:
        # Handle invalid input values
        messagebox.showinfo( title: "Error",  message: "Please enter valid numeric values.")

# Creating the main window
root = Tk()

# Setting the title for the window
root.title("Arithmetic Operations Calculator")

# Setting the window size
root.geometry("400x300")

# Widgets for Arithmetic Operations tab
arithmetic_tab = Frame(root, bg="#E6E6FA", pady=20)

# Label for selecting the arithmetic operation
operation_label = Label(arithmetic_tab, text='Select Operation:', font=("Helvetica", 12), bg="#E6E6FA")
operation_label.pack()

# Options for the arithmetic operations presented in a drop-down menu
operations = ["Addition", "Subtraction", "Multiplication", "Division", "Exponentiation"]
operation_var = StringVar()
operation_var.set(operations[0])
operation_option = OptionMenu(arithmetic_tab, operation_var,  *values: *operations)
```

```
operation_option.pack(pady=10)

# Labels and Entry widgets for inputting numeric values
num1_label = Label(arithmetic_tab, text="Enter Number 1:", font=("Helvetica", 12), bg="#E6E6FA")
num1_label.pack(pady=5)
num1_entry = Entry(arithmetic_tab, font=("Helvetica", 12))
num1_entry.pack(pady=5)

num2_label = Label(arithmetic_tab, text="Enter Number 2:", font=("Helvetica", 12), bg="#E6E6FA")
num2_label.pack(pady=5)
num2_entry = Entry(arithmetic_tab, font=("Helvetica", 12))
num2_entry.pack(pady=5)

# Button to trigger the calculation with specified text, command, font, and color
calculate_button = Button(arithmetic_tab, text='Calculate', command=calculate_result, font=("Helvetica", 12), bg="#4CAF50", fg="white")
calculate_button.pack(pady=10)

# Packing the Arithmetic Operations tab
arithmetic_tab.pack(fill=BOTH, expand=YES)

# Run the main event loop
root.mainloop()

# End Marker
```

Output



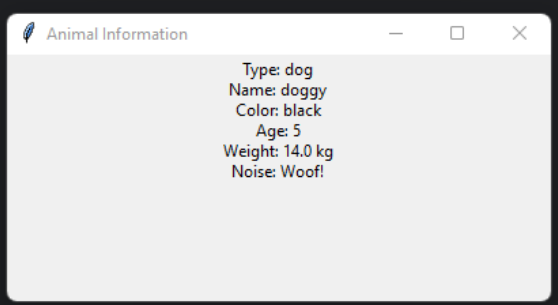
Chap-5 Bonus

```python
# Chapter 5 Bonus Exercise: Playing around in class

# Importing the tkinter library
import tkinter as tk

# Using class as animal
1 usage
class Animal:
    def __init__(self, animal_type, name, color, age, weight, noise):
        # Initializing the Animal object with specified attributes
        self.animal_type = animal_type
        self.name = name
        self.color = color
        self.age = age
        self.weight = weight
        self.noise = noise

    def sayHello(self):
        # Method to make the animal say hello
        return f"{self.name} says hello!"

    def makeNoise(self):
        # Method to make the animal produce its characteristic noise
        return f"{self.name} makes the noise: {self.noise}"

    1 usage
    def animalDetails(self):
        # Method to get a string representation of all details of the animal
        details = (
            f"Type: {self.animal_type}\n"
            f"Name: {self.name}\n"
            f"Color: {self.color}\n"
            f"Age: {self.age}\n"
            f"Weight: {self.weight} kg\n"
            f"Noise: {self.noise}"
        )
        return details # Returning details

# Using class as animalGUI
1 usage
class AnimalGUI(tk.Tk):
    def __init__(self):
        # Initializing the AnimalGUI window
        super().__init__()
        self.title("Animal Information")
        self.geometry("400x300")
        self.creating_animal_objects()

# Defining function to create animal objects
    1 usage
    def creating_animal_objects(self):
        # Getting user input for creating instances of the Animal class
        animal_type = input("Enter animal type: ")
        name = input("Enter name: ")
        color = input("Enter color: ")
        age = int(input("Enter age: "))
        weight = float(input("Enter weight in kg: "))
        noise = input("Enter characteristic noise: ")

        # Creating an instance of the Animal class with user input
        user_animal = Animal(animal_type, name, color, age, weight, noise)

        # Creating a label to display information for the user's animal
        user_animal_label = tk.Label(self, text=user_animal.animalDetails())
        user_animal_label.pack()
```

