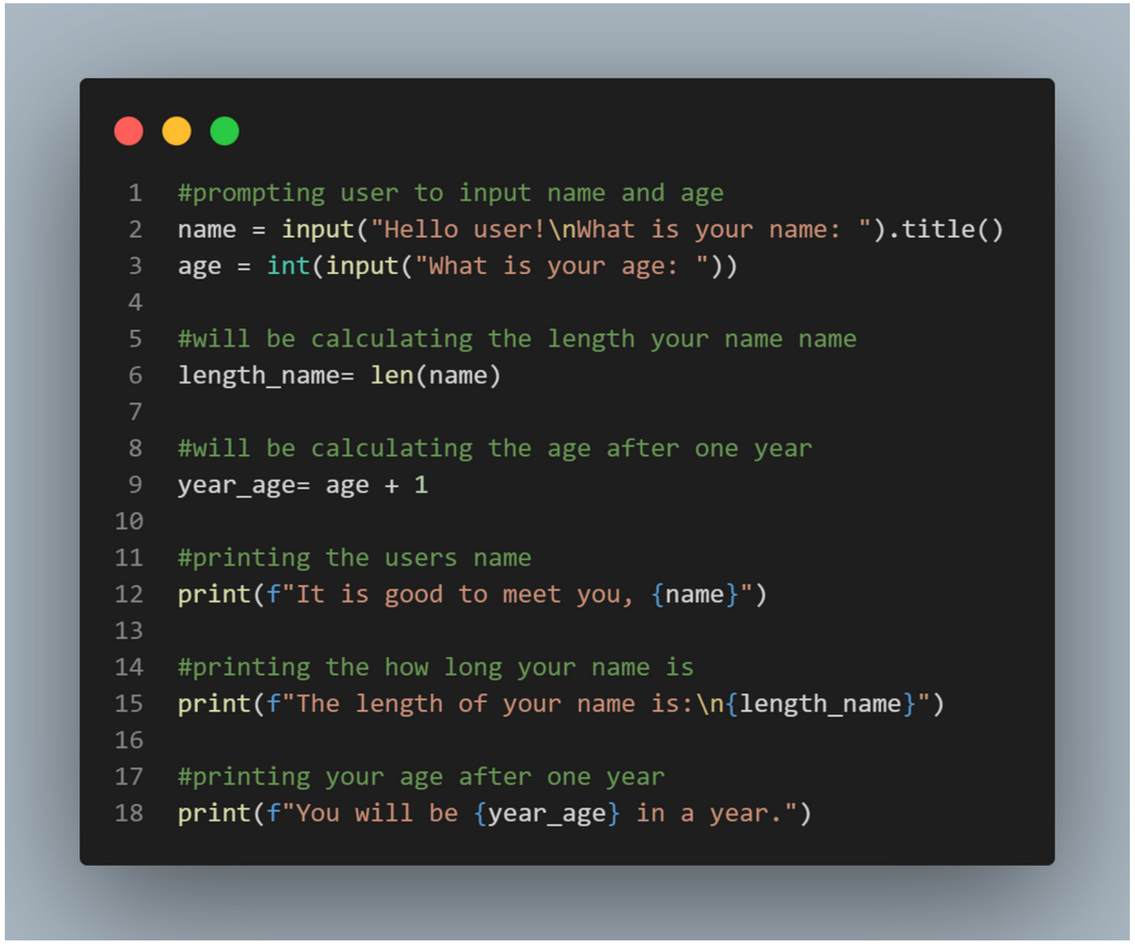
Chapter 1

Exercise 1



```python
#prompting user to input name and age
name = input("Hello user!\nWhat is your name: ").title()
age = int(input("What is your age: "))

#will be calculating the length your name name
length_name= len(name)

#will be calculating the age after one year
year_age= age + 1

#printing the users name
print(f"It is good to meet you, {name}")

#printing the how long your name is
print(f"The length of your name is:\n{length_name}")

#printing your age after one year
print(f"You will be {year_age} in a year.")
```

