Advance Programming

MUHAMMAD RAHIL SAEED

Chapter-1

CHAPTER 1-PYTHON REFRESHER

Chapter 1 Ex-1

Code

```python
# Greeting User
print("Hello,user!")
# Asking for Input
Name = input("Enter your Name: ")
Age = int(input("Enter your Age:"))
# the length of your name
length = len(Name)
# Age after 1 year
Next_age = Age+1
# printing the statements
print("Name:"+Name)
print("Age:"+str(Age))
print("The Length of your Name:"+str(length))
print("Your age next year:"+str(Next_age))
```

OUTPUT

```
Hello,user!
Enter your Name: Rahil
Enter your Age: 19
Name:Rahil
Age:19
The Length of your Name:5
Your age next year:20
```

Chapter 1 Ex-2

```python
# Asking input from user
num1 = int(input("Enter the first integer: "))
num2 = int(input("Enter the Second integer: "))
sum_result = num1+num2
sub_result = num1-num2
mul_result = num1 * num2
quotient_result = num1 / num2
remainder_result = num1 % num2
# Printing all the results
print(f"Sum: {sum_result}")
print(f"Subtract: {sub_result}")
print(f"multiple: {mul_result}")
print(f"Quotient: {quotient_result}")
print(f"Remainder: {remainder_result}")
```

OUTPUT

```
Enter the first integer: 5
Enter the Second integer: 5
Sum: 10
Subtract: 0
multiple: 25
Quotient: 1.0
Remainder: 0
```

Chapter 1 Ex-3

```python
# taking input for the 3 side of triangle
a = float(input("Enter the length of the first side of triangle:"))
b = float(input("Enter the length of the second side of triangle:"))
c = float(input("Enter the length of the third side of triangle:"))

# nested if statement to check the type of triangle
if a + b > c and a + c > b and b + c > a:
    if a == b == c:
        print("Equilateral Triangle")
    elif a == b or a == c or b == c:
        print("Isosceles Triangle")
    else:
        print("Scalene Triangle")
else:
    print("Not a valid triangle")
```

OUTPUT

```
Enter the length of the first side of triangle:5
Enter the length of the second side of triangle:4
Enter the length of the third side of triangle:5
Isosceles Triangle
```

Chapter 1 Ex-4

```python
# Asking user to input 3 numbers
num1 = int(input("Enter your first number:"))
num2 = int(input("Enter your Second number:"))
num3 = int(input("Enter your third number:"))
# Multiple if else statements
if num1>= num2 and num1>=num3:
    largest = num1
elif num2>= num1 and num2>=num3:
    largest = num2
else:
    largest = num3
# Printing the output
print("The largest number is:",largest)
```

OUTPUT

```
Enter your first number: 4
Enter your Second number: 6
Enter your third number: 7
The largest number is: 7
```

Chapter 1 Ex-5

```python
count = 0
# while statement
while True:
    user_input = input("Do you want to continue? (Y/N):") # user input
    if user_input.upper() == 'Y':
        count += 1
    else:
        break

# printing the statement
    print(f"The loop was executed {count} times.")
```

```
Do you want to continue? (Y/N):Y
The loop was executed 1 times.
Do you want to continue? (Y/N):Y
The loop was executed 2 times.
Do you want to continue? (Y/N):N
```

Chapter 1 Ex-6

```python
# Using for loop
for i in range(1,100):
    # if, elif and else statements
    if i % 3==0 and i % 5==0:
        print("FizzBuzz")
    elif i % 3 ==0:
        print("Fizz")
    elif i % 5==0:
        print("Buzz")
    else:
        print(i)
```

```
1
2
Fizz
4
Buzz
Fizz
7
8
```

```
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
```

Chapter 1 Ex-7

```python
for a in range(1,101):
    if a % 2 !=0:
        continue
    print(a)
```

```
2
4
6
8
10
12
14
16
```

Chapter 1 Ex-8

```python
# using for loop
for i in range(1,6):
    for j in range(1, i+1): # adding to i
        print(j, end=" ")
    print()
```

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

Chapter 1 Ex-9

```python
# creating an integer list with 10 values
list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print("List using a for loop:")
for number in list:
    print(number,end="")
# printing the max and min values
print("\nHighest value:",max(list))
print("Lowest value:",min(list))

# Sorting the values
list.sort()
print("Sorted in ascending order:", list)

# descending order
list.sort(reverse=True)
print("Sorted in descending order:", list)
 # Appending the list
 list.append(11)
 list.append(12)

 print("List after appending:", list)
```

```
List using a for loop:
12345678910
Highest value: 10
Lowest value: 1
Sorted in ascending order: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sorted in 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, 12]
```

Chapter 1 Ex-10

```python
# Creating a dictionary of film details
films=[{
    "Title":"Inception",
    "Director":"Christopher Nolan",
    "Release year": 2010
},
 {  "Title":"The Dark Knight",
    "Director":"Christopher Nolan",
    "Release year": 2008
},
 ]
# using for loop as required
for film in films:
    print("Title:", film["Title"])
    print("Director:", film["Director"])
    print("Release year:", film["Release year"])
```

```
Title: Inception
Director: Christopher Nolan
Release year: 2010
Title: The Dark Knight
Director: Christopher Nolan
Release year: 2008
```

Chapter 1 Ex-11

```python
# Creating tuple with values
Year= (2017,2003,2011,2005,1987,2009,2020,2018,2009)

# Access value at index-3
value_index= Year[-3]
print("Value at index-3:",value_index)

# Reversing the tuple
reversed_year = tuple(reversed(Year))
print("Original tuple:", Year)
print("Reversed tuple:", reversed_year)

# counting number of times 2009 in the tuple
count= Year.count(2009)
print("Number of times 2009 is in the tuple:", count)

# Getting the index value
index= Year.index(2018)
print("Index of 2018:", index)
# length of tuple
length= len(Year)
print("Length of the tuple:", length)
```

```
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
```

Chapter 1 Ex-12

```
#Area and circumfrence of Circle
choice = int(input("Which shape do you want to calculate: \n 1.Circle \n 2.rectangle \n 3.triangle \n"))
if choice == 1:
  choice2 = int(input("what do you want to calculate: \n1.Area \n2.Circumference \n"))
  if choice2 == 1:
    r = float(input("Enter the radius"))
    Area = 3.14*r**2
    print("The area of Circle is:",Area)
  elif choice2 == 2:
    r = float(input("Enter the radius"))

    Circumference=2*3.14*r
    print("The circumference of circle  is:",Circumference)
  else:
    print("Invalid")

#Area and Circumference of rectangle
if choice == 2:
  choice2 = int(input("what do you want to calculate: \n1.Area \n2.Circumference \n"))
```

Chapter 1 Ex-12

```
if choice2 == 1:
  r = float(input("Enter the radius"))
  Area = 4*3.14*r**2
  print("The area of rectangle is:",Area)
elif choice2 == 2:

  L = float(input("Enter the value of L \n"))
  w = float(input("Enter the value of w \n"))
  Circumference = 2*L+2*w
  print("The circumference of rectangle  is:",Circumference)

else:
  print("Invalid")
```

Chapter 1 Ex-12

```
#Area and Circumference of triangle
if choice == 3:
  choice2 = int(input("what do you want to calculate: \n1.Area \n2.Circumference \n"))
  if choice2 == 1:

    b = float(input("Enter the value of b \n"))
    h = float(input("Enter the value of h \n"))
    Area = 1/2*b*h
    print("The area of triangle is:", Area)
  elif choice2 == 2:

    #Asking user to input
    A = float(input("Enter the value of A \n"))
    B = float(input("Enter the value of B \n"))
    C = float(input("Enter the value of C \n"))
    Circumference = A+B+C
    print("The circumference of triangle  is:", Circumference)
  else:
    print("Invalid")
```

Chapter 1 Ex-12 - Output

```
Which shape do you want to calculate:
 1.Circle
 2.rectangle
 3.triangle
1
```

```
what do you want to calculate:
1.Area
2.Circumference
1
Enter the radius5
The area of Circle is: 78.5
```

Chapter 1 Ex-13

```python
# first defining the function
1 usage
def calculate_number(list):
    product = 1
    for value in list:
        return  product

list=[2,3,4,5]

# calculate the product using function
result= calculate_number(list)

# printing the output
print("The product of the list values is:", result)
```

```
The product of the list values is: 1
```

Chapter 1 Bonus Ex-A

```python
# using for loop for the range
for i in range(1,11):
    print(f"Multiplication Table for {i}:")
    for j in range (1,11):
        print(f"{i}x{j}={i*j}")
        print()
```

Chapter 1 Bonus Ex-A Output

```
Multiplication Table for 1:
1x1=1

1x2=2

1x3=3

1x4=4

1x5=5

1x6=6

1x7=7

1x8=8

1x9=9

1x10=10
```

```
Multiplication Table for 2:
2x1=2

2x2=4

2x3=6

2x4=8

2x5=10

2x6=12

2x7=14

2x8=16

2x9=18

2x10=20
```

```
Multiplication Table for 10:
10x1=10

10x2=20

10x3=30

10x4=40

10x5=50

10x6=60

10x7=70

10x8=80

10x9=90

10x10=100
```

Chapter 1 Bonus Ex-B

```python
# list for locations
list =['dubai','paris', 'switzerland', 'London', 'amsterdam', 'New York']
# Printing the locations and it' length
print("list:", list)
print("Length of list:",len(list))

# Use sorted() to print the list in alphabetical order without modifying the actual list
print("Sorted list (alphabetical order):", sorted(list))


print("Original list:", list)

# Use sorted() to print the list in reverse alphabetical order without modifying the actual list
print("Sorted list (reverse alphabetical order):", sorted(list, reverse=True))

print("Original list:", list)

# Use reverse() to change the order of the list
list.reverse()
print("Reversed list:", list)

list.sort()
print("Sorted list (alphabetical order):", list)

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

Chapter 1 Bonus Ex-B Output

```
list: ['dubai', 'paris', 'switzerland', 'London', 'amsterdam', 'New York']
Length of list: 6
Sorted list (alphabetical order): ['London', 'New York', 'amsterdam', 'dubai', 'paris', 'switzerland']
Original list: ['dubai', 'paris', 'switzerland', 'London', 'amsterdam', 'New York']
Sorted list (reverse alphabetical order): ['switzerland', 'paris', 'dubai', 'amsterdam', 'New York', 'London']
Original list: ['dubai', 'paris', 'switzerland', 'London', 'amsterdam', 'New York']
Reversed list: ['New York', 'amsterdam', 'London', 'switzerland', 'paris', 'dubai']
Sorted list (alphabetical order): ['London', 'New York', 'amsterdam', 'dubai', 'paris', 'switzerland']
Sorted list (reverse alphabetical order): ['switzerland', 'paris', 'dubai', 'amsterdam', 'New York', 'London']
```

Chapter 1 Bonus Ex-C

```
1 usage
def calculator_menu():
    while True:
        print("Calculator Menu:")
        print("1. Add")
        print("2. Subtract")
        print("3. Multiply")
        print("4. Divide")
        print("5. Modulus")

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

        num1 = float(input("Enter first number: "))
        num2 = float(input("Enter second number: "))

        result = 0

        def add(x, y):
            return x + y

        def subtract(x, y):
            return x - y

        def multiply(x, y):
            return x * y

        def divide(x, y):
            if y == 0:
                return "Cannot divide by zero!"
            else:
                return x / y
```

```
        def modulus(x, y):
            if y == 0:
                return "Cannot find modulus by zero!"
            else:
                return x % y

        if choice == 1:
            result = add(num1, num2)
        elif choice == 2:
            result = subtract(num1, num2)
        elif choice == 3:
            result = multiply(num1, num2)
        elif choice == 4:
            result = divide(num1, num2)
        elif choice == 5:
            result = modulus(num1, num2)
        else:
            print("Invalid choice!")

        print("Result:", result)

        repeat = input("Do you want to perform another calculation? (yes/no): ")
        if repeat.lower() != "yes":
            break

calculator_menu()
```

Chapter 1 Bonus Ex-C Output

```
Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Modulus
Enter operation choice: 1
Enter first number: 7
Enter second number: 8
Result: 15.0
Do you want to perform another calculation? (yes/no): no
```

Chapter 1 Further Ex-1

```python
# Asking user to enter the number of days
days = int(input("Enter the number of days:"))
# converting days into hours and hours into minutes and seconds
seconds = days * 24 * 60 * 60

print(f"There are {seconds} seconds in {days} days")
```

```
Enter the number of days:1
There are 86400 seconds in 1 days
```

Chapter 1 Further Ex-2

```python
# asking for input from user
num=input("Enter a number: ")

sum = 0

for digit in num:
    sum += int(digit)

# printing the statement
print(f"The sum of digits in the number {num} is: {sum}")
```

```
Enter a number: 1234
The sum of digits in the number 1234 is: 10
```

Chapter 1 Further Ex-3

```python
# 5 rows
rows = 5
for i in range(1, rows + 1):
    # printing the spaces
    for j in range(rows-i):
        print(" ", end="")
        # printing the asterisks
    for k in range(2 * i-1):
        print("*", end="")
    print()
```

```
    *
   ***
  *****
 *******
*********

Process finished with exit code 0
```

Chapter 1 Further Ex-4

```python
# Assigning the variable to the names
names = ["Arshiya", "Usman",
         "Iftikhar", "Usman", "Rafia",
         "Mary", "Anmol", "Zainab", "Iftikhar",
         "Arshiya", "Rafia", "Jake"
        ]

names_counts = {}

# Count occurrences of each item in the list
for item in names:
    if item in names_counts:
        names_counts[item] += 1
    else:
        names_counts[item] = 1

# Display the number of times each item appears
for key, value in names_counts.items():
    print(f"{key}: {value} times")
```

```
Arshiya: 2 times
Usman: 2 times
Iftikhar: 2 times
Rafia: 2 times
Mary: 1 times
Anmol: 1 times
Zainab: 1 times
Jake: 1 times

Process finished with exit code 0
```

Chapter 1 Further Ex-5

```python
marks = [("CodeLab I", 67), ("web Development", 75),
         ("CodeLabII", 74), ("Smartphone Apps", 68),
         ("Games Development", 70), ("Responsive web", 65)]

#  (low to high)
sorted_high = sorted(marks, key=lambda x: x[1])
print("Sorted by marks (low to high):", sorted_high)

# (high to low)
sorted_low = sorted(marks, key=lambda x: x[1], reverse=True)
print("Sorted by marks (high to low):", sorted_low)
```

Chapter 1 Further Ex-5 Output