```
        # Creating a label to display information for the user's animal
        user_animal_label = tk.Label(self, text=user_animal.animalDetails())
        user_animal_label.pack()

# Instantiate the AnimalGUI class
app = AnimalGUI()
app.mainloop()
```

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-5_Bonus.py
Enter animal type: dog
Enter name: doggy
Enter color: black
Enter age: 5
Enter weight in kg: 14
Enter characteristic noise: Woof!
```



Chapter 6
Python Standard Library

Chap-6 Ex-1

Code and Output

```python
# Chapter 6 Exercise 1: Basic Math

# Importing the math library
import math

# Finding the ceil of a
ceil_a = math.ceil(2.3)
print(f"Ceil of 2.3: {ceil_a}")

# Finding the floor of a
floor_a = math.floor(2.3)
print(f"Floor of 2.3: {floor_a}")

# Finding the factorial of a
factorial_a = math.factorial(5)
print(f"Factorial of 5: {factorial_a}")

# Finding the value of 2^3
power_a = math.pow( _x: 2,  _y: 3)
print(f"Value of 2^3: {power_a}")

# Finding the square root of a
sq_rt_a = math.sqrt(16)
print(f"Square root of 16: {sq_rt_a}")
```

```
un  Chap-6_Ex-1 ×
⋮
Floor of 2.3: 2
Factorial of 5: 120
Value of 2^3: 8.0
Square root of 16: 4.0

Process finished with exit code 0
```

Chap-6 Ex-2

```python
# Chapter 6 Exercise 2: Numpy Array

# Importing the numpy library
import numpy as np

# The given array
a = np.array([20, 23, 82, 40, 32, 15, 67, 52])

# Finding the indices of even numbers
indices_even = np.where(a % 2 == 0)
print(f"Indices of even numbers: {indices_even}")

# Sorting the array
array_sorted = np.sort(a)
print(f"Sorted array: {array_sorted}")

# Slicing the elements beginning from index 3 to the end of the array
slice_01 = a[3:]
print(f"Slice from index 3 to the end: {slice_01}")

# Slicing the elements beginning from index 0 to index 4
slice_02 = a[:5]
print(f"Slice from index 0 to index 4: {slice_02}")

# Printing the following [32 15 67] using negative slicing
slice_negative = a[-5:-2]
print(f"Negative slicing to get [32 15 67]: {slice_negative}")
```

Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-6_Ex-2.py
Indices of even numbers: (array([0, 2, 3, 4, 7], dtype=int64),)
Sorted array: [15 20 23 32 40 52 67 82]
Slice from index 3 to the end: [40 32 15 67 52]
Slice from index 0 to index 4: [20 23 82 40 32]
Negative slicing to get [32 15 67]: [40 32 15]

Process finished with exit code 0
```

Chap-6 Ex-3

```python
# Chapter 6 Exercise 3: Calculator

# Importing operator library
import operator

# Defining function to perform addition
1 usage
def add(x, y):
    return operator.add( *args: x, y)

# Defining function to perform subtraction
1 usage
def subtract(x, y):
    return operator.sub( *args: x, y)

# Defining function to perform multiplication
1 usage
def multiply(x, y):
    return operator.mul( *args: x, y)

# Defining function to perform division
1 usage
def divide(x, y):
    if y != 0:
        return operator.truediv( *args: x, y)
    else:
        return "Unable to divide by zero"

# Defining function to calculate modulus
1 usage
def modulus(x, y):
    return operator.mod( *args: x, y)
```

```python
# Defining function to check and return the greater number
1 usage
def checking_greater(x, y):
    return f"{max(x, y)} is greater"

# Defining function to get user input for the calculator
1 usage
def getting_input():
    try:
        # Getting user input for the operation choice
        choose = input("Choose operation (1-6) or Q to quit: ").upper()
        if choose == 'Q': # Using if statement for choice selection
            return 7, None, None  # Returning a different choice value for quitting
        else:
            choose = int(choose)
            # Checking if the choice is within the valid range or not
            if choose < 1 or choose > 6: # Using if statement for choice condition
                raise ValueError("Invalid choice") # If not show invalid
            x = float(input("Enter first number: "))
            y = float(input("Enter second number: "))
            return choose, x, y
    except ValueError:
        print("Invalid input. Please enter a valid choice and numerical values.")
        return None, None, None

# Defining Main function to run the calculator
def main():
    while True:
        print("\nCalculation Menu: Select your choice")
        print("1. Add\n2. Subtract\n3. Multiply\n4. Divide\n5. Modulus\n6. Check greater number")