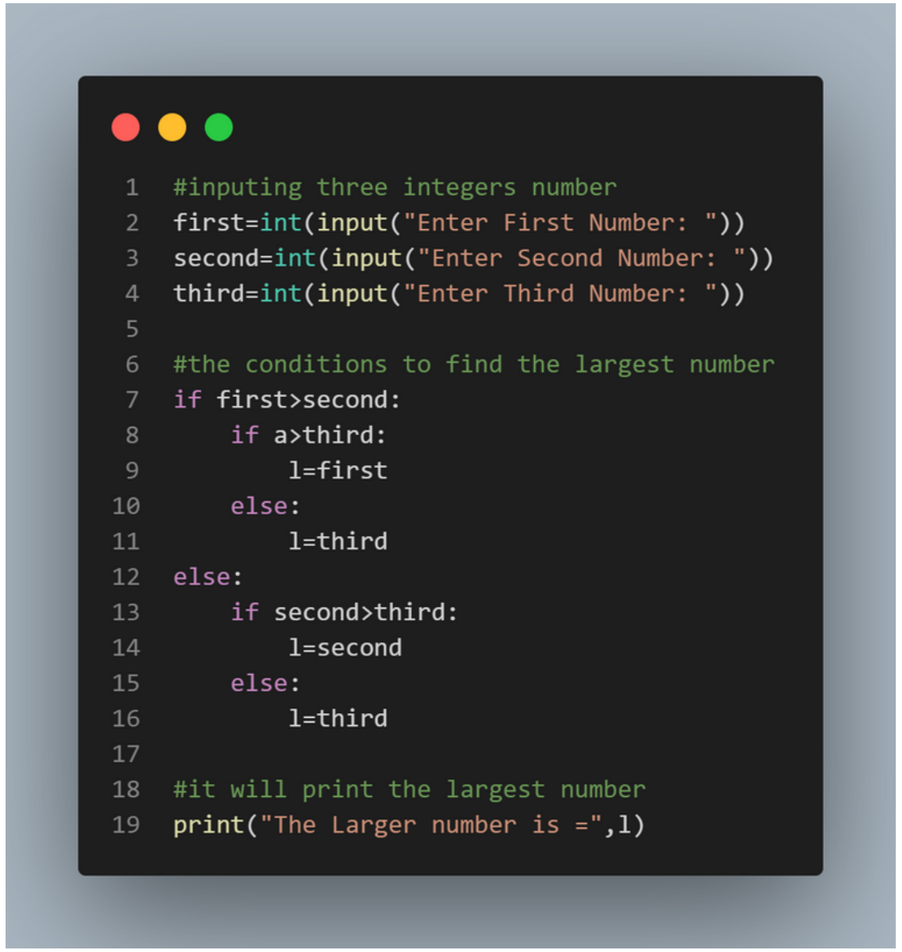
Exercise 2

```python
#it will get two integer numbers from the user
num1 = int(input("Enter the first integer number: "))
num2 = int(input("Enter the second integer number: "))

#it will perform calculations
result_sum = num1 + num2
result_difference = num1 - num2
result_product = num1 * num2

#checking if the second number is not zero for division
if num2 != 0:
    result_division = num1 / num2
else:
    result_division = "Cannot divide by zero"

#calculating remainder only if the second number is not zero
if num2 != 0:
    result_remainder = num1 % num2
else:
    result_remainder = "Cannot calculate remainder, second number is zero"

#output of the results
print(f"Sum: {result_sum}")
print(f"Difference: {result_difference}")
print(f"Product: {result_product}")
print(f"Division: {result_division}")
print(f"Remainder: {result_remainder}")
```

Exercise 3

```python
#entering the first side of the triangle
side1 = int(input("Pleaser enter first side of the triangle : "))

#entering the second side of the triangle
side2= int(input("Please enter second side of the triangle : "))

#entering the third side of the triangle
side3 = int(input("Please enter third side of the triangle : "))

#the conditions whether the triangle can be formed or not
if side1 + side2 > side3 and side2 + side3 > side1 and side1 + side3 > side2 :

    #if its possible it will say that it is a triangle
    print("It is a triangle")

    #if its not possible it will say that it is not a triangle
else :
    print("It is not a triangle")
```

Exercise 4



```python
#inputing three integers number
first=int(input("Enter First Number: "))
second=int(input("Enter Second Number: "))
third=int(input("Enter Third Number: "))

#the conditions to find the largest number
if first>second:
    if a>third:
        l=first
    else:
        l=third
else:
    if second>third:
        l=second
    else:
        l=third

#it will print the largest number
print("The Larger number is =",l)
```