```
Sorted by marks (low to high): [('Responsive web', 65), ('CodeLab I', 67), ('Smartphone Apps', 68), ('Games Development', 70), ('CodeLabII', 74), ('web Development', 75)]
Sorted by marks (high to low): [('web Development', 75), ('CodeLabII', 74), ('Games Development', 70), ('Smartphone Apps', 68), ('CodeLab I', 67), ('Responsive web', 65)]
```

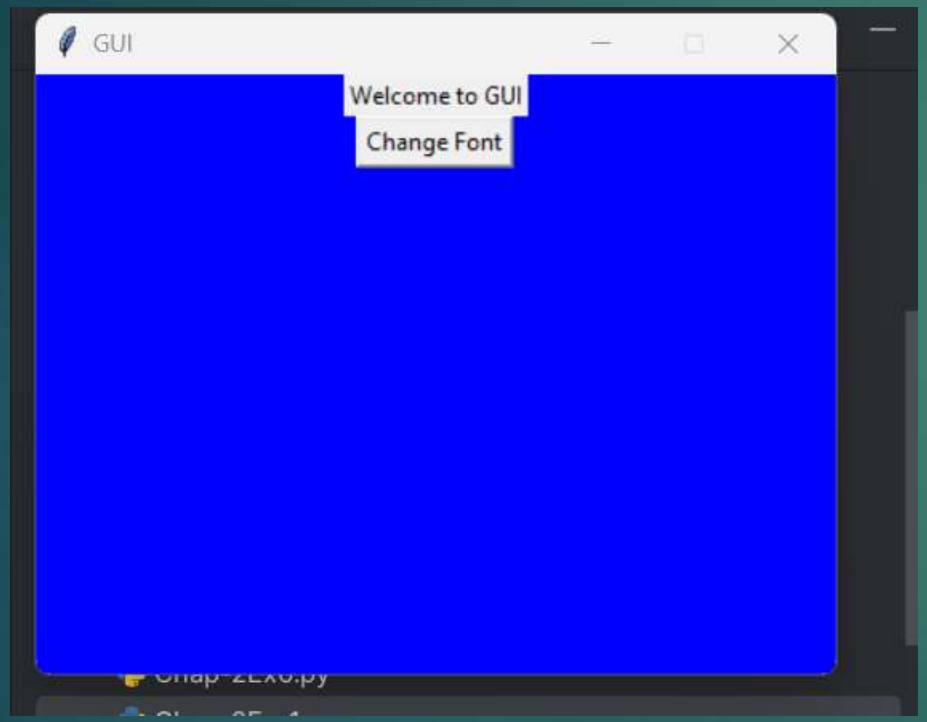
Chapter-2

CHAPTER 2 -GRPAHICAL USER INTERFACE

Chapter 2 Ex-1

```python
# importing tkinkter
import tkinter as tk

1 usage
def font():
    label.config(font='Arial') # changing font

root=tk.Tk()
root.title("GUI")

# creating a label with welcome message
label=tk.Label(root, text="Welcome to GUI")
label.pack()

# button
button=tk.Button(root, text="Change Font", command=font)
button.pack()

root.geometry("400x300")
# Disable the resizeable option
root.resizable( width: False,  height: False)

# bg color
root.config(background='Blue')

root.mainloop()
```

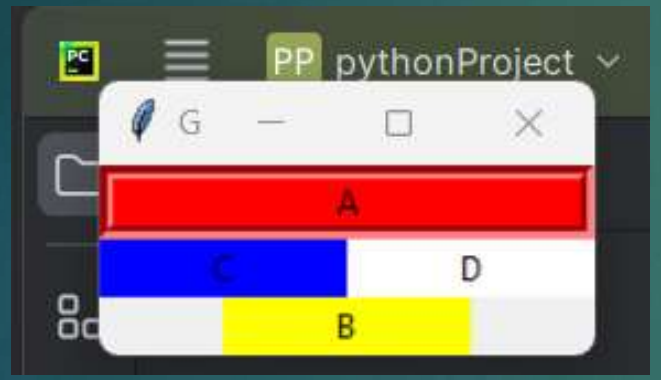
Chapter 2 Ex-1 Output

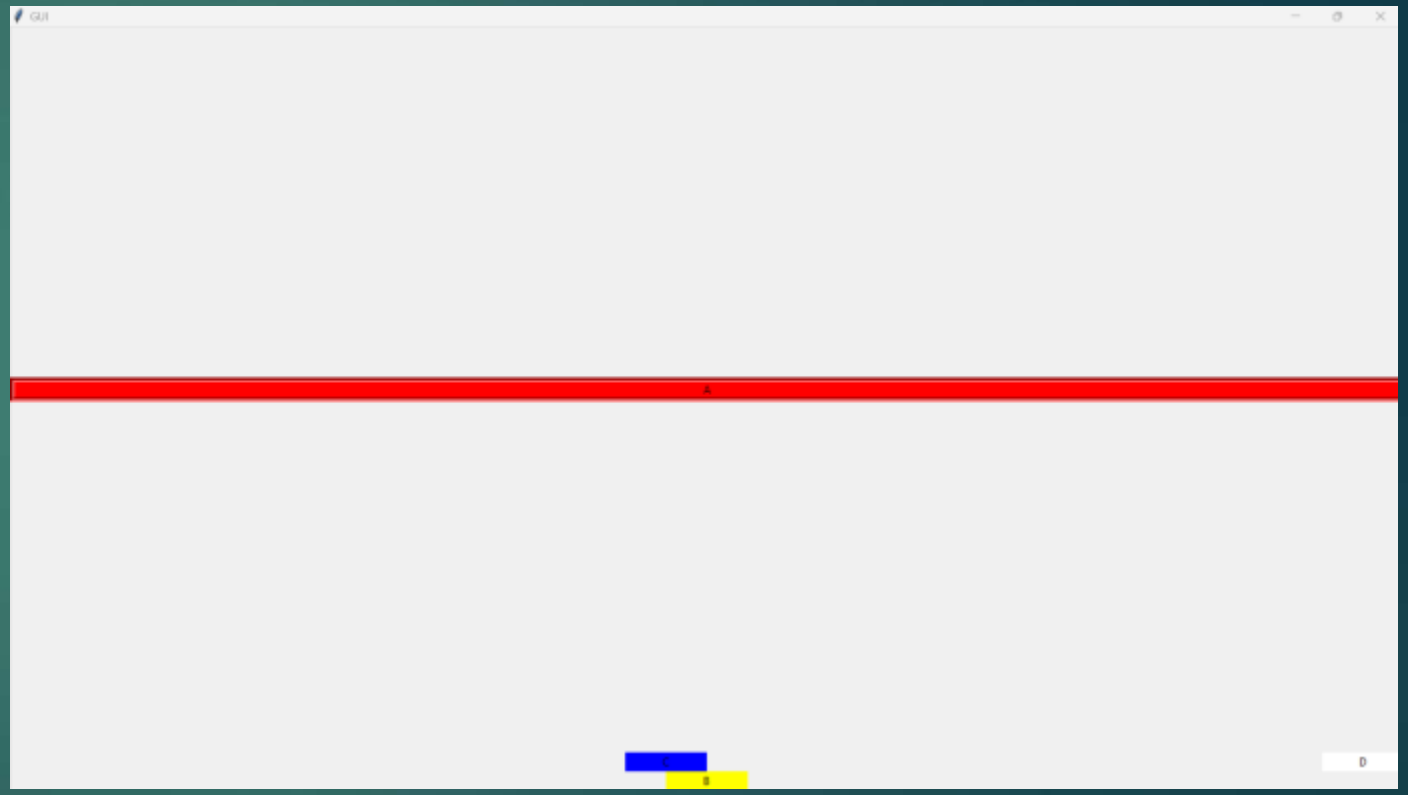


Chapter 2 Ex-2

```
from tkinter import *
import random
app=Tk()
# App title
app.title("GUI")
# labels for the buttons
bA = Label(app,text="A",width=12,bg='red',relief=GROOVE,bd=5)
bB = Label(app,text="B",width=12,bg='yellow')
bC = Label(app,text="C",width=12,bg='blue')
bD = Label(app,text="D",width=12,bg='white')
bA.pack(side='top',fill=X,expand=1)
bB.pack(side='bottom')
bC.pack(side='left',fill=Y,expand=1)
bD.pack(side='right')
app.mainloop()
```

Chapter 2 Ex-2 Output

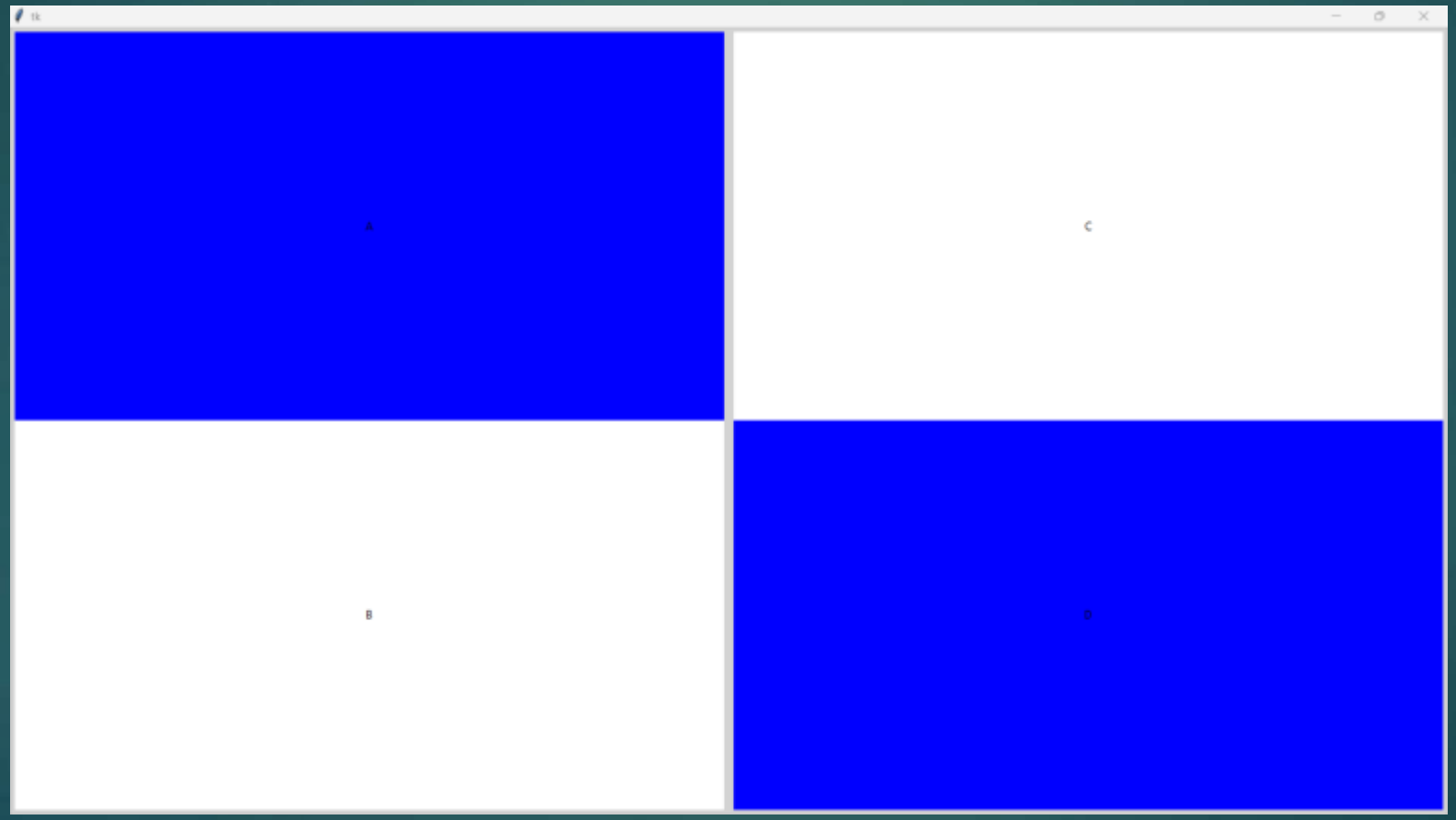




Chapter 2 Ex-2 Part B

```python
# import tkinter as tk.
import tkinter as tk
# The root
root = tk.Tk()


left_frame = tk.Frame(root, borderwidth=5, background='Light grey')
left_frame.pack(side=tk.LEFT, expand=True, fill=tk.BOTH)

# This label will have a text of A inside and background as blue.
label_a = tk.Label(left_frame, text='A', background='blue')
label_a.pack(side=tk.TOP, expand=True, fill=tk.BOTH)

# This label will have a text of B inside and background as white.
label_b = tk.Label(left_frame, text='B', background='White')
label_b.pack(side=tk.BOTTOM, expand=True, fill=tk.BOTH)

# C and B.
MyRight_frame = tk.Frame(root, borderwidth=5, background='Light Grey')
MyRight_frame.pack(side=tk.RIGHT, expand=True, fill=tk.BOTH)

#background as white.
label_c = tk.Label(MyRight_frame, text='C', background='White')
label_c.pack(side=tk.TOP, expand=True, fill=tk.BOTH)

#This label will have a text of D inside and background as blue.
label_d = tk.Label(MyRight_frame, text='D', background='blue')
label_d.pack(side=tk.BOTTOM, expand=True, fill=tk.BOTH)

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

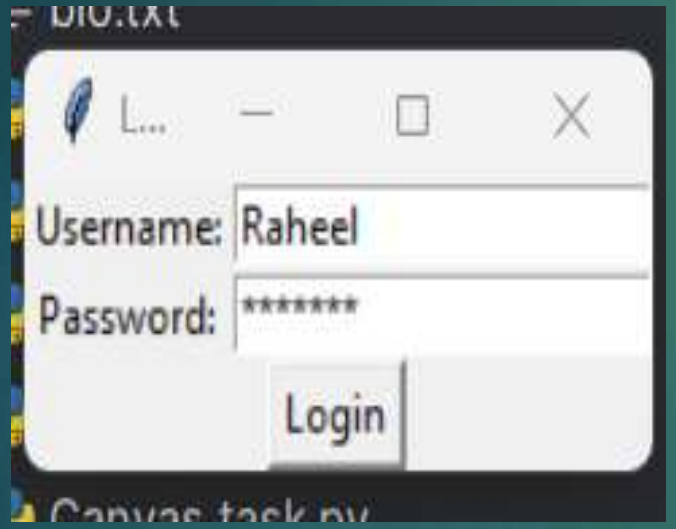
Chapter 2 Ex-2 Part B Output

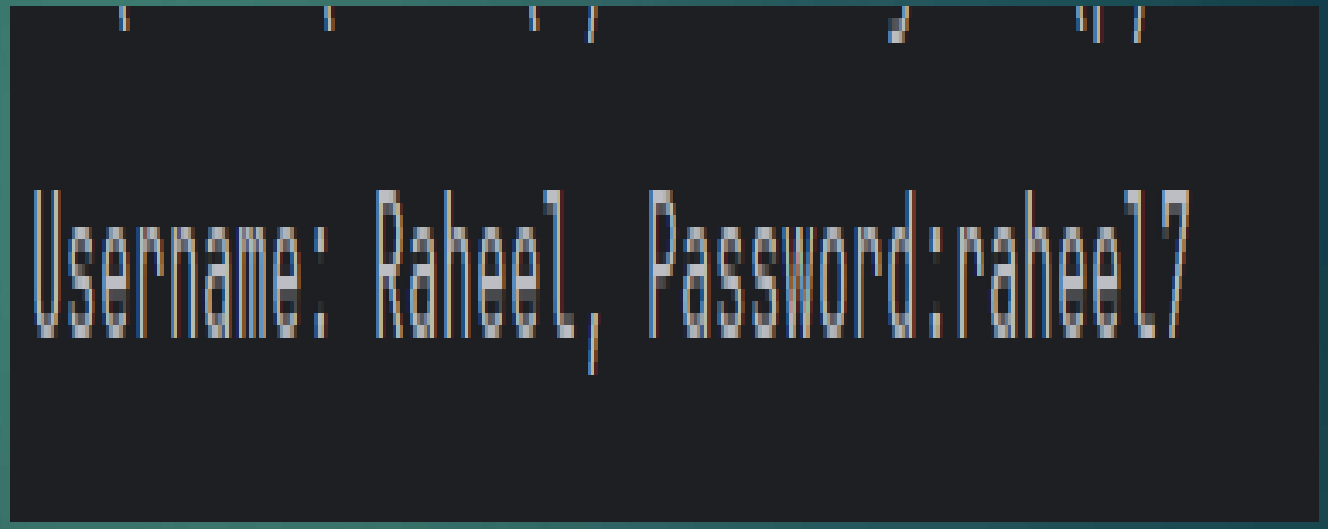


Chapter 2 Ex-3

```python
import tkinter as tk


1 usage
def login():
    username = username_entry.get()
    password = password_entry.get()
    print(f"Username: {username}, Password:{password}")

root=tk.Tk()
root.title("Login Page")
# Creating labels for the username and password
username_label = tk.Label(root, text="Username:")
password_label = tk.Label(root, text="Password:")
username_entry = tk.Entry(root)
password_entry = tk.Entry(root, show="*")

username_label.grid(row=0, column=0)
username_entry.grid(row=0, column=1)
password_label.grid(row=1, column=0)
password_entry.grid(row=1, column=1)

button = tk.Button(root, text="Login", command=login)
button.grid(row=2, columnspan=2)
root.mainloop()
```