        # Getting user input for the operation choice and numbers
        choose, x, y = getting_input()
```

```
        # Using if statement for choice selection
        if choose is None:
            continue

        if choose == 6:
            final = checking_greater(x, y)
            print(f"Result: {final}")
        elif choose == 7:
            print("Exiting calculator. Goodbye!")
            break
        elif 1 <= choose <= 5:
            operations = [add, subtract, multiply, divide, modulus]
            # Performing the selected operation and print the result
            final = operations[choose - 1](x, y)
            print(f"Result: {final}")

# Running the main function
if __name__ == "__main__":
    main()
```

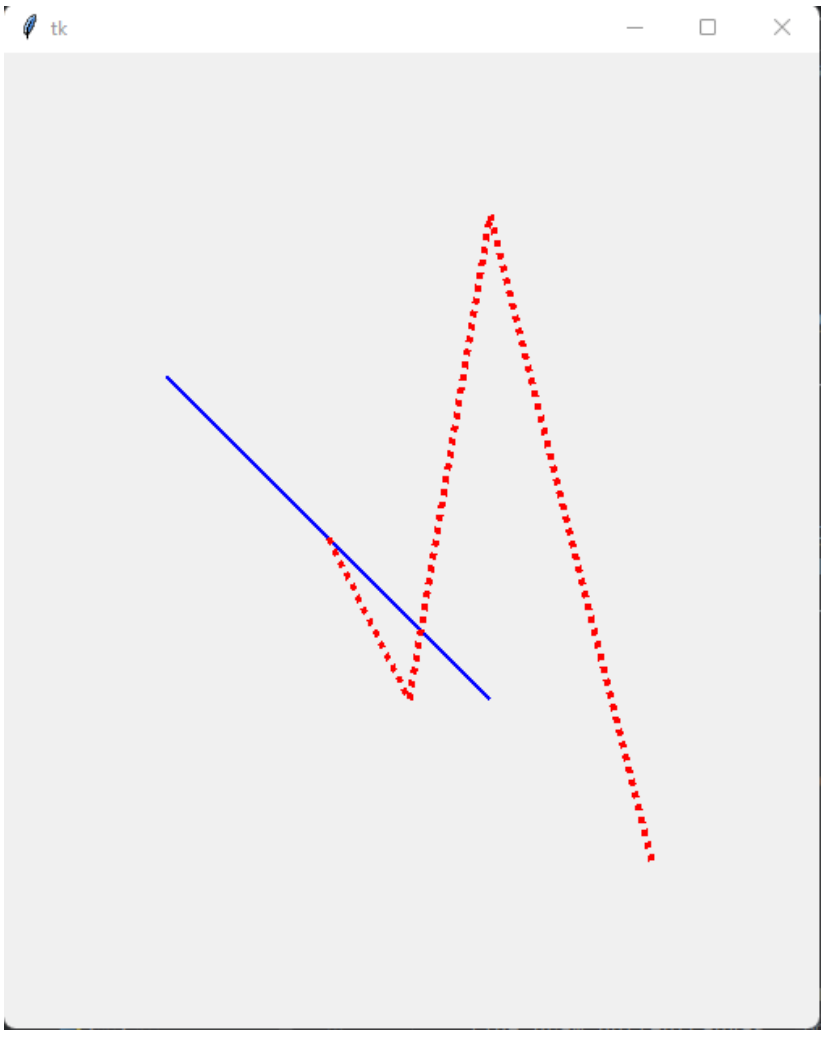
Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-6_Ex-3.py

Calculation Menu: Select your choice
1. Add
2. Subtract
3. Multiply
4. Divide
5. Modulus
6. Check greater number
Choose operation (1-6) or Q to quit: 2
Enter first number: 78
Enter second number: -65
Result: 143.0

Calculation Menu: Select your choice
1. Add
2. Subtract
3. Multiply
4. Divide
5. Modulus
6. Check greater number
Choose operation (1-6) or Q to quit: q
Exiting calculator. Goodbye!

Process finished with exit code 0
```