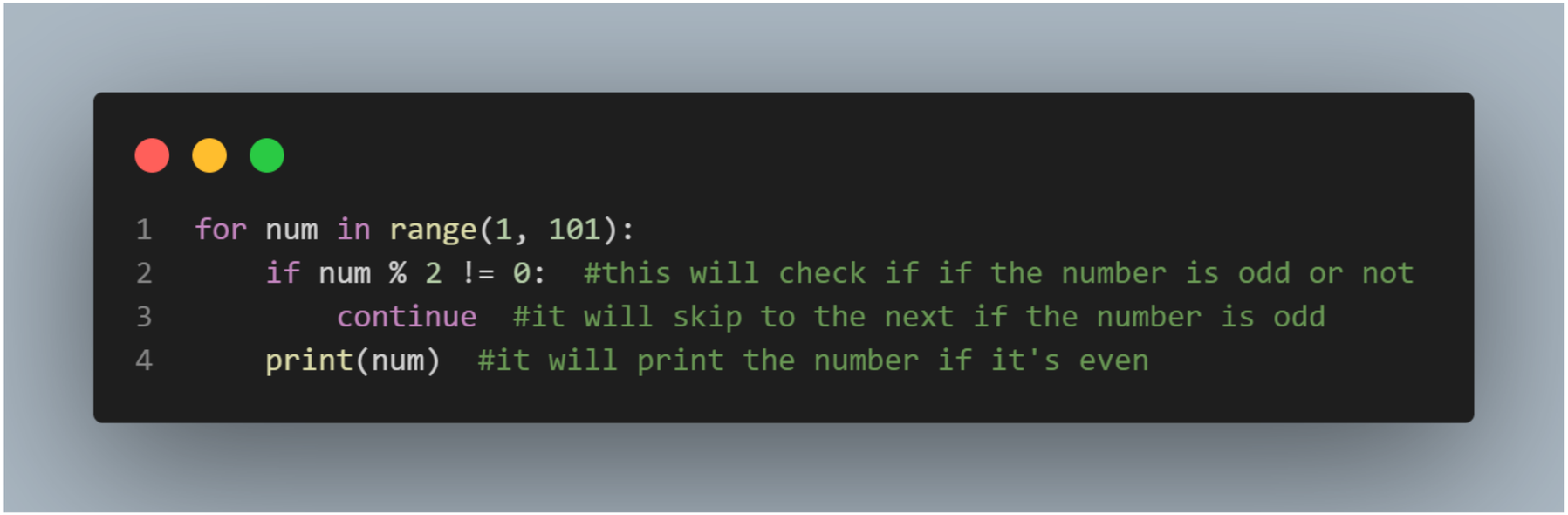
Exercise 5

```python
#a count variable to keep track of loop iterations
count = 0

#this is an infinite loop
while True:

    #getting the user input
    user_input = input("Would you like to continue? (Y/N): ").upper()

    #if the user enters Y
    if user_input == 'Y':

        #computing the count if the user enters Y
        count += 1

    #when the users enter N
    elif user_input == 'N':
        break
    else:
        #the user will enter Y or N
        print("It's invalid. Please enter 'Y' or 'N'.")
        continue  #it will continue the loop to get a valid input

#if the loop breaks and if the user entered 'N', it will print the count of loop executions
print(f"The loop was executed {count} times.")
```

Exercise 6

```
1
2  for num in range(1, 101):
3      #checking if the numbers can be divided by 3 or 5
4      if num % 3 == 0 and num % 5 == 0:
5          print("FizzBuzz")#it will print fizzbuzz if its divisible by 3 or 5
6      #it will check if the numbers can be divided by 3
7      elif num % 3 == 0:
8          print("Fizz")#it will print fizz if divisible by 3
9      #checking if the numbers can be divided by 5
10     elif num % 5 == 0:
11         print("Buzz") #it will print buzz if the number is divded by 5
12     else:
13         print(num)#printing the numbers itself
```

Exercise 7



```python
for num in range(1, 101):
    if num % 2 != 0:  #this will check if if the number is odd or not
        continue  #it will skip to the next if the number is odd
    print(num)  #it will print the number if it's even
```

Exercise 8

```
for l in range(1, 6):#the total numbers
    for j in range(1, l + 1):#this will print 1 to i
        print(j, end=" ")#printing its number
    print()#it wil move to the next line after the number is printed
```