Chapter 2 Ex-3 Output





Chapter 2 Ex-4

```python
import tkinter as tk
from tkinter import ttk

1usage
def form():
    # Retrieve and print 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())

1usage
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 choice
    var_course.set("")  # Clear buttons
    var_languages.set("")  # Clear the boxes
    scale_communication.set(0)  # Reset the scale
```

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

# University LOGO Picture
img = tk.PhotoImage(file='uni3.png')
# Adjust width and height
img = img.subsample( x: 1,  y: 1)
label_picture = tk.Label(root, image=img)
label_picture.grid(row=0, column=0, columnspan=2)

# 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)

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

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

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

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

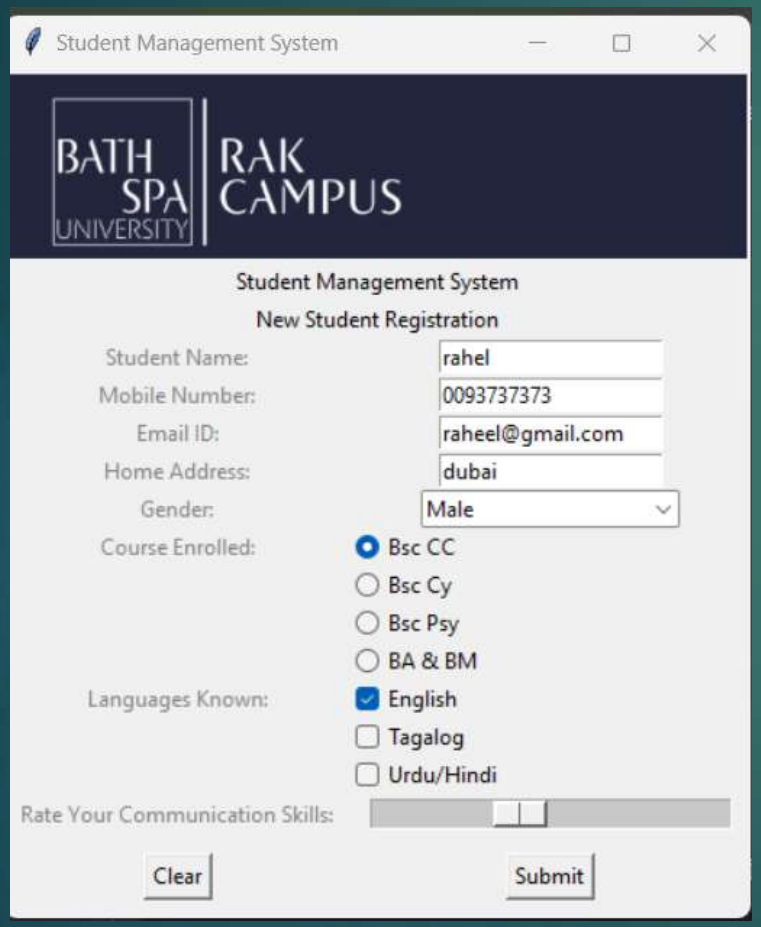
Chapter 2 Ex-4

```
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", "Other"])
gender_choices.grid(row=7, column=1)

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")

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")

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)
```

Chapter 2 Ex-4

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

# Clear and Submit 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=form)
submit_button.grid(row=17, column=1, pady=10)

# Run the GUI
root.mainloop()
```

Chapter 2 Ex-4 Output




```
Student Name: rahel
Mobile Number: 0093737373
Email ID: raheel@gmail.com
Home Address: dubai
Gender: Male
Course Enrolled: Bsc CC
Languages Known: English
Communication Skills: 4
```

Chapter 2 Ex-5

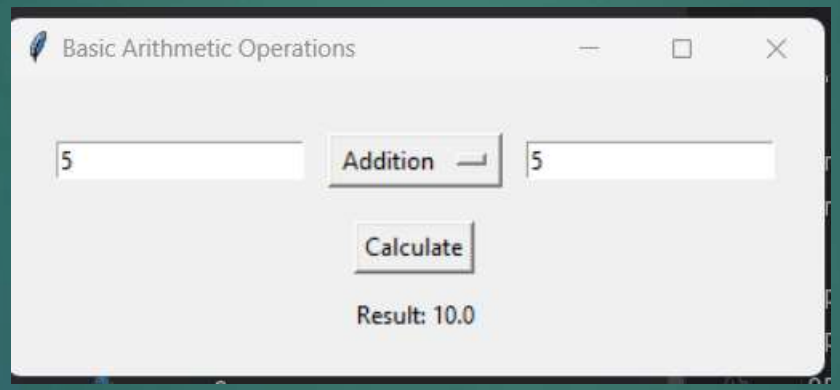
```python
# Importing tkinter
import tkinter as tk

1usage
def operations():
    try:
        num1 = float(entry1.get())
        num2 = float(entry2.get())
        operation = operation_var.get()
# nested statements
        if operation == "Addition":
            result = num1 + num2
        elif operation == "Subtraction":
            result = num1 - num2
        elif operation == "Multiplication":
            result = num1 * num2
        elif operation == "Division":
            if num2 == 0:
                result = "Cannot divide by zero"
            else:
                result = num1 / num2
        elif operation == "Modulo Division":
            if num2 == 0:
                result = "Cannot modulo divide by zero"
            else:
                result = num1 % num2
        else:
            result = "Select an operation"
```

```python
        label.config(text=f"Result: {result}")
    except ValueError:
        label.config(text="Please enter valid numbers")

# main window
root = tk.Tk()
root.title("Basic Arithmetic Operations")

frame = tk.Frame(root)
frame.pack(padx=20, pady=20)
# frame for entry 1
entry1 = tk.Entry(frame)
entry1.grid(row=0, column=0, padx=5, pady=5)

operation_var = tk.StringVar(root)
operation = ["Addition", "Subtraction", "Multiplication", "Division", "Modulo Division"]
operation_menu = tk.OptionMenu(frame, operation_var, *values: *operation)
operation_menu.grid(row=0, column=1, padx=5, pady=5)
operation_var.set("Addition")

# frame for entry 2
entry2 = tk.Entry(frame)
entry2.grid(row=0, column=2, padx=5, pady=5)

# button
calculate_button = tk.Button(frame, text="Calculate", command=operations)
calculate_button.grid(row=1, columnspan=3, padx=5, pady=10)
```

Chapter 2 Ex-5

```
# button
calculate_button = tk.Button(frame, text="Calculate", command= operations)
calculate_button.grid(row=1, columnspan=3, padx=5, pady=10)

# adding label
label = tk.Label(frame, text="Result: ")
label.grid(row=2, columnspan=3)

# returning to the main loop
root.mainloop()
```

Chapter 2 Ex-5 Output

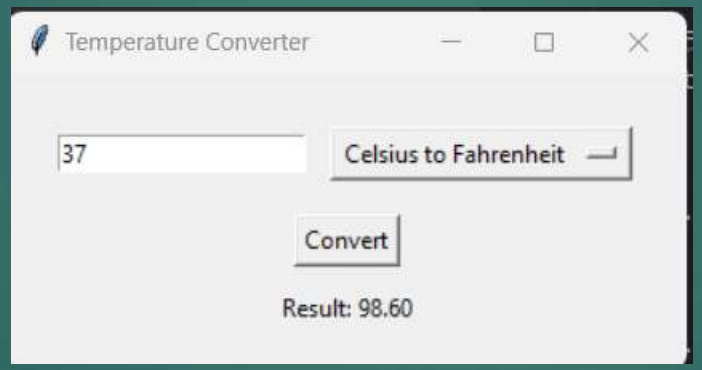


Chapter 2 Bonus-A

```python
import tkinter as tk
# defining the function
1 usage
def temperature():
    try:
        temperature = float(entry.get())
        converted_temperature = 0

        if temp_var.get() == "Celsius to Fahrenheit":
            converted_temperature = (temperature * 9/5) + 32
        elif temp_var.get() == "Fahrenheit to Celsius":
            converted_temperature = (temperature - 32) * 5/9

        result_label.config(text=f"Result: {converted_temperature:.2f}")
    except ValueError:
        result_label.config(text="Please enter a valid number")

root = tk.Tk()
root.title("Temperature Converter")

frame = tk.Frame(root)
frame.pack(padx=20, pady=20)

entry = tk.Entry(frame)
entry.grid(row=0, column=0, padx=5, pady=5)

temp_var = tk.StringVar(root)
options = ["Celsius to Fahrenheit", "Fahrenheit to Celsius"]
operation_menu = tk.OptionMenu(frame, temp_var, *values *options)
operation_menu.grid(row=0, column=1, padx=5, pady=5)
temp_var.set(options[0])
```

```python
convert_button = tk.Button(frame, text="Convert", command=temperature)
convert_button.grid(row=1, columnspan=2, padx=5, pady=10)

result_label = tk.Label(frame, text="Result: ")
result_label.grid(row=2, columnspan=2)

root.mainloop()
```

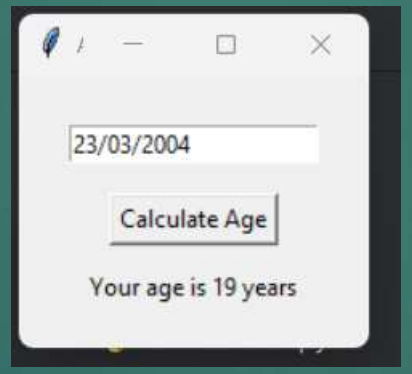
Chapter 2 Bonus-A Output



Chapter 2 Bonus-B

```python
# importing libraries
import tkinter as tk
from datetime import datetime

# defining the function
1usage
def calculate_age():
    try:
        dob = datetime.strptime(entry.get(), _format: "%d/%m/%Y")
        today = datetime.today()
        age = today.year - dob.year - ((today.month, today.day) < (dob.month, dob.day))
        result_label.config(text=f"Your age is {age} years")
    except ValueError:
        result_label.config(text="Please enter date in DD/MM/YYYY format")

root = tk.Tk()
root.title("Age Calculator")

frame = tk.Frame(root)
frame.pack(padx=20, pady=20)

entry = tk.Entry(frame)
entry.grid(row=0, column=0, padx=5, pady=5)

calculate_button = tk.Button(frame, text="Calculate Age", command=calculate_age)
calculate_button.grid(row=1, padx=5, pady=10)

result_label = tk.Label(frame, text="Your age is: ")
result_label.grid(row=2)
# Running the main loop
root.mainloop()
```

Chapter 2 bonus-B Output

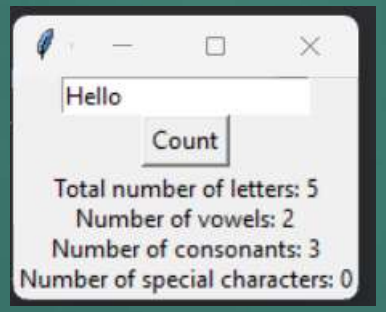


Chapter 2 Further Ex 1

```python
# importing tkinter
import tkinter as tk
# define the function
1 usage
def count():
    user_input = entry.get()
    vowels = 0
    consonants = 0
    characters = 0

# statements
    for char in user_input:
        if char.isalpha():
            if char.lower() in 'aeiou':
                vowels += 1
            else:
                consonants += 1
        else:
            characters += 1

    result_label.config(text=f"Total number of letters: {len(user_input)}\nNumber of vowels: "
                        f"{vowels}\nNumber of consonants: {consonants}\nNumber of special characters: {characters}")

root = tk.Tk()
root.title("Counter")

entry = tk.Entry(root)
entry.pack()

count_button = tk.Button(root, text="Count", command=count)
count_button.pack()
```

```python
count_button = tk.Button(root, text="Count", command=count)
count_button.pack()

result_label = tk.Label(root)
result_label.pack()

# Run the GUI
root.mainloop()
```

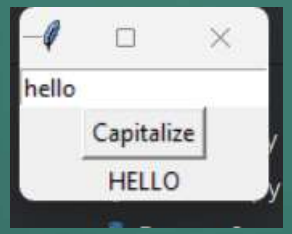
Chapter 2 Further Ex-1 Output



Chapter 2 Further Ex-2

```python
# importing tkinter
import tkinter as tk

1 usage
def capital_letters():
    user_input = entry.get() # getting input from the user
    capital_text = user_input.upper()

    result_label.config(text=capital_text)

root= tk.Tk()
root.title("Capital Letters")

entry = tk.Entry(root)
entry.pack()

#creating button
capital_button = tk.Button(root, text="Capitalize", command=capital_letters)
capital_button.pack()

# creating label
result_label = tk.Label(root)
result_label.pack()

root.mainloop()
```

Chapter 2 Further Ex-2 Output



Chapter-3

CHAPTER 3 - GRAPHICAL USER INTERFACE(CONT.)

Chapter 3 Ex-1

```python
# importing tkinter
import tkinter as tk

1 usage
def greeting():
    name = name_entry.get() # getting input from the user
    selected_color = color_var.get()
    greeting_label.config(text=f"Hello, {name}!", fg=selected_color)

root = tk.Tk()
root.title("Greeting App")  # title for the main window

# InputFrame as per the requirement
input_frame = tk.Frame(root, bg="blue", padx=10, pady=10)
input_frame.pack(padx=10, pady=10, side=tk.LEFT)

title_label = tk.Label(input_frame, text="Greeting App", fg="orange", font=("Arial", 16), pady=10)
title_label.pack()

name_label = tk.Label(input_frame, text="Enter your name:")
name_label.pack()

name_entry = tk.Entry(input_frame)
name_entry.pack()

color_label = tk.Label(input_frame, text="Select text color:")
color_label.pack()

colors = ['black', 'red', 'green', 'blue', 'purple']  # Option for colors
color_var = tk.StringVar(root)
color_var.set(colors[0])  # Set default color(black)
```