Chap-6 Ex-4

```python
# Chapter 6 Exercise 4: Line graph

# Importing tkinter library
import tkinter as tk

# Defining function for drawing the line using canvas
1 usage
def line_draw(canvas, start, end, color, width):
    # Drawing a line from start to end
    line = canvas.create_line(start[0], start[1], end[0], end[1], fill=color, width=width)

# Defining function for drawing a dotted line using canvas
1 usage
def line_draw_dotted(canvas, points, color, width, dash):
    # Draw a dotted line through the specified points
    dotted_line = canvas.create_line(*points, fill=color, width=width, dash=dash)

# Defining the main function
def main():
    root = tk.Tk()
    canvas = tk.Canvas(root, width=500, height=600) # Setting the width and the height
    canvas.pack()

    # Drawing a solid line from (1, 2) to (6, 8)
    line_draw(canvas, start: (100, 200), end: (300, 400), color: "blue", width: 2)

    # Drawing a dotted line from (1, 3) to (2, 8) then to (6, 1) and finally to (8, 10)
    line_draw_dotted(canvas, points: [200, 300, 250, 400, 300, 100, 400, 500], color: "red", width: 4, dash: (5, 5))

    # Running the main function
    root.mainloop()

if __name__ == "__main__":
```

Output



Chap-6 Ex-5

```python
# Chapter 6 Exercise 5: Working with JSON File

# Importing json library
import json

# Defining function to write into json file
1 usage
def writing_into_json(filename, data):
    # Opening the JSON file in write mode and use json.dump to write data to the file
    with open(filename, 'w') as file:
        json.dump(data, file)

# Defining function to read from the json file
1 usage
def reading_from_json(filename):
    # Opening the JSON file in read mode and use json.load to read data from the file
    with open(filename, 'r') as file:
        data = json.load(file)
        return data

def main():
    # Creating Task 1 which was to Write to JSON file
    student_data = {}
    # Get input from the user for student name, ID, and course
    student_data['Name'] = input("Enter the student name: ")
    student_data['ID'] = int(input("Enter the student ID: "))
    student_data['Course'] = input("Enter the student course: ")

    # Appending CourseDetails dictionary to student_data
    student_data['CourseDetails'] = {
        'Group': input("Enter the student group: "),
        'Year': int(input("Enter the student year: "))
    }

    # Calling the write_into_json function to write data to the JSON file
    writing_into_json( filename: 'StudentJson.json', student_data)
    print("Details of the student are written to StudentJson.json")

    # Creating Task 2 which is to Read from JSON file
    # Call the reading_from_json function to read data from the JSON file
    retrieved_data = reading_from_json('StudentJson.json')

    print("\nDetails of the Student are")
    print(f"  Name: {retrieved_data['Name']}")
    print(f"  ID: {retrieved_data['ID']}")
    print(f"  Course: {retrieved_data['Course']}")

    # Displaying the CourseDetails
    print("  CourseDetails:")
    print(f"    Group: {retrieved_data['CourseDetails']['Group']}")
    print(f"    Year: {retrieved_data['CourseDetails']['Year']}")

# Ending the code
if __name__ == "__main__":
    main()
```