Exercise 9

```python
#creating an int list with 10 values
int_list = [17, 29, 43, 56, 81, 92, 14, 67, 12, 23]

#the output using for loop
print("List:")
for num in int_list:
    print(num, end=" ")  #printing the elements
print()

#output for the highest and lowest value
high = max(int_list)#finding the highest value
low = min(int_list)#finding the lowest value
print(f"Highest value: {high}")
print(f"Lowest value: {low}")

#elements in ascending order
ascending_order = sorted(int_list)#the list in ascending orders
print("Sort in ascending order:", ascending_order)

#elements in descending order
descending_order = sorted(int_list, reverse=True)#the list in descending order
print("Sort in descending order:", descending_order)

#appending two elements
int_list.append(111)#appending value 111 to the list
int_list.append(5)#appending the value 5 to the list
print("The list after appending two elements:", int_list)
```

Exercise 10

```python
#creating a dictionary for film details
film = {
    "Genre": "Thriller/ Action/ History/ Documentary/ Mystery/ Drama",
    "Title": "Oppenheimer",
    "Director": "Christopher Nolan",
    "Year": 2023,


}

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

Exercise 11

```python
#defining the tuple
year = (2017, 2003, 2011, 2005, 1987, 2009, 2020, 2018, 2009)

#value at index -3
value_at_index_minus_three = year[-3]
print("The value at index -3:", value_at_index_minus_three)

#reversing the tuple and printing the original and reversed tuple
reversed_year = tuple(reversed(year))
print("Original tuple:", year)
print("Reversed tuple:", reversed_year)

#counting the number of times 2009 is at the tuple
count_2009 = year.count(2009)
print("Number of times 2009 appeared:", count_2009)

#getting the index value of 2018
index_2018 = year.index(2018)
print("Index of 2018:", index_2018)

#finding the length of the tuple
tuple_length = len(year)
print("Length of the tuple:", tuple_length)
```

Exercise 12

```python
import math  #importing math

#a function to calculate the are of the square
def calculate_square_area():
    side = float(input("Enter the side length of the square: "))#asking the user for side length
    area = side ** 2#calculating the are of the square
    print(f"The area of the square is: {area}") #printing the calculated area

#a function to calculate area of the circle
def calculate_circle_area():
    r = float(input("Enter the radius of circle: "))#asking user for radius
    a = math.pi * (r ** 2)  #area of the circle using pi * r^2 formula
    print(f"The area of the circle is: {a}")#printing the calculated area

#the fucntion to calculate the area of a triangle
def calculate_triangle_area():
    b = float(input("Enter the base length of triangle: "))# asking the user for base length
    h = float(input("Enter the height of the triangle: "))  #asking the user for height
    ar = 0.5 * b * h  # calculating area of the triangle using (base * height) / 2 formula
    print(f"The area of the triangle is: {ar}") #printing the calculated area

#displaying the menu and what are the player wants to calculate
while True:
    print("What do you want to Calculate\nMenu:")
    print("1: Calculate the area of a square")
    print("2: Calculate the area of a circle")
    print("3: Calculate the area of a triangle")

    choice = input("Enter your choice (1/2/3): ") #asking the user for their choice

    if choice == '1':
        calculate_square_area() #function to calculate square
    elif choice == '2':
        calculate_circle_area()  #function to calculate circle
    elif choice == '3':
        calculate_triangle_area()  #function to calculate triangle
    else:
        print("Invalid choice. Please enter a valid option.")  #error message for invalid input

```

Chapter 2

Exercise 1

```python
import tkinter as tk

root = tk.Tk()
root.geometry("500x400") #default window size
root.resizable(False, False) #disabling the resizing window
root.configure(bg="green") #background color

#function for changing the label font style
def change_font():
    label_welcome.config(font=("Times", 30, "bold"))

#displaying the welcome message
label_welcome = tk.Label(root, text="Welcome!!!", font=("Helvetica", 17,), bg="beige")
label_welcome.pack(pady=50)#adjusting the label's vertical position

#the button to change the font style
button_font = tk.Button(root, text="Change the Font", command=change_font)
button_font.pack()#the button for window

root.mainloop()
```

Exercise 2

```python
from tkinter import *

root = Tk()  #the main window

#creating labels with specified text, width, background color, relief, and border width
ab = Label(root, text="A", width=12, bg="red", relief=GROOVE, bd=5)
bb = Label(root, text="B", width=12, bg="yellow")
bc = Label(root, text="C", width=12, bg="blue")
ba = Label(root, text="D", width=12, bg="white")