```python
dropdown = tk.OptionMenu(input_frame, color_var, *values: *colors)
dropdown.pack()

update_button = tk.Button(input_frame, text="Update Greeting", command=greeting)
update_button.pack()

# DisplayFrame as per the requirement
display_frame = tk.Frame(root, bg="orange", padx=20, pady=20)
display_frame.pack(padx=10, pady=10, side=tk.RIGHT)

greeting_label = tk.Label(display_frame, text="", font=("Arial", 12))
greeting_label.pack()

# Run the GUI
root.mainloop()
```

Chapter 3 Ex-1 Output

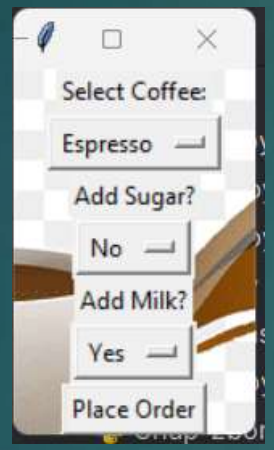
Chapter 3 Ex-2

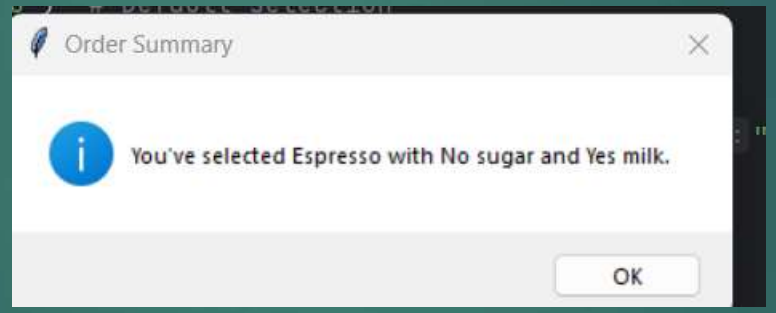
```python
# importing libraries
import tkinter as tk
from tkinter import messagebox

# the menu or order that will be displayed
1usage
def order():
    message = f"You've selected {coffee_var.get()} with {sugar_var.get()} sugar and {milk_var.get()} milk."
    messagebox.showinfo( title: "Order Summary", message)

# Create main window
root = tk.Tk()
root.title("Coffee Vending Machine")

# Load and set background image
bg_image = tk.PhotoImage(file="coffee2.png")  # An image displayed
bg_label = tk.Label(root, image=bg_image)
bg_label.place(relwidth=1, relheight=1)

# Coffee selection
coffee_var = tk.StringVar()
coffee_var.set("Espresso")  # Default selection
coffee_label = tk.Label(root, text="Select Coffee:")
coffee_label.pack()
coffee_menu = tk.OptionMenu(root, coffee_var, value: "Espresso", *values: "Latte", "Cappuccino")  # values
coffee_menu.pack()

# Sugar selection
sugar_var = tk.StringVar()
sugar_var.set("No")  # Default selection
```

Chapter 3 Ex-2

```
# Sugar selection
sugar_var = tk.StringVar()
sugar_var.set("No")  # Default selection
sugar_label = tk.Label(root, text="Add Sugar?")
sugar_label.pack()
sugar_menu = tk.OptionMenu(root, sugar_var, value: "No", *values: "Yes")
sugar_menu.pack()

# Milk selection
milk_var = tk.StringVar()
milk_var.set("No")  # Default selection
milk_label = tk.Label(root, text="Add Milk?")
milk_label.pack()
milk_menu = tk.OptionMenu(root, milk_var, value: "No", *values: "Yes")
milk_menu.pack()

# Submit with the button
submit_button = tk.Button(root, text="Place Order", command=order)
submit_button.pack()

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

Chapter 3 Ex-2 Output





Chapter 3 Ex-3

```python
# import tkinter
import tkinter as tk
from tkinter import ttk
# import math library
import math
# defining the functions
1 usage
def circle():
    radius = float(circle_radius_entry.get())
    area = math.pi * radius**2
    circle_result_label.config(text=f"Area of Circle: {area:.2f} sq units")

1 usage
def square():
    side_length = float(square_side_entry.get())
    area = side_length**2
    square_result_label.config(text=f"Area of Square: {area:.2f} sq units")

1 usage
def rectangle():
    length = float(rectangle_length_entry.get())
    width = float(rectangle_width_entry.get())
    area = length * width
    rectangle_result_label.config(text=f"Area of Rectangle: {area:.2f} sq units")

root = tk.Tk()
root.title("Shapes")

# Creating Tabbed Interface
notebook = ttk.Notebook(root)
notebook.pack(padx=10, pady=10)
```

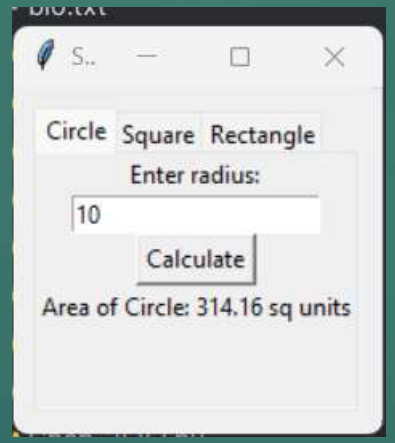
Chapter 3 Ex-3

```
# Circle Tab
circle_tab = ttk.Frame(notebook)
notebook.add(circle_tab, text='Circle')

circle_radius_label = tk.Label(circle_tab, text="Enter radius:")
circle_radius_label.pack()

circle_radius_entry = tk.Entry(circle_tab)
circle_radius_entry.pack()

circle_calculate_button = tk.Button(circle_tab, text="Calculate", command=circle)
circle_calculate_button.pack()

circle_result_label = tk.Label(circle_tab, text="")
circle_result_label.pack()

# Square Tab
square_tab = ttk.Frame(notebook)
notebook.add(square_tab, text='Square')

square_side_label = tk.Label(square_tab, text="Enter side length:")
square_side_label.pack()

square_side_entry = tk.Entry(square_tab)
square_side_entry.pack()

square_calculate_button = tk.Button(square_tab, text="Calculate", command=square)
square_calculate_button.pack()
```

Chapter 3 Ex-3

```
square_calculate_button = tk.Button(square_tab, text="Calculate", command=square)
square_calculate_button.pack()

square_result_label = tk.Label(square_tab, text="")
square_result_label.pack()

# Rectangle Tab
rectangle_tab = ttk.Frame(notebook)
notebook.add(rectangle_tab, text='Rectangle')

rectangle_length_label = tk.Label(rectangle_tab, text="Enter length:")
rectangle_length_label.pack()

rectangle_length_entry = tk.Entry(rectangle_tab)
rectangle_length_entry.pack()

rectangle_width_label = tk.Label(rectangle_tab, text="Enter width:")
rectangle_width_label.pack()

rectangle_width_entry = tk.Entry(rectangle_tab)
rectangle_width_entry.pack()_# using pack

rectangle_calculate_button = tk.Button(rectangle_tab, text="Calculate", command=rectangle)_# button
rectangle_calculate_button.pack()

rectangle_result_label = tk.Label(rectangle_tab, text="")
rectangle_result_label.pack()_# using pack

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

Chapter 3 Ex-3 Output



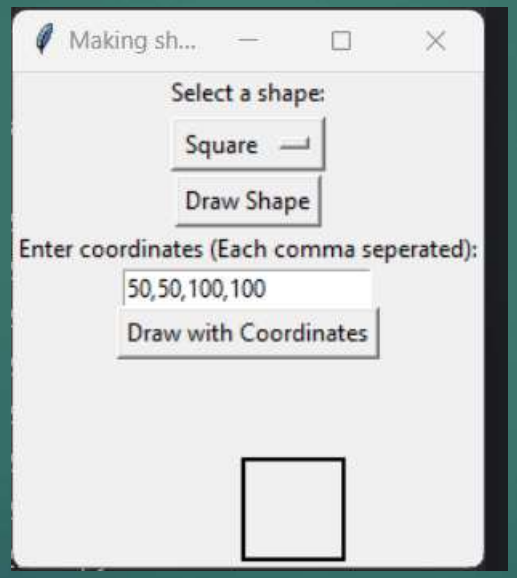
Chapter 3 Ex-4

```python
# importing tkinter
import tkinter as tk
# define the function
1 usage
def shapes():
    canvas.delete("all")
    shape = shape_var.get()
# nested if statements
    if shape== "Oval":
        canvas.create_oval(50, 50, 150, 100, outline="black", width=2)
    elif shape == "Rectangle":
        canvas.create_rectangle(50, 50, 150, 100, outline="black", width=2)
    elif shape == "Square":
        canvas.create_rectangle(50, 50, 100, 100, outline="black", width=2)
    elif shape == "Triangle":
        canvas.create_polygon(50, 100, 100, 50, 150, 100, outline="black", width=2)
1 usage
def coordinates():
    canvas.delete("all")
    shape = shape_var.get()
    coordinates = coordinates_entry.get().split(',')
# nested if statements
    if shape == "Oval":
        canvas.create_oval(*coordinates, outline="black", width=2)
    elif shape == "Rectangle" or shape == "Square":
        canvas.create_rectangle(*coordinates, outline="black", width=2)
    elif shape == "Triangle":
        canvas.create_polygon(*coordinates, outline="black", width=2)
```

Chapter 3 Ex-4

```python
# main window
root = tk.Tk()
root.title("Making shapes wit the help of coordinates")  # title

shape_var = tk.StringVar(root)
shape_var.set("Select a shape")  # selecting a shape from the options

shape_label = tk.Label(root, text="Select a shape:")
shape_label.pack()

option = tk.OptionMenu(root, shape_var, value: "Oval", *values: "Rectangle", "Square", "Triangle")
option.pack()

button = tk.Button(root, text="Draw Shape", command=shapes)
button.pack()

coordinates_label = tk.Label(root, text="Enter coordinates (Each comma seperated):")
coordinates_label.pack()

coordinates_entry = tk.Entry(root)
coordinates_entry.pack()

draw_coordinates_button = tk.Button(root, text="Draw with Coordinates", command=coordinates)
draw_coordinates_button.pack()

canvas = tk.Canvas(root, width=100, height=100)
canvas.pack()

root.mainloop()
```

Chapter 3 Ex-4 Output



Chapter 3 Bonus Ex-1

```python
#  Burger Shack Vendor

# Defining Function for displaying the menu
1 usage
def 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():
    #  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]
```

```python
    # Adding toppings as required
    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 as required
    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 as required
    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
        return order
    except (ValueError, IndexError):_# Codes for errors
        print("Invalid input. Please enter a valid choice.")

# function for calculation
1usage
def calculate_total(order):
    # Assuming the fixed costs for items
    burger_cost = 6.00
    topping_cost = 2.50
    condiment_cost = 2.00
    side_cost = 3.50

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

    return cost


#  payment process
1usage
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: "))
```

```python
        # if statement
        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.")

# function for ordering process
def main():
    # running the main function
    menu()
    order = place_order()
    total_cost = calculate_total(order)
    process_payment(total_cost)

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

Chapter 3 Bonus Output

```
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): 2
   Topping (1-3): 0

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

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

Total Cost: 14.00 AED
Enter Amount Paid: 12
Insufficient payment. Please provide enough funds.

Process finished with exit code 0
```

Chapter 3 Further Ex-1

```
# ******************************************VENDING MACHINE*********************************
# ******************************************************************************************
import time

print("\033[1;33;40m WELCOME TO THE VENDING MACHINE")  # adding yellow color
# input
number_of_dirhams = int(input("How many  dirhams would you like to insert? "))

# money inserted
change = round(number_of_dirhams)

print("\nYou have entered dirhams", change)
time.sleep(2)
# items with prices.

item_1 = "Coffee"
price_1 = 4
item_2 = "biscuits"
price_2 = 3.50
item_3 = "mango juice"
price_3 = 3
item_4 = "cup cake"
price_4 = 2.50
item_5 = "chips"
price_5 = 1.75
item_6 = "chocolate"
price_6 = 2.15
```

```python
# item purchased
coffee_purchased = 0
biscuits_purchased = 0
mango_juice_purchased = 0
cup_cake_purchased = 0
chips_purchased = 0
chocolate_purchased = 0

print("There are 6 items available to pick from;\n")
time.sleep(2)
print(" Item                    Prices")
print(" ----                    -----")
print(" {}         ----->  {} ".format( *args: item_1, price_1))
print(" {}  ----->  {} ".format( *args: item_2, price_2))
print(" {}     ----->  {} ".format( *args: item_3, price_3))
print(" {}       ----->  {} ".format( *args: item_4, price_4))
print(" {}    ----->  {} ".format( *args: item_5, price_5))
print(" {}          ----->  {} ".format( *args: item_6, price_6))
print("")

while change > 0:
    customers_selection = input("What would you like to buy? Type N when you are finished \n")
    if customers_selection == "Coffee" or customers_selection == "coffee" and change >= price_1:
        print("You have selected a", item_1, "these cost", price_1, "each,")
        change = round((change - price_1), 2)
        coffee_purchased = (coffee_purchased + 1)

        print("Your balance remaining : dirhams", change)
```

```python
elif customers_selection == "biscuits" or customers_selection == "biscuits" and change >= price_2:
    print("You have selected a", item_2, "these cost", price_2, "each.")
    change = round((change - price_2), 2)
    biscuits_purchased = (biscuits_purchased + 1)

    print("Your balance remaining : dirhams", change)


elif customers_selection == "mango juice" or customers_selection == "mango juice" and change >= price_3:
    print("You have selected a", item_3, "these cost", price_3, "each.")
    change = round((change - price_3), 2)
    mango_juice_purchased = (mango_juice_purchased + 1)

    print("Your balance remaining : dirhams", change)


elif customers_selection == "cup cake" or customers_selection == "cup cake" and change >= price_4:
    print("You have selected a", item_4, "these cost", price_4, "each.")
    change = round((change - price_4), 2)
    cup_cake_purchased = (cup_cake_purchased + 1)

    print("Your balance remaining ", change)


elif customers_selection == "Chips" or customers_selection == "chips" and change >= price_5:
    print("You have selected a", item_5, "these cost", price_5, "each.")
    change = round((change - price_5), 2)
    chips_purchased = (chips_purchased + 1)

    print("Your balance remaining ", change)
```

```python
    elif customers_selection == "chocolate" or customers_selection == "chocolate" and change >= price_6:
        print("You have selected a", item_6, "these cost", price_6, "each.")
        change = round((change - price_6), 2)
        chocolate_purchased = (chocolate_purchased + 1)

        print("Your balance remaining ", change)


    elif customers_selection == "N" or customers_selection == "n":

        break

    elif change <= 0:
        print("You have insufficient balance.")

        break
    else:
        print("There has been an error or you may not have enough credit.")

print("\nYou have dirhams", change, "remaining.")
time.sleep(2)
# receipt
print("\nHere is your receipt;")
time.sleep(2)
if coffee_purchased > 0:
    print("You purchased", coffee_purchased, item_1, "at dirhams", (price_1 * coffee_purchased))
if biscuits_purchased > 0:
    print("You purchased", biscuits_purchased, item_2, "at dirhams", (price_2 * biscuits_purchased))
if mango_juice_purchased > 0:
    print("You purchased", mango_juice_purchased, item_3, "at dirhams", (price_3 * mango_juice_purchased))
if cup_cake_purchased > 0:
```

```
if cup_cake_purchased > 0:
    print("You purchased", cup_cake_purchased, item_4, "at dirhams", (price_4 * cup_cake_purchased))
if chips_purchased > 0:
    print("You purchased", chips_purchased, item_5, "at dirhams", (price_5 * chips_purchased))
if chocolate_purchased > 0:
    print("You purchased", chocolate_purchased, item_6, "at dirhams", (price_6 * chocolate_purchased))
print("\033[1;33;40m")  # adding yellow color
print("\033[1;33;40m THANKS FOR USING THE VENDING MACHINE")
```