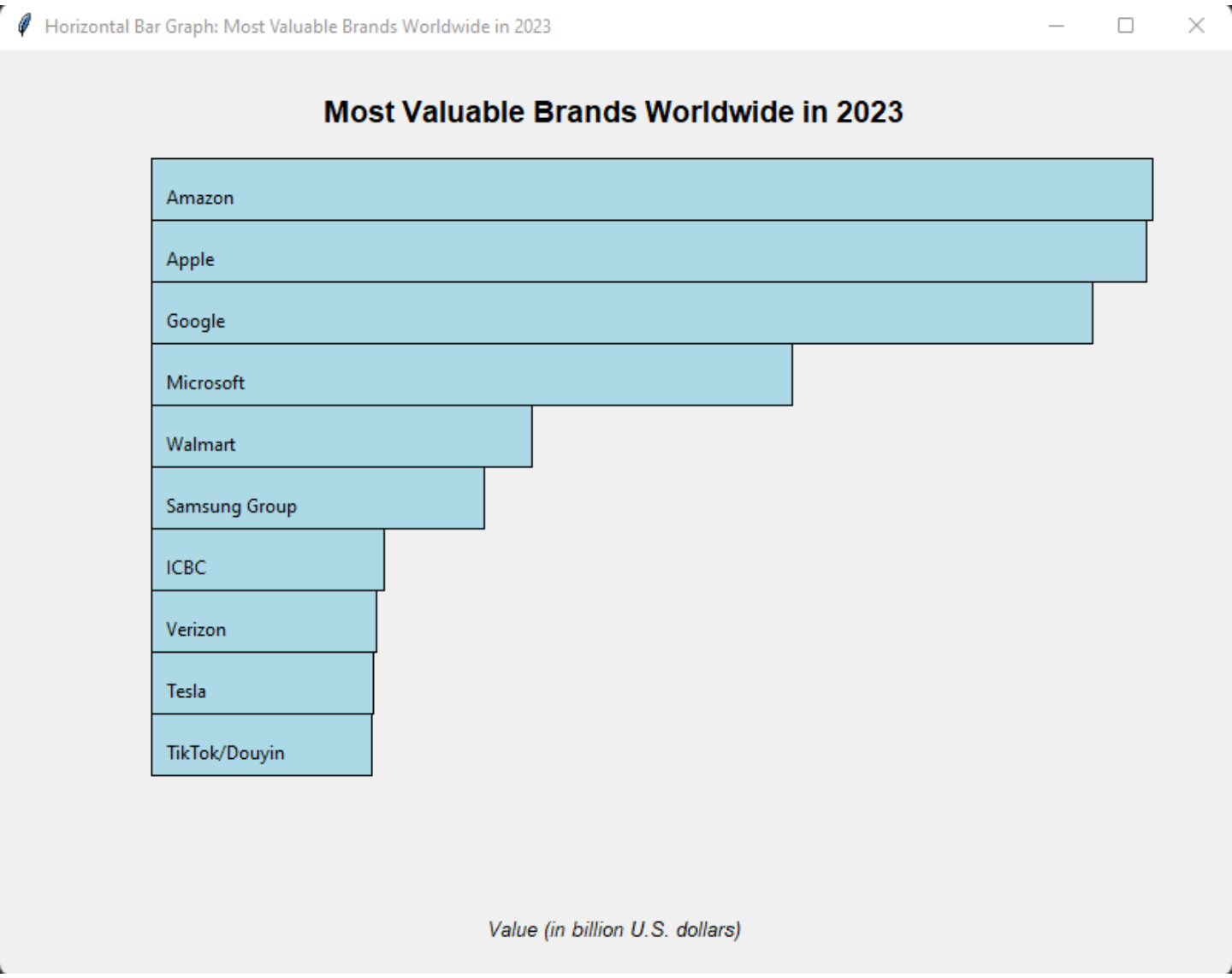
Output

```
C:\Users\work\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\work\PycharmProjects\pythonProject1\Chap-6_Ex-5.py
Enter the student name: Muhammad Awais
Enter the student ID: 14
Enter the student course: CC
Enter the student group: 4
Enter the student year: 2
Details of the student are written to StudentJson.json

Details of the Student are
  Name: Muhammad Awais
  ID: 14
  Course: CC
  CourseDetails:
    Group: 4
    Year: 2

Process finished with exit code 0
```

Chap-6 Bonus

```python
# Chapter 6 Exercise: Bar graph

# Importing tkinter library
import tkinter as tk

# Using class function for bar graph and setting the required parameters
1 usage
class Horizontal_BarGraph(tk.Tk):
    def __init__(self, brands, values):
        super().__init__()
        self.title("Horizontal Bar Graph: Most Valuable Brands Worldwide in 2023")
        self.geometry("800x600")
        self.brands = brands
        self.values = values
        self.drawing_horizontal_bar_graph()

# Defining function for making the graph horizontal
    1 usage
    def drawing_horizontal_bar_graph(self):
        canvas = tk.Canvas(self, width=700, height=600)
        canvas.pack(pady=20)

        # Calculating the maximum value to scale the bar widths
        max_value = max(self.values)
        scale_factor = 650 / max_value

        # Drawing bars for each brand using for loop
        for a in range(len(self.brands)):
            bar_width = self.values[a] * scale_factor
            canvas.create_rectangle(
                50, 50 + a * 40,
                50 + bar_width, 90 + a * 40,
                fill="lightblue",
                outline="black"
            )
            # Displaying brand names in such a way that they are on the left of the bars
            canvas.create_text(
                60, 75 + a * 40,
                text=self.brands[a],
                anchor=tk.W  # Anchor to the west (left) for better visibility
            )

        # Displaying the title
        canvas.create_text(
            350, 20,
            text="Most Valuable Brands Worldwide in 2023",
            font=("Helvetica", 14, "bold")
        )

        # Displaying the horizontal label
        canvas.create_text(
            350, 550,
            text="Value (in billion U.S. dollars)",
            font=("Helvetica", 10, "italic")
        )

# Given Data of brands and values
brands = ["Amazon", "Apple", "Google", "Microsoft", "Walmart", "Samsung Group", "ICBC", "Verizon", "Tesla", "TikTok/Douyin"]
values = [299.28, 297.51, 281.38, 191.57, 113.78, 99.66, 69.55, 67.44, 66.21, 65.67]

# Instantiate the HorizontalBarGraph class with the provided data
app = Horizontal_BarGraph(brands, values)
app.mainloop()
```

Output



Regards: Muhammad Awais