#the labels into the main window with specified configurations
ab.pack(side="top", fill=X, expand=1)# placing label A at the top, horizontally stretch to fill the X-axis
bb.pack(side="bottom") #placing a label B at the bottom
bc.pack(side="left", expand=1) #placing label C at the left, horizontally stretch to expand
ba.pack(side="right") #placing label D at the right

root.mainloop()  #starting the GUI

```

Exercise 2(b)

```python
import tkinter as tk

#creating the main window
root = tk.Tk()
root.geometry("300x200")#window size

#creating left frame
frame_left = tk.Frame(root, borderwidth=5) #a frame with border width
frame_left.pack(side=tk.LEFT, expand=True, fill=tk.BOTH)

#right frame
frame_right = tk.Frame(root, borderwidth=5)  #creating a frame with border width
frame_right.pack(side=tk.RIGHT, expand=True, fill=tk.BOTH) #right frame to the right side

# labels A, B, C, and D with border and background color
A_label= tk.Label(frame_left, text="A", border=30, bg="#22263d", fg="white")
B_label = tk.Label(frame_left, text="B", border=30)
C_label = tk.Label(frame_right, text="C", border=30)
D_label = tk.Label(frame_right, text="D", border=30, bg="#22263d", fg="white")

#labels A, B, C, and D inside their respective frames
A_label.pack(side=tk.TOP, expand=True, fill=tk.X)
B_label.pack(side=tk.BOTTOM, expand=True, fill=tk.X)
C_label.pack(side=tk.TOP, expand=True, fill=tk.X)
D_label.pack(side=tk.BOTTOM, expand=True, fill=tk.X)

root.mainloop()
```

Exercise 3

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

#codes to perform login authentication
def on_login():

    #getting username and password from entry fields
    username = entry_username.get()
    password = entry_password.get()

    #checking if username and password match
    if username == "user" and password == "pass":
        results = "Login successful!"
    else:
        results = "Invalid username or password."

    tk.messagebox.showinfo("Result", results, parent=root)


root = tk.Tk()
root.title("Login Page")
root.geometry("200x200")
root.columnconfigure(0, weight=1)
root.rowconfigure(0, weight=1)
#username for label and entry
label_username = ttk.Label(root, text="Username:")
label_username.grid(row=0, column=0, padx=6, pady=6, sticky="w")

entry_username = ttk.Entry(root, width=20)
entry_username.grid(row=0, column=1, padx=6, pady=6)
#label for password and entry
label_password = ttk.Label(root, text="Password:")
label_password.grid(row=1, column=0, padx=6, pady=6, sticky="w")

entry_password = ttk.Entry(root, width=20, show="*")
entry_password.grid(row=1, column=1, padx=6, pady=6)

#login button
button_login = ttk.Button(root, text="Login", command=on_login)
button_login.grid(row=5, column=0, columnspan=2, padx=6, pady=6)#button place in grid

root.mainloop()
```

Exercise 4

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

#the main window
root = tk.Tk()
root.title("Student Management System")
root.geometry("550x900")

#creating custom style for frame
style = ttk.Style()
style.configure("TFrame", background="#f4f4f6")

imagelogo = PhotoImage(file="logo.png")
label_images = ttk.Label(root, image=imagelogo)
label_images.pack(pady=(20, 0))

#the header labels
header_labels1 = ttk.Label(root, text="Student Management System", font=("Helvetica", 20, "bold"))
header_labels1.pack(pady=(20, 11))

header_labels2 = ttk.Label(root, text="New Student Registration", font=("Helvetica", 16, "bold"))
header_labels2.pack(pady=(0, 20))
#creating a frame
frame = ttk.Frame(root, padding="50", style="TFrame")
frame.pack()

#entry widgets for the data inputs
entry_name = ttk.Entry(frame)
entry_name.grid(row=0, column=1, padx=11, pady=12)

entry_mobile = ttk.Entry(frame)
entry_mobile.grid(row=1, column=1, padx=12, pady=12)

entry_email = ttk.Entry(frame)
entry_email.grid(row=2, column=1, padx=11, pady=12)

entry_home_address = ttk.Entry(frame)
entry_home_address.grid(row=3, column=1, padx=11, pady=12)

gender_var = tk.StringVar()
combox_gender = ttk.Combobox(frame, textvariable=gender_var)
combox_gender.grid(row=4, column=1, padx=11, pady=12)
combox_gender['values'] = ('Male', 'Female', 'Other')

course_var = tk.StringVar()
combox_course = ttk.Combobox(frame, textvariable=course_var)
combox_course.grid(row=5, column=1, padx=11, pady=12)
combox_course['values'] = ('BSc CC', 'BSc CY', 'BSc PSY', 'BA & BM')

language_var = tk.StringVar()
combox_language = ttk.Combobox(frame, textvariable=language_var)
combox_language.grid(row=6, column=1, padx=11, pady=12)
combox_language['values'] = ('English', 'Tagalog', 'Hindi / Urdu')