Chapter 3 Further Ex 1 Output

```
 WELCOME TO THE VENDING MACHINE
How many  dirhams would you like to insert? 5

You have entered dirhams 5
There are 6 items available to pick from;

 Item                  Prices
 ----                  -----
 Coffee         -----> 4
 biscuits  -----> 3.5
 mango juice     -----> 3
 cup cake       -----> 2.5
 chips    -----> 1.75
 chocolate          -----> 2.15

What would you like to buy? Type N when you are finished
chips
You have selected a chips these cost 1.75 each,
Your balance remaining  3.25
What would you like to buy? Type N when you are finished
n

You have dirhams 3.25 remaining.

Here is your receipt;
You purchased 1 chips at dirhams 1.75

 THANKS FOR USING THE VENDING MACHINE
```

Chapter 3 Further Ex-2

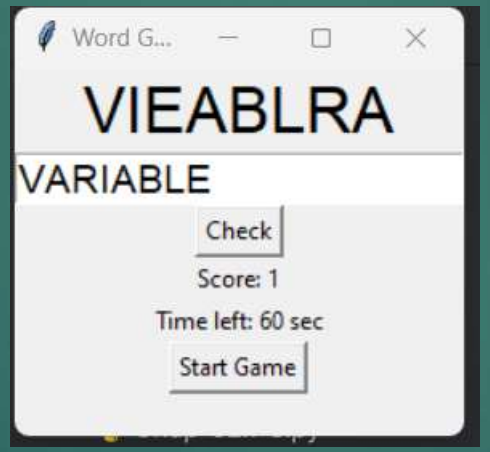
```python
# import tkinter
import tkinter as tk
import random

# List of words for the game
words = ["PYTHON", "COMPUTER", "PROGRAMMING", "DEVELOPMENT", "INTERFACE", "ALGORITHM", "VARIABLE"]

# Function to shuffle the words
1 usage
def shuffle_word(word):
    shuffled = list(word)
    random.shuffle(shuffled)
    return ''.join(shuffled)

# Function to start the game
1 usage
def start_game():
    global current_word, score, remaining_time
    score = 0
    remaining_time = 60  # Change the time here (in seconds)
```

```python
def start_game():
    global current_word, score, remaining_time
    score = 0
    remaining_time = 60  # Change the time here (in seconds)
    update_score()
    next_word()

2 usages
def next_word():
    global current_word
    if len(words) > 0:
        current_word = random.choice(words)
        words.remove(current_word)
        shuffled_word = shuffle_word(current_word)
        lbl_word.config(text=shuffled_word)
    else:
        end_game()

1 usage
def check_answer():
    global score
    user_answer = entry_guess.get().upper()
    if user_answer == current_word:
        score += 1
        update_score()
        entry_guess.delete( first: 0, tk.END)
        next_word()
```

```python
def update_score():
    lbl_score.config(text=f"Score: {score}")

2 usages
def update_time():
    global remaining_time
    if remaining_time > 0:
        remaining_time -= 1
        lbl_timer.config(text=f"Time left: {remaining_time} sec")
        root.after( ms: 1000, update_time)
    else:
        end_game()

2 usages
def end_game():
    lbl_word.config(text="Game Over!")
    entry_guess.config(state="disabled")
    btn_check.config(state="disabled")
    lbl_final_score.config(text=f"Final Score: {score}")

# Create the tkinter window
root = tk.Tk()
root.title("Word Guessing Game")

# Create and pack GUI elements
lbl_word = tk.Label(root, text="", font=("Arial", 24))
lbl_word.pack()

entry_guess = tk.Entry(root, font=("Arial", 14))
entry_guess.pack()
```

```python
# Create the tkinter window
root = tk.Tk()
root.title("Word Guessing Game")

# Create and pack GUI elements
lbl_word = tk.Label(root, text="", font=("Arial", 24))
lbl_word.pack()

entry_guess = tk.Entry(root, font=("Arial", 14))
entry_guess.pack()

btn_check = tk.Button(root, text="Check", command=check_answer)
btn_check.pack()

lbl_score = tk.Label(root, text="Score: 0")
lbl_score.pack()

lbl_timer = tk.Label(root, text="Time left: 60 sec")
lbl_timer.pack()

btn_start = tk.Button(root, text="Start Game", command=start_game)
btn_start.pack()

lbl_final_score = tk.Label(root, text="")
lbl_final_score.pack()

# Start the tkinter main loop
root.after( ms: 1000, update_time)
root.mainloop()
```

Chapter 3 Further Ex-2 Output



Chapter-4

CHAPTER 4 - FILE HANDLING AND REGULAR EXPRESSIONS

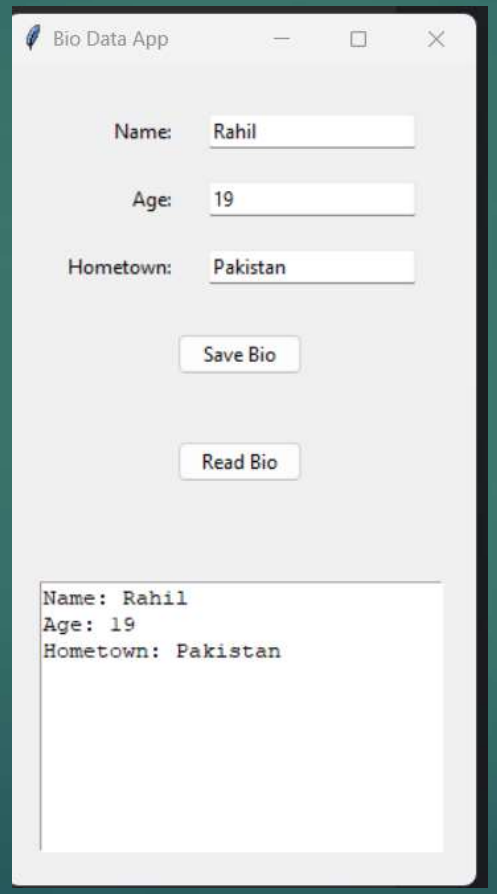
Chapter 4 Ex-1

```python
import tkinter as tk
from tkinter import ttk

def bio():
    # Get input from the user
    name = entry_name.get()
    age = entry_age.get()
    hometown = entry_hometown.get()

    # Write data to the file created named as bio.txt
    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")

2 usages
def bio():
    # Read 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)  # Clear previous output
        text_output.insert(tk.END, bio_data)
    except FileNotFoundError:
        text_output.delete( index1: 1.0, tk.END)
        text_output.insert(tk.END,  chars: "Bio file not found. Please save your bio first.")
```

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

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

# Create 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)

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)

# Create buttons to save and read bio
button_save = ttk.Button(frame, text="Save Bio", command=bio)
button_save.grid(row=3, column=0, columnspan=2, pady=20)
button_read = ttk.Button(frame, text="Read Bio", command=bio)
button_read.grid(row=4, column=0, columnspan=2, pady=20)
```

```python
# Create a text widget to display bio data
text_output = tk.Text(root, width=30, height=10)
text_output.grid(row=1, column=0, padx=20, pady=20)

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

Chapter 4 Ex-1 Output



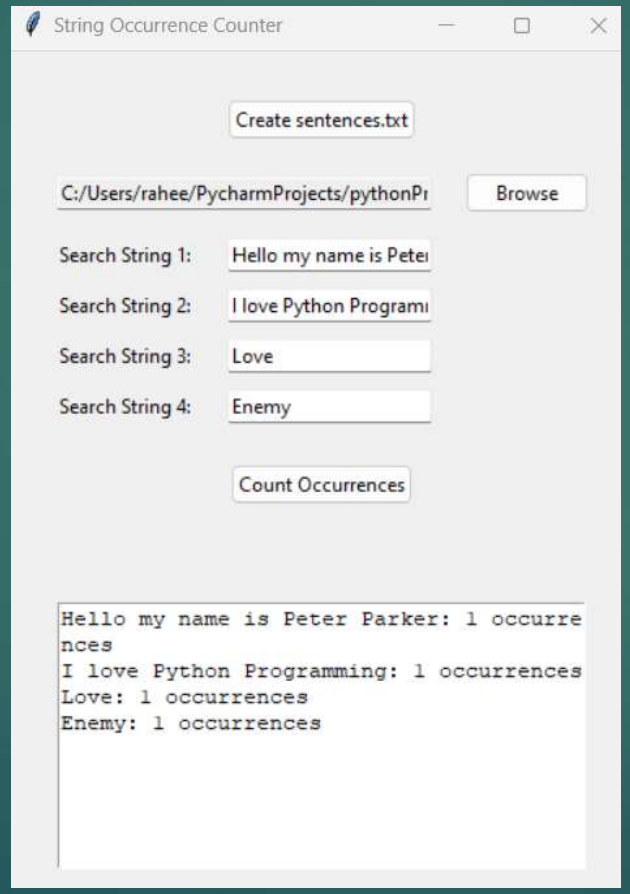
Chapter 4 Ex-2

```python
# importing the libraries
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog

1usage
def sentences_file():
    # Create 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)

1usage
def occurrences():
    # Get the selected file
    file_path = file_var.get()

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

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

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

```python
        # Display 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")

    except FileNotFoundError:
        text_output.delete( index1: 1.0, tk.END)
        text_output.insert(tk.END,  chars: "File not found. Please create sentences.txt first.")

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

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

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

# Create 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")

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

```python
# Create 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)

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

# Create 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)

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

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

Output


String Occurrence Counter
Create sentences.txt
C:/Users/rahee/PycharmProjects/pythonPr
Browse
Search String 1:
Hello my name is Pete
Search String 2:
I love Python Programi
Search String 3:
Love
Search String 4:
Enemy
Count Occurrences
Hello my name is Peter Parker: 1 occurre
nces
I love Python Programming: 1 occurrences
Love: 1 occurrences
Enemy: 1 occurrences

Chapter 4 Ex-3

```python
# Open the file in read mode
# file name with the name of numbers.txt and open it in the read mode
with open('numbers.txt', 'r') as file:
    # Read all lines from the file and convert each line to an integer
    numbers = [int(line.strip()) for line in file]

# Get the output in integer format
for number in numbers:
    print(number) # printing the numbers
```

```
1
2
3
4
5
6
7
8
9
0
10
```

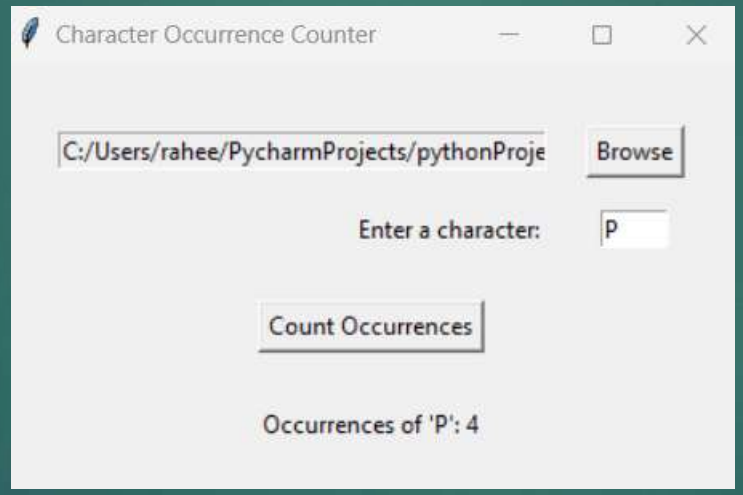
Chapter 4 Ex-4

```python
import tkinter as tk
from tkinter import filedialog

1usage
def occurrences():
    # Get the selected file
    file_path = file_var.get()

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

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

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

        # Display 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.")

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

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

# Create 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)

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

# Create 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)

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

# Label for the results
label = tk.Label(frame, text="")
label.grid(row=3, column=0, columnspan=2, pady=5)
```

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

# Run program
root.mainloop()
```

Output


Character Occurrence Counter
C:/Users/rahee/PycharmProjects/pythonProje
Browse
Enter a character:
P
Count Occurrences
Occurrences of 'P': 4

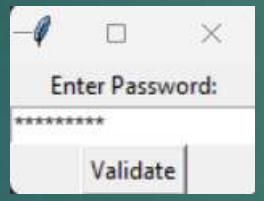
Chapter 4 Ex-5

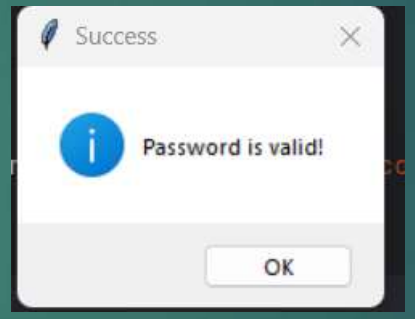
```python
# import tkinter
import tkinter as tk
from tkinter import messagebox

1 usage
def password():
    password = password_entry.get()

    # Password criteria as given in the excercise
    lowercase = any(c.islower() for c in password)
    uppercase = any(c.isupper() for c in password)
    digits = any(c.isdigit() for c in password)
    special_char = any(c in '$#@' for c in password)
    length = 6 <= len(password) <= 12

    # nested if statements to check whether all criteria are fulfilled
    if lowercase and uppercase and digits and special_char and length:
        messagebox.showinfo( title: "Success",  message: "Password is valid!")
        root.destroy()
    else:
        attempts_left[0] -= 1
        if attempts_left[0] > 0:
            messagebox.showwarning( title: "Invalid Password",
                                    message: f"Invalid password. Attempts left: {attempts_left[0]}")
        else:
            messagebox.showerror( title: "Alert!",  message: "Authorities have been alerted!")
```

```python
# import tkinter
import tkinter as tk
from tkinter import messagebox

1 usage
def password():
    password = password_entry.get()

    # Password criteria as given in the excercise
    lowercase = any(c.islower() for c in password)
    uppercase = any(c.isupper() for c in password)
    digits = any(c.isdigit() for c in password)
    special_char = any(c in '$#@' for c in password)
    length = 6 <= len(password) <= 12

    # nested if statements to check whether all criteria are fulfilled
    if lowercase and uppercase and digits and special_char and length:
        messagebox.showinfo( title: "Success",  message: "Password is valid!")
        root.destroy()
    else:
        attempts_left[0] -= 1
        if attempts_left[0] > 0:
            messagebox.showwarning( title: "Invalid Password",
                                    message: f"Invalid password. Attempts left: {attempts_left[0]}")
        else:
            messagebox.showerror( title: "Alert!",  message: "Authorities have been alerted!")
```

```python
        else:
            messagebox.showerror( title: "Alert!",  message: "Authorities have been alerted!")
            root.destroy()


# The password attempts details
attempts_left = [5]

root = tk.Tk()
root.title("Password Validator")

# Creating the label for the password
password_label = tk.Label(root, text="Enter Password:")
password_label.pack()

# creating asterisk for hiding the input from the user
password_entry = tk.Entry(root, show='*')
password_entry.pack()

# button and command
validate_button = tk.Button(root, text="Validate", command=password)
validate_button.pack()

root.mainloop()
```

Output





Chapter 4 Bonus Ex-1

```python
# import tkinter
import tkinter as tk


def calculate():
    file_path = 'petrolPrice.txt'
    total_cost = 0
    total_liters = 0
    count_under_3_5 = 0

    with open(file_path, 'r') as file:
        next(file)  # Skip the header
        for line in file:
            liters, cost = map(float, line.split('\t'))
            total_cost += cost
            total_liters += liters
            if cost / liters < 3.5:
                count_under_3_5 += liters

    # Calculation 1: Cost of petrol per liter bought
    cost_per_liter = total_cost / total_liters if total_liters != 0 else 0

    # Calculation 2: Overall average price per liter of petrol bought
    overall_average = total_cost / total_liters if total_liters != 0 else 0

    # Calculation 3: Petrol bought at under 3.5 AED per liter
    liters_under_3_5 = count_under_3_5
```

```python
    # Calculation 3: Petrol bought at under 3.5 AED per liter
    liters_under_3_5 = count_under_3_5

    # Displaying the results in a simple GUI
    result_text = f"Cost per liter bought: {cost_per_liter:.2f} AED\n"
    result_text += f"Overall average price per liter: {overall_average:.2f} AED\n"
    result_text += f"Liters bought at under 3.5 AED per liter: {liters_under_3_5:.2f} liters"

    result_label.config(text=result_text)


# Creating the GUI
root = tk.Tk()
root.title("Petrol Price Calculator")

calculate_button = tk.Button(root, text="Calculate", command=calculate, bg='green')
calculate_button.pack()

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

root.mainloop()
```

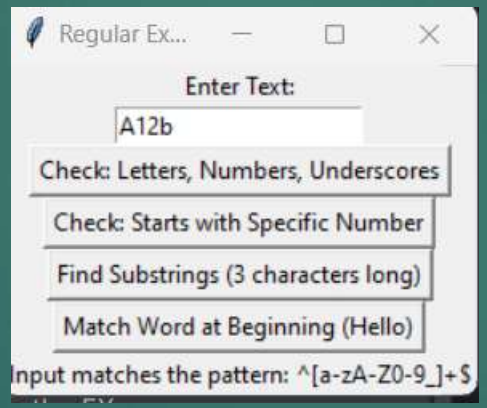
Chapter 4 Bonus Output

Chapter 4 Further Ex

```python
# importing tkinter and regular expressions module
import tkinter as tk
import re

# defining the options
1 usage
def numbers_underscore():
    pattern = r'^[a-zA-Z0-9_]+$'
    check_pattern(pattern)

1 usage
def specific_number():
    pattern = r'^7\d+'
    check_pattern(pattern)

1 usage
def substrings():
    pattern = r'\b\w{3}\b'
    check_pattern(pattern)

1 usage
def beginning():
    pattern = r'^Hello\b'
    check_pattern(pattern)

4 usages
def check_pattern(pattern):
    user_input = entry.get()
    match = re.search(pattern, user_input)
    if match:
        result_label.config(text=f"Input matches the pattern: {pattern}")
```

```python
    else:
        result_label.config(text=f"Input does not match the pattern: {pattern}")

root = tk.Tk()
root.title("Regular Expression")

# creating label for the entry of data
entry_label = tk.Label(root, text="Enter Text:")
entry_label.pack()

entry = tk.Entry(root)
entry.pack()

# creating buttons for each operation
underscore_button = tk.Button(root, text="Check: Letters, Numbers, Underscores", command=numbers_underscore)
underscore_button.pack()

specific_number_button = tk.Button(root, text="Check: Starts with Specific Number", command=specific_number)
specific_number_button.pack()

substrings_button = tk.Button(root, text="Find Substrings (3 characters long)", command=substrings)
substrings_button.pack()

beginning_button = tk.Button(root, text="Match Word at Beginning (Hello)", command=beginning)
beginning_button.pack()

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

root.mainloop()
```

Chapter 4 Further Ex Output



Chapter-5

CHAPTER 5 - OBJECT ORIENTED PROGRAMMING

Chapter 5 Ex-1

```python
import tkinter as tk


2 usages
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    1 usage
    def woof(self):
        return f"{self.name} says woof!"


# Create two dog objects
dog1 = Dog( name: "German Shephard", age: 10)
dog2 = Dog( name: "Husky", age: 8)

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


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

```python
    1 usage
    def display_dog_info(self):
        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()
        oldest_dog_label = tk.Label(self, text=f"The oldest dog is {oldest_dog.name}")
        oldest_dog_label.pack()
        woof_button = tk.Button(self, text="Make the oldest dog woof", command=self.woof)
        woof_button.pack()

    1 usage
    def woof(self):
        result = oldest_dog.woof()
        woof_label = tk.Label(self, text=result)
        woof_label.pack()


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

Chapter 5 Ex-1 Output

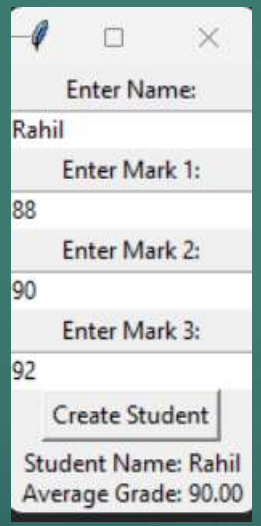
Chapter 5 Ex-2

```python
# importing tkinter
import tkinter as tk

# creating main class called students
1usage
class Students:
    def __init__(self, name, mark1, mark2, mark3):
        self.name = name
        self.mark1 = mark1
        self.mark2 = mark2
        self.mark3 = mark3
# calculated grade to calculate the average combining marks
    1usage
    def calcGrade(self):
        average = (self.mark1 + self.mark2 + self.mark3) / 3
        return average
# display function
    1usage
    def display(self):
        grade = self.calcGrade()
        return f"Student Name: {self.name}\nAverage Grade: {grade:.2f}"

# getting the entry from the user
1usage
def student():
    name = name_entry.get()
    mark1 = int(mark1_entry.get())
    mark2 = int(mark2_entry.get())
    mark3 = int(mark3_entry.get())
```

```python
    student = Students(name, mark1, mark2, mark3)
    result_label.config(text=student.display())

# tkinter
root = tk.Tk()
root.title("Students")

# creating label
name_label = tk.Label(root, text="Enter Name:")
name_label.pack()

# name and marks entry
name_entry = tk.Entry(root)
name_entry.pack()_# using pack

mark1_label = tk.Label(root, text="Enter Mark 1:")
mark1_label.pack()_# using pack

mark1_entry = tk.Entry(root)
mark1_entry.pack()_# using pack

mark2_label = tk.Label(root, text="Enter Mark 2:")
mark2_label.pack()_# using pack

mark2_entry = tk.Entry(root)
mark2_entry.pack()_# using pack

mark3_label = tk.Label(root, text="Enter Mark 3:")
mark3_label.pack()_# using pack
```

```python
mark3_entry = tk.Entry(root)
mark3_entry.pack() # using pack

# creating button and giving command
student_button = tk.Button(root, text="Create Student", command=student)
student_button.pack() # using pack

result_label = tk.Label(root, text="")
result_label.pack() # using pack

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

Chapter 5 Ex-2 Output



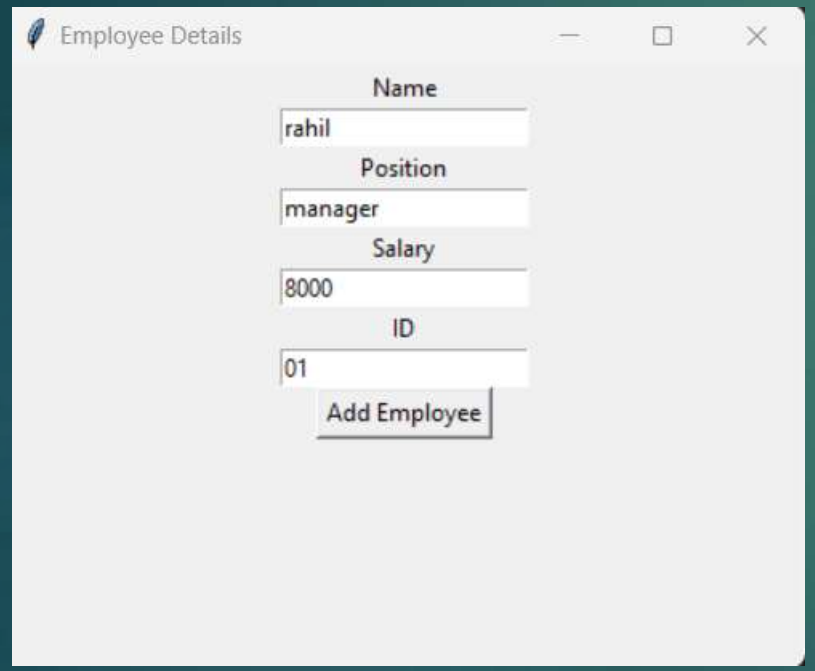
Chapter 5 Ex-3

```python
import tkinter as tk

1 usage
class Employee:
    def __init__(self):
        self.name = ""
        self.position = ""
        self.salary = 0.0
        self.id = ""

    1 usage
    def setData(self, name, position, salary, emp_id):
        self.name = name
        self.position = position
        self.salary = salary
        self.id = emp_id

    1 usage
    def getData(self):
        return f"{self.name}\t{self.position}\t{self.salary}\t{self.id}"


1 usage
class EmployeeGUI(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Employee Details")
        self.geometry("400x300")
        self.employees_added = 0
        self.create_employee_data_fields()
```

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

1 usage
def create_employee_data_fields(self):
    self.name_label = tk.Label(self, text="Name")
    self.name_label.pack()
    self.name_entry = tk.Entry(self)
    self.name_entry.pack()

    self.position_label = tk.Label(self, text="Position")
    self.position_label.pack()
    self.position_entry = tk.Entry(self)
    self.position_entry.pack()

    self.salary_label = tk.Label(self, text="Salary")
    self.salary_label.pack()
    self.salary_entry = tk.Entry(self)
    self.salary_entry.pack()

    self.id_label = tk.Label(self, text="ID")
    self.id_label.pack()
    self.id_entry = tk.Entry(self)
    self.id_entry.pack()
```

```
1 usage
def add_employee(self):
    if self.employees_added >= 5:
        print("Maximum number of employees reached (5 employees).")
    else:
        name = self.name_entry.get()
        position = self.position_entry.get()
        salary = float(self.salary_entry.get())
        emp_id = self.id_entry.get()

        employee = Employee()
        employee.setData(name, position, salary, emp_id)
        print("Employee Added:")
        print(employee.getData())

        self.employees_added += 1


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

Chapter 5 Ex-3 Output




```
Employee Added:
rahil   manager 8000.0  01
```

Chapter 5 Ex-4

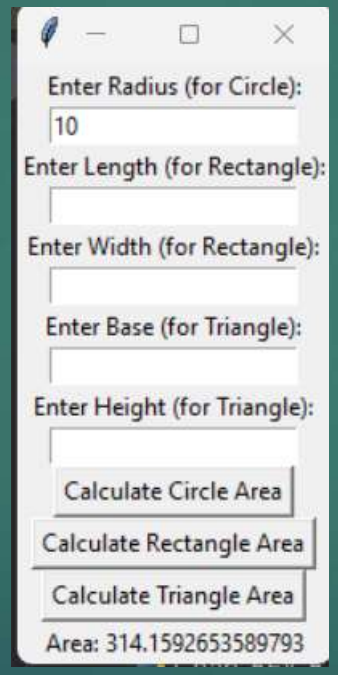
```
# importing tkinter and math library
import ...
# creating a main class in which many subclasses are added
3 usages
class Shape:
    def __init__(self):
        self.sides = []

    1 usage (1 dynamic)
    def inputSides(self):
        pass  # subclass

    1 usage (1 dynamic)
    def area(self):
        pass  # subclass

1 usage
class Circle(Shape):
    1 usage (1 dynamic)
    def inputSides(self):
        radius = float(radius_entry.get())
        self.sides.append(radius)

    1 usage (1 dynamic)
    def area(self):
        radius = self.sides[0]
        return pi * radius**2
```

```
1 usage
class Rectangle(Shape):
    1 usage (1 dynamic)
    def inputSides(self):
        length = float(length_entry.get())
        width = float(width_entry.get())
        self.sides.extend([length, width])

    1 usage (1 dynamic)
    def area(self):
        length, width = self.sides
        return length * width  # formula for calculating

1 usage
class Triangle(Shape):
    1 usage (1 dynamic)
    def inputSides(self):
        base = float(base_entry.get())
        height = float(height_entry.get())
        self.sides.extend([base, height])

    1 usage (1 dynamic)
    def area(self):
        base, height = self.sides
        return 0.5 * base * height

3 usages
```

```
3 usages
def calculate_area(shape):
    shape.inputSides()
    result_label.config(text=f"Area: {shape.area()}")

root = tk.Tk()
root.title("Shapes")

radius_label = tk.Label(root, text="Enter Radius (for Circle):")
radius_label.pack()

radius_entry = tk.Entry(root)
radius_entry.pack()

length_label = tk.Label(root, text="Enter Length (for Rectangle):")
length_label.pack()

length_entry = tk.Entry(root)
length_entry.pack()

width_label = tk.Label(root, text="Enter Width (for Rectangle):")
width_label.pack()

width_entry = tk.Entry(root)
width_entry.pack()

base_label = tk.Label(root, text="Enter Base (for Triangle):")
base_label.pack()
```

```python
base_entry = tk.Entry(root)
base_entry.pack()

height_label = tk.Label(root, text="Enter Height (for Triangle):")
height_label.pack()

height_entry = tk.Entry(root)
height_entry.pack()

# buttons for every shape
circle_button = tk.Button(root, text="Calculate Circle Area",
                          command=lambda: calculate_area(Circle()))
circle_button.pack()

rectangle_button = tk.Button(root, text="Calculate Rectangle Area",
                             command=lambda: calculate_area(Rectangle()))
rectangle_button.pack()

triangle_button = tk.Button(root, text="Calculate Triangle Area",
                            command=lambda: calculate_area(Triangle()))
triangle_button.pack()

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

# running the program
root.mainloop()
```

Chapter 5 Ex-4 Output



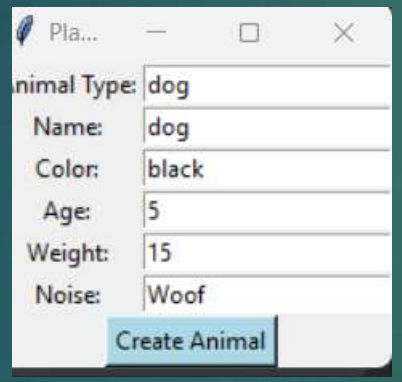
Chapter 5 Ex-5

```python
# importing tkinter as tk
import tkinter as tk

# defining the function Animal as required
1 usage
class Animal:
    def __init__(self, Type, Name, Colour, Age, Weight, Noise):
        self.Type = Type
        self.Name = Name
        self.Colour = Colour
        self.Age = Age
        self.Weight = Weight
        self.Noise = Noise

    1 usage
    def sayHello(self):  # function for saying hello
        print(f"Hello, I am {self.Name}!")

    1 usage
    def makeNoise(self):  # function for making noise
        print(f"{self.Name} says: {self.Noise}")

    2 usages (1 dynamic)
    def animalDetails(self):  # function for animal details
        details = (f"Type: {self.Type}, Name: {self.Name}, Colour: {self.Colour}, Age: {self.Age}, Weight: {self.Weight}, "
                   f"Noise: {self.Noise}")
        print(details)
```

```python
# Creating GUI(graphical user interface)
1usage
def create_animal():
    animal_type = animal_type_entry.get()
    name = name_entry.get()
    color = color_entry.get()
    age = age_entry.get()
    weight = weight_entry.get()
    noise = noise_entry.get()

    animal = Animal(animal_type, name, color, age, weight, noise)
    animal.sayHello()
    animal.makeNoise()
    animal.animalDetails()


root = tk.Tk()
root.title("Playing around in class")

# Labels
tk.Label(root, text="Animal Type:").grid(row=0, column=0)
tk.Label(root, text="Name:").grid(row=1, column=0)
tk.Label(root, text="Color:").grid(row=2, column=0)
tk.Label(root, text="Age:").grid(row=3, column=0)
tk.Label(root, text="Weight:").grid(row=4, column=0)
tk.Label(root, text="Noise:").grid(row=5, column=0)
```

```python
# Entries
animal_type_entry = tk.Entry(root)
name_entry = tk.Entry(root)
color_entry = tk.Entry(root)
age_entry = tk.Entry(root)
weight_entry = tk.Entry(root)
noise_entry = tk.Entry(root)

# creating grid for the entries
animal_type_entry.grid(row=0, column=1)
name_entry.grid(row=1, column=1)
color_entry.grid(row=2, column=1)
age_entry.grid(row=3, column=1)
weight_entry.grid(row=4, column=1)
noise_entry.grid(row=5, column=1)

# Button for creating animal
tk.Button(root, text="Create Animal", command=create_animal,bg='lightblue').grid(row=6, columnspan=2)

root.mainloop()
```

Chapter 5 Ex-5 Output




```
Hello, I am dog!
dog says: Woof
Type: dog, Name: dog, Colour: black, Age: 5, Weight: 15, Noise: Woof
```

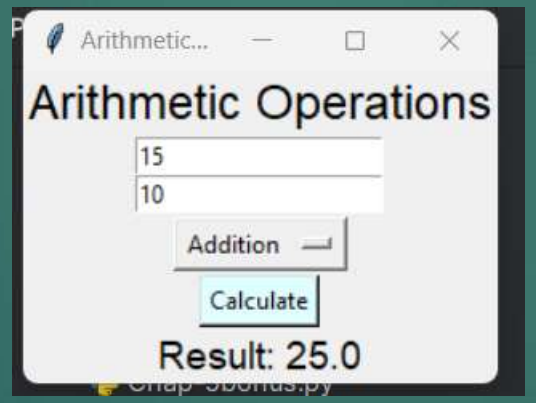
Chapter 5 Ex-6

```python
# importing tkinter and messagebox
import tkinter as tk
from tkinter import messagebox
# creating a main class for the operations
1 usage
class Arithmetic:
    def __init__(self, root):
        self.root = root
        self.root.title("Arithmetic Operations")

        self.result = tk.StringVar()
        self.result.set("Result: ")

        self.widgets()

    # getting the input from the user
    def calculate(self):
        operation = self.operation.get()
        try:
            num1 = float(self.num1.get())
            num2 = float(self.num2.get())

            # nested if statements
            if operation == "Addition":  # if statement
                self.result.set("Result: " + str(num1 + num2))
            elif operation == "Subtraction":
                self.result.set("Result: " + str(num1 - num2))
            elif operation == "Multiplication":
                self.result.set("Result: " + str(num1 * num2))
            elif operation == "Division":
```

```python
        elif operation == "Division":
            if num2 != 0:
                self.result.set("Result: " + str(num1 / num2))
            else:
                self.result.set("Result: Undefined")
                messagebox.showerror( title: "Error",  message: "Cannot divide by zero!")
    except ValueError:
        messagebox.showerror( title: "Error",  message: "Please enter valid numbers.")

1 usage
def widgets(self):
    tk.Label(self.root, text="Arithmetic Operations", font=("Arial", 18)).pack()

    self.num1 = tk.Entry(self.root)
    self.num1.pack()  # using pack

    self.num2 = tk.Entry(self.root)
    self.num2.pack()_# using pack

    operations = ["Addition", "Subtraction", "Multiplication", "Division"]_# the operations available
    self.operation = tk.StringVar()
    self.operation.set(operations[0])

    operation_menu = tk.OptionMenu(self.root, self.operation,  *values: *operations)
    operation_menu.pack()

    # creating a button for calculating
    tk.Button(self.root, text="Calculate", bg='lightcyan', command=self.calculate).pack()
```

```python
        operation_menu = tk.OptionMenu(self.root, self.operation, *values: *operations)
        operation_menu.pack()

        # creating a button for calculating
        tk.Button(self.root, text="Calculate", bg='lightcyan', command=self.calculate).pack()

        # creating a label for the text size and font style
        tk.Label(self.root, textvariable=self.result, font=("Arial", 14)).pack()

if __name__ == "__main__":
    root = tk.Tk()
    app = Arithmetic(root)
    root.mainloop()
```

Chapter 5 Ex-6 Output



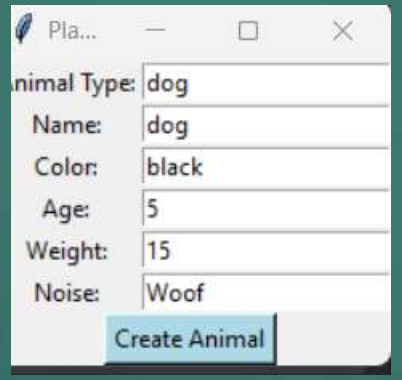
Chapter 5 Bonus

```python
# importing tkinter as tk
import tkinter as tk

# defining the function Animal as required
1 usage
class Animal:
    def __init__(self, Type, Name, Colour, Age, Weight, Noise):
        self.Type = Type
        self.Name = Name
        self.Colour = Colour
        self.Age = Age
        self.Weight = Weight
        self.Noise = Noise

    1 usage
    def sayHello(self):  # function for saying hello
        print(f"Hello, I am {self.Name}!")

    1 usage
    def makeNoise(self):  # function for making noise
        print(f"{self.Name} says: {self.Noise}")

    2 usages (1 dynamic)
    def animalDetails(self):  # function for animal details
        details = (f"Type: {self.Type}, Name: {self.Name}, Colour: {self.Colour}, Age: {self.Age}, Weight: {self.Weight}, "
                   f"Noise: {self.Noise}")
        print(details)
```

```python
# Creating GUI(graphical user interface)
1usage
def create_animal():
    animal_type = animal_type_entry.get()
    name = name_entry.get()
    color = color_entry.get()
    age = age_entry.get()
    weight = weight_entry.get()
    noise = noise_entry.get()

    animal = Animal(animal_type, name, color, age, weight, noise)
    animal.sayHello()
    animal.makeNoise()
    animal.animalDetails()


root = tk.Tk()
root.title("Playing around in class")

# Labels
tk.Label(root, text="Animal Type:").grid(row=0, column=0)
tk.Label(root, text="Name:").grid(row=1, column=0)
tk.Label(root, text="Color:").grid(row=2, column=0)
tk.Label(root, text="Age:").grid(row=3, column=0)
tk.Label(root, text="Weight:").grid(row=4, column=0)
tk.Label(root, text="Noise:").grid(row=5, column=0)
```

```python
# Entries
animal_type_entry = tk.Entry(root)
name_entry = tk.Entry(root)
color_entry = tk.Entry(root)
age_entry = tk.Entry(root)
weight_entry = tk.Entry(root)
noise_entry = tk.Entry(root)

# creating grid for the entries
animal_type_entry.grid(row=0, column=1)
name_entry.grid(row=1, column=1)
color_entry.grid(row=2, column=1)
age_entry.grid(row=3, column=1)
weight_entry.grid(row=4, column=1)
noise_entry.grid(row=5, column=1)

# Button for creating animal
tk.Button(root, text="Create Animal", command=create_animal,bg='lightblue').grid(row=6, columnspan=2)

root.mainloop()
```

Chapter 5 Bonus Output



Chapter 5 Further Ex

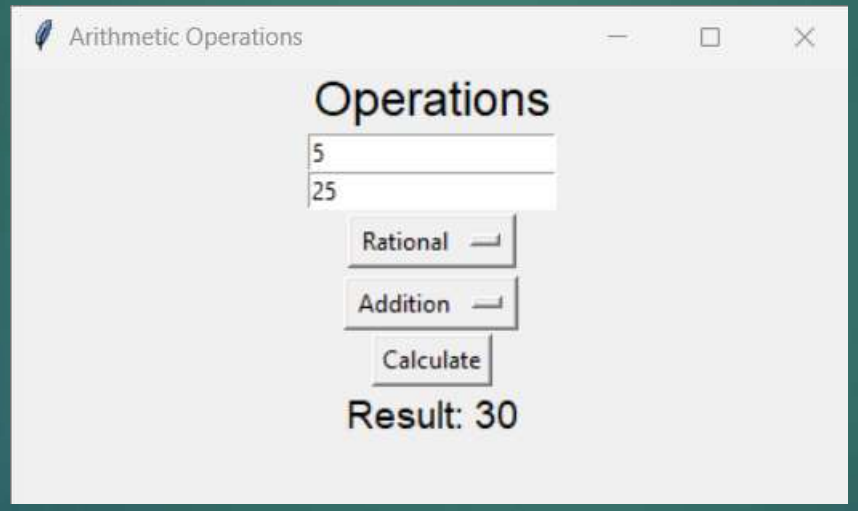
```python
# import tkinter
import tkinter as tk
from tkinter import messagebox
from fractions import Fraction  # importing fractions

class Operations:
    def __init__(self, root):
        self.root = root
        self.root.title("Arithmetic Operations")
        self.result = tk.StringVar()
        self.result.set("Result: ")

        self.widgets()

    def calculate(self):
        operation_type = self.operation_type.get()
        operation = self.operation.get()
        try:
            if operation_type == "Arithmetic":
                num1 = float(self.num1.get())
                num2 = float(self.num2.get())

                if operation == "Addition":
                    self.result.set("Result: " + str(num1 + num2))
                elif operation == "Subtraction":
                    self.result.set("Result: " + str(num1 - num2))
                elif operation == "Multiplication":
                    self.result.set("Result: " + str(num1 * num2))
                elif operation == "Division":
                    if num2 != 0:
```

```python
                if num2 != 0:
                    self.result.set("Result: " + str(num1 / num2))
                else:
                    self.result.set("Result: Undefined")
                    messagebox.showerror( title: "Error",  message: "Cannot divide by zero!")
            elif operation_type == "Rational":
                frac1 = Fraction(self.num1.get())
                frac2 = Fraction(self.num2.get())

                if operation == "Addition":
                    self.result.set("Result: " + str(frac1 + frac2))
                elif operation == "Subtraction":
                    self.result.set("Result: " + str(frac1 - frac2))
                elif operation == "Multiplication":
                    self.result.set("Result: " + str(frac1 * frac2))
                elif operation == "Division":
                    if frac2 != 0:
                        self.result.set("Result: " + str(frac1 / frac2))
                    else:
                        self.result.set("Result: Undefined")
                        messagebox.showerror( title: "Error",  message: "Cannot divide by zero!")

        except ValueError:
            messagebox.showerror( title: "Error",  message: "Please enter valid numbers.")
        except ZeroDivisionError:
            messagebox.showerror( title: "Error",  message: "Cannot divide by zero in rational operations.")

# creating widgets for the user input
    1usage
    def widgets(self):
        tk.Label(self.root, text="Operations", font=("Arial", 18)).pack()
```

```python
self.num1 = tk.Entry(self.root)
self.num1.pack()

self.num2 = tk.Entry(self.root)
self.num2.pack()

operation_types = ["Arithmetic", "Rational"]  # types of operations
self.operation_type = tk.StringVar()
self.operation_type.set(operation_types[0])

operation_type_menu = tk.OptionMenu(self.root, self.operation_type, *operation_types)
operation_type_menu.pack()

arithmetic_operations = ["Addition", "Subtraction", "Multiplication", "Division"]_# options for arithmetic
rational_operations = ["Addition", "Subtraction", "Multiplication", "Division"]  # options for rational

self.operation = tk.StringVar()
self.operation.set(arithmetic_operations[0])

operation_menu = tk.OptionMenu(self.root, self.operation, *arithmetic_operations)
operation_menu.pack()

# button for calculating
tk.Button(self.root, text="Calculate", command=self.calculate).pack()

# label for the result
tk.Label(self.root, textvariable=self.result, font=("Arial", 14)).pack()
```

```python
        operation_menu = tk.OptionMenu(self.root, self.operation, *values: *arithmetic_operations)
        operation_menu.pack()

        # button for calculating
        tk.Button(self.root, text="Calculate", command=self.calculate).pack()

        # label for the result
        tk.Label(self.root, textvariable=self.result, font=("Arial", 14)).pack()

# printing the output
if __name__ == "__main__":
    root = tk.Tk()  # main window
    app = Operations(root)
    root.mainloop()  # running the main loop
```

Chapter 5 Further Ex Output



Chapter-6

CHAPTER 6 - PYTHON STANDARD LIBRARY MODULES

Chapter 6 Ex-1

```python
# importing math library
import math

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

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

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

# Find the value of 2^3
power = math.pow( __x: 2,  __y: 3)
print(f"Value of 2^3: {power}")

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

Output

```
Ceil of 2.3: 3
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
```

Chapter 6 Ex-2

```python
# importing numpy as np
import numpy as np

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

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

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

# Slice to the end of the list
slice_1 = a[3:]
print(f"Slice from index 3 to the end: {slice_1}")

# Slice elements from index 0 to index 4
slice_2 = a[:5]
print(f"Slice from index 0 to index 4: {slice_2}")

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

Output

```
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
```

Chapter 6 Ex-3

```python
# importing operator
import operator

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

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

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

# defining function to perform division
1 usage
def divide(x, y):
    return operator.truediv(x, y)

# defining function to perform modulos calculation
1 usage
def modulus(x, y):
    return operator.mod(x, y)

# defining function to check greater number
1 usage
```

```python
# defining function to check greater number
1 usage
def check_greater(x, y):
    return operator.gt(x, y)

# The menu which will be displayed to the user to choose from
1 usage
def menu():
    print("Calculator Menu:")
    print("1. Add")
    print("2. Subtract")
    print("3. Multiply")
    print("4. Divide")
    print("5. Modulus")
    print("6. Check greater number")
    print("Q. Quit")

# nested if statements to check whether the input is valid or not
def calculate(choice):
    if choice == '1':
        x = float(input("Enter first number: "))
        y = float(input("Enter second number: "))
        print("Result:", add(x, y))
    elif choice == '2':
        x = float(input("Enter first number: "))
        y = float(input("Enter second number: "))
        print("Result:", subtract(x, y))
    elif choice == '3':
        x = float(input("Enter first number: "))
        y = float(input("Enter second number: "))
```

```python
        print("Result:", multiply(x, y))
    elif choice == '4':
        x = float(input("Enter first number: "))
        y = float(input("Enter second number: "))
        if y != 0:
            print("Result:", divide(x, y))
        else:
            print("Error! Division by zero.")
    elif choice == '5':
        x = float(input("Enter first number: "))
        y = float(input("Enter second number: "))
        print("Result:", modulus(x, y))
    elif choice == '6':
        x = float(input("Enter first number: "))
        y = float(input("Enter second number: "))
        print("Greater number:", check_greater(x, y))
    elif choice.upper() == 'Q':
        print("Exiting the calculator.")
    else:
        print("Invalid input.")

def main():
    while True:
        menu()
        user_choice = input("Enter your choice: ")
        if user_choice.upper() == 'Q':
            break
        else:
            calculate(user_choice)

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

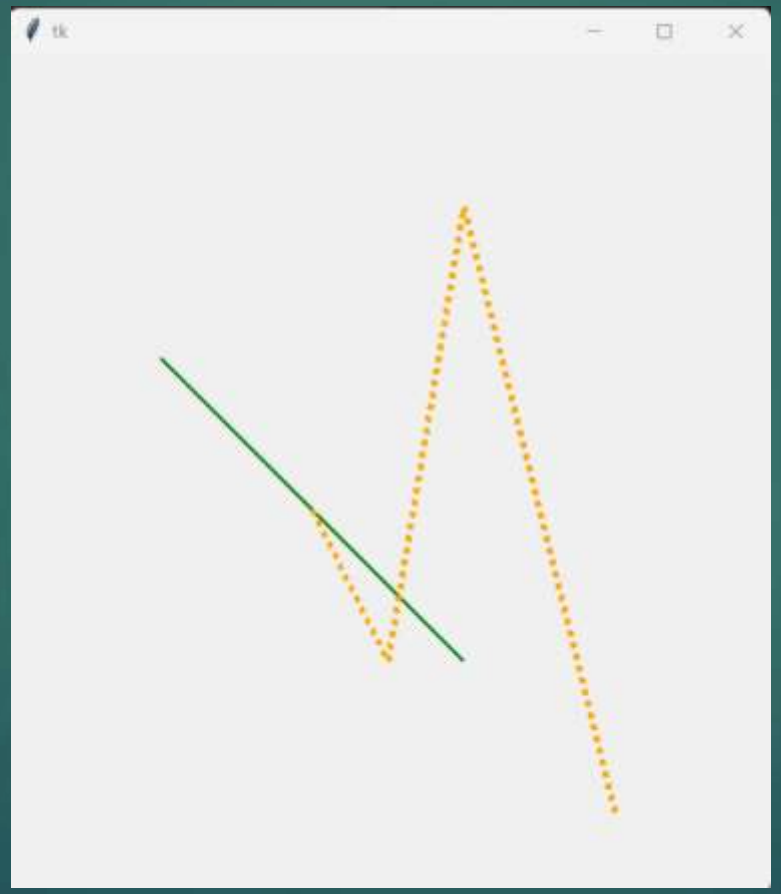
Chapter 6 Ex-3 Output

```
Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Modulus
6. Check greater number
Q. Quit
Enter your choice: 1
Enter first number: 3
Enter second number: 9
Result: 12.0
Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Modulus
6. Check greater number
Q. Quit
Enter your choice: Q

Process finished with exit code 0
```

Chapter 6 Ex 4

```python
# importing tkinter as tk
import tkinter as tk

# defining the function draw_line to draw a line from start to end
1usage
def draw_line(canvas, start, end, color, width):
    # Draw a line from start to end
    line = canvas.create_line(start[0], start[1], end[0], end[1], fill=color, width=width)

1usage
def dotted_line(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)

def main():
    root = tk.Tk()
    canvas = tk.Canvas(root, width=500, height=550)
    canvas.pack()

# solid line from (1, 2) to (6, 8)
    draw_line(canvas, start: (100, 200), end: (300, 400), color: "green", width: 2)

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

    root.mainloop()

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

Chapter 6 Ex 4 Output



Chapter 6 Ex 5

```python
# importing json
import json

# Getting user input for the name, id, course
name = input("Enter student name: ")
student_id = input("Enter student ID: ")
course = input("Enter student course: ")

# Creating dictionary
student_info = {
    "Name": name,
    "ID": student_id,
    "course": course
}

# Writing to JSON file
with open('StudentJson.json', 'w') as json_file:  # file created by name of StudentJson.json
    json.dump(student_info, json_file)


# Read from the JSON file
with open('StudentJson.json', 'r') as json_file:
    student_data = json.load(json_file)

# Append additional details as given in the question
student_data["CourseDetails"] = {
    "Group": "A",
    "Year": 2
}
```

```python
# Update the JSON file
with open('StudentJson.json', 'w') as json_file:
    json.dump(student_data, json_file)


# Read the updated JSON file
with open('StudentJson.json', 'r') as json_file:
    student_data = json.load(json_file)

# Display the output by displaying individual values
print("Details of the Student are")
print("\tName:", student_data["Name"])
print("\tID:", student_data["ID"])
print("\tcourse:", student_data["course"])
print("\tGroup:", student_data["CourseDetails"]["Group"])
print("\tYear:", student_data["CourseDetails"]["Year"])
```

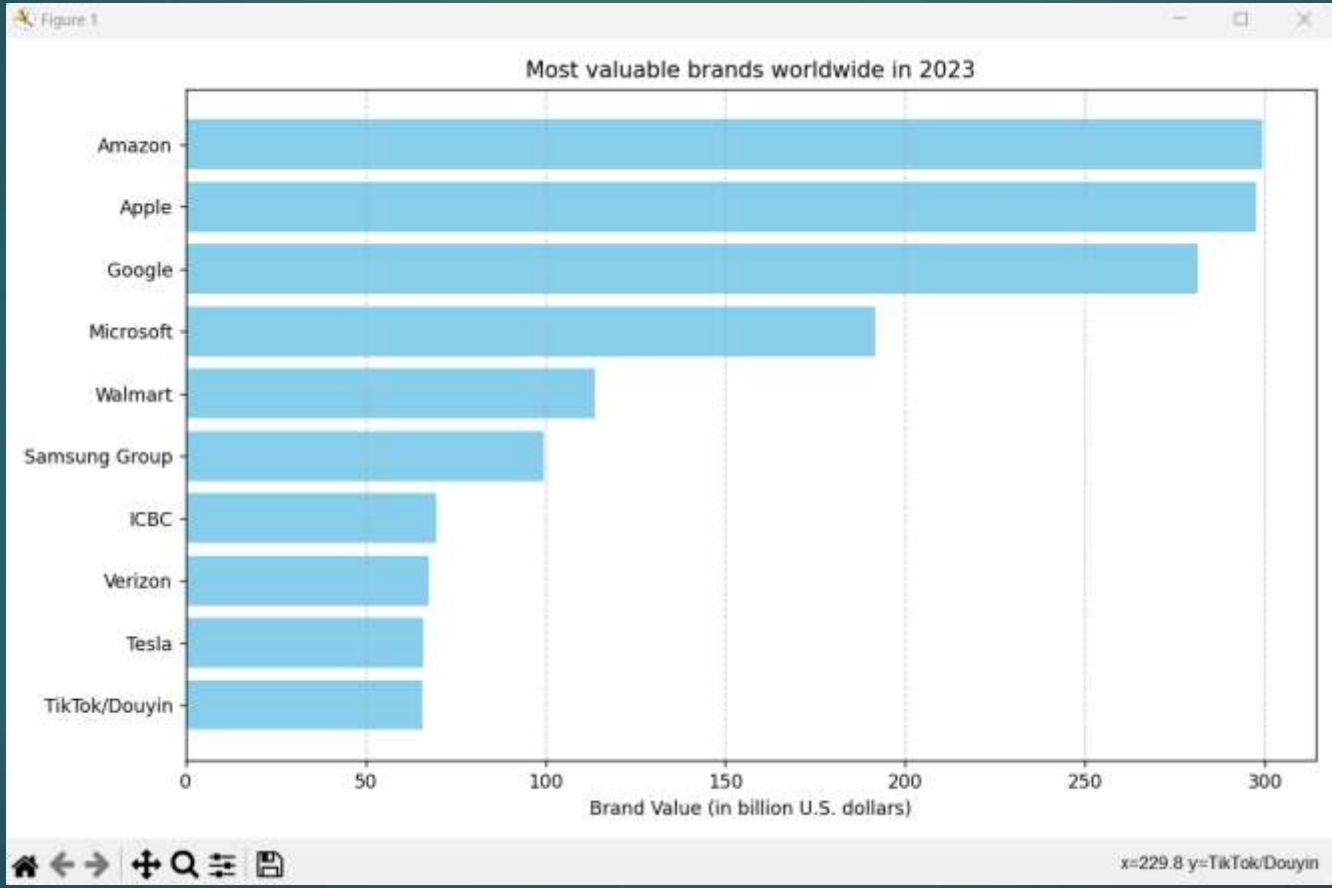
Chapter 6 Ex 5 Output

```
Enter student name: Rahil
Enter student ID: 01
Enter student course: CC
Details of the Student are
    Name: Rahil
    ID: 01
    course: CC
    Group: A
    Year: 2

Process finished with exit code 0
```

Chapter 6 Bonus Ex

```python
# importing the matplot library
import matplotlib.pyplot as plt

# Data
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]

# Plotting
plt.figure(figsize=(10, 6))
plt.barh(brands, values, color='skyblue')
plt.xlabel('Brand Value (in billion U.S. dollars)')
plt.title('Most valuable brands worldwide in 2023')
plt.gca().invert_yaxis()  # To display the highest value at the top
plt.grid(axis='x', linestyle='--', alpha=0.7)

# Show plot
plt.tight_layout()
plt.show()
```

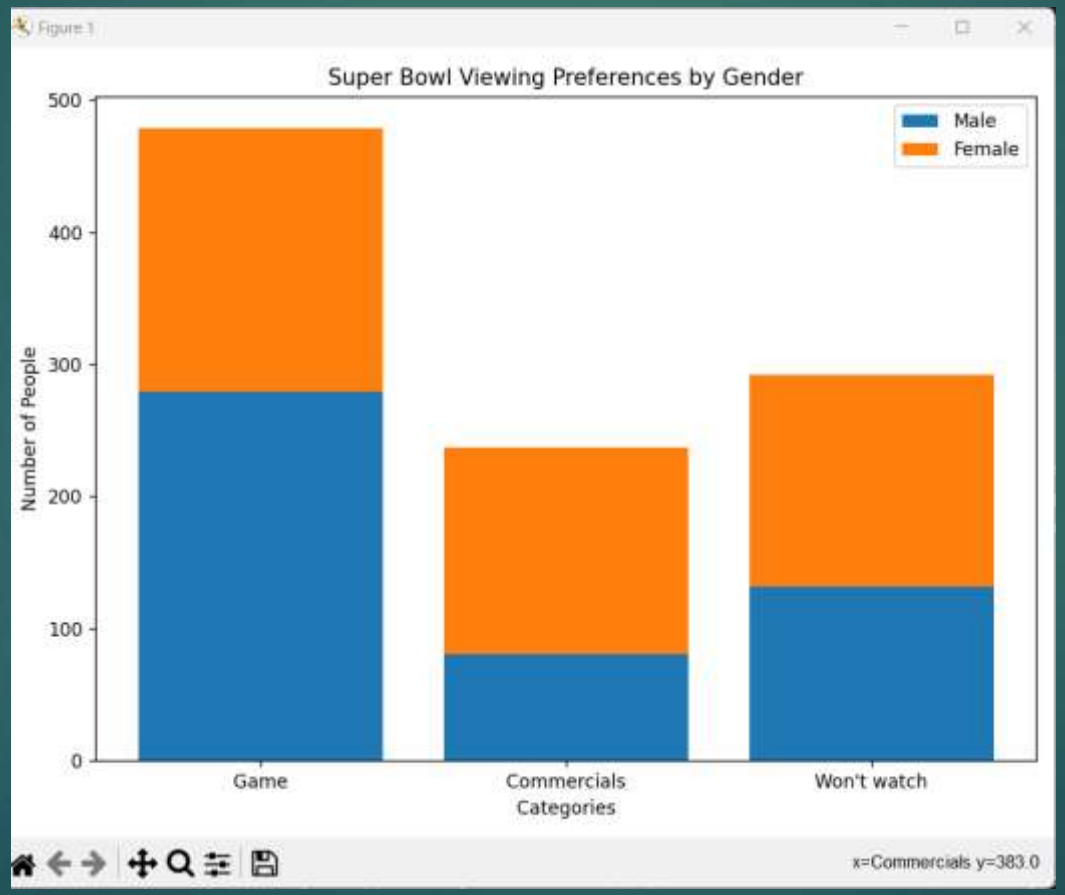
Chapter 6 Bonus Output



Chapter 6 Further Ex-1

```python
# importing the library
import matplotlib.pyplot as plt

# Data
categories = ['Game', 'Commercials', "Won't watch"]
male = [279, 81, 132]
female = [200, 156, 160]

# Plotting
fig, ax = plt.subplots(figsize=(8, 6))
bar1 = ax.bar(categories, male, label='Male')
bar2 = ax.bar(categories, female, bottom=male, label='Female')

# Adding labels, title, and legend
ax.set_xlabel('Categories')   # label for categories
ax.set_ylabel('Number of People')   # number of people
ax.set_title('Super Bowl Viewing Preferences by Gender')   # title naming
ax.legend()

# Show plot
plt.tight_layout()
plt.show()
```

Chapter 6 Further Ex-1 Output



Chapter 6 Further Ex 2

```python
# importing json
import json

# Getting user input for the name, id, course
name = input("Enter student name: ")
student_id = input("Enter student ID: ")
course = input("Enter student course: ")

# Creating dictionary
student_info = {
    "Name": name,
    "ID": student_id,
    "course": course
}

# Writing to JSON file
with open('StudentJson.json', 'w') as json_file:  # file created by name of StudentJson.json
    json.dump(student_info, json_file)


# Read from the JSON file
with open('StudentJson.json', 'r') as json_file:
    student_data = json.load(json_file)

# Append additional details as given in the question
student_data["CourseDetails"] = {
    "Group": "A",
    "Year": 2
}
```

```python
# Update the JSON file
with open('StudentJson.json', 'w') as json_file:
    json.dump(student_data, json_file)


# Read the updated JSON file
with open('StudentJson.json', 'r') as json_file:
    student_data = json.load(json_file)

# Display the output by displaying individual values
print("Details of the Student are")
print("\tName:", student_data["Name"])
print("\tID:", student_data["ID"])
print("\tcourse:", student_data["course"])
print("\tGroup:", student_data["CourseDetails"]["Group"])
print("\tYear:", student_data["CourseDetails"]["Year"])
```

Output

```
Enter student name: Rahil
Enter student ID: 01
Enter student course: CC
Details of the Student are
    Name: Rahil
    ID: 01
    course: CC
    Group: A
    Year: 2

Process finished with exit code 0
```