#label widgets for data inputs
label_name = ttk.Label(frame, text="Student Name",)
label_name.grid(row=0, column=0, padx=11, pady=12)

label_mobile = ttk.Label(frame, text="Mobile Number")
label_mobile.grid(row=1, column=0, padx=11, pady=12, )

label_email = ttk.Label(frame, text="Email Id")
label_email.grid(row=2, column=0, padx=11, pady=12)

label_home_address = ttk.Label(frame, text="Home Address")
label_home_address.grid(row=3, column=0, padx=11, pady=12)

label_gender = ttk.Label(frame, text="Gender")
label_gender.grid(row=4, column=0, padx=11, pady=12)

label_course = ttk.Label(frame, text="Course Enrolled")
label_course.grid(row=5, column=0, padx=11, pady=12)

label_language = ttk.Label(frame, text="Language Known")
label_language.grid(row=6, column=0, padx=11, pady=12)

#labels for the english communication skills
english_rating_label = ttk.Label(frame, text="Rate your English communication skills")
english_rating_label.grid(row=7, column=0, padx=11, pady=12)

#slider for the english communications skills
english_rating = ttk.Scale(frame, from_=0, to=11, orient=tk.HORIZONTAL, length=145)
english_rating.grid(row=7, column=1, padx=11, pady=12)


#button
style = ttk.Style()
style.configure("TButton", background="#22263d")

add_button = ttk.Button(frame, text="Submit", style="TButton")
add_button.grid(row=8, column=0, padx=22, pady=55)
clear_button = ttk.Button(frame, text="Clear", style="TButton")
clear_button.grid(row=8, column=1, padx=22, pady=55)

root.mainloop()
```

Exercise 5

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

#function for perform calculations
def calculate():
    try:
        #getting user to input values
        num1 = float(entry1.get())
        num2 = float(entry2.get())

        #getting the selected operation
        operation = operation_var.get()

        #addition
        if operation == "Addition":
            result = num1 + num2
            #subtraction
        elif operation == "Subtraction":
            result = num1 - num2
            #multilpication
        elif operation == "Multiplication":
            result = num1 * num2
            #division
        elif operation == "Division":
            if num2 == 0:
                messagebox.showerror("Error", "Division by zero is not allowed")
                return
            else:
                result = num1 / num2
        elif operation == "Modulo Division":
            if num2 == 0:
                messagebox.showerror("Error", "Modulo division by zero is not allowed")
                return
            else:
                result = num1 % num2
        else:
            result = "Invalid Operation"

        #updating the result label
        result_label.config(text=f"Result: {result}")
    except ValueError:
        messagebox.showerror("Error", "Please enter valid numeric values.")

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

#entry the widgets for numbers
entry1 = tk.Entry(root)
entry1.pack()

entry2 = tk.Entry(root)
entry2.pack()

#dropdown for the operations
operations = ["Addition", "Subtraction", "Multiplication", "Division", "Modulo Division"]
operation_var = tk.StringVar(root)
operation_var.set(operations[0]) #set the default operation

operation_dropdown = tk.OptionMenu(root, operation_var, *operations)
operation_dropdown.pack()

#the button to calculate
calculate_button = tk.Button(root, text="Calculate", command=calculate)
calculate_button.pack()

#label for display result
result_label = tk.Label(root, text="Result: ", font=("Arial", 10))
result_label.pack()

#starting the main loop
root.mainloop()

```

Chapter 3

Exercise 1

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

def update_greeting():
    name = entry_name.get()
    color = color_var.get()
    greeting = f"Hello, {name}!"
    label_display.config(text=greeting, foreground=color)

#main window
root = tk.Tk()
root.title("Greeting App")
root.geometry("700x450")

# inputframe with blue background
frame_input = ttk.Frame(root, padding=23, style="Input.TFrame")
frame_input.grid(row=0, column=0, padx=23, pady=23)

#displayframe with green background
frame_display = ttk.Frame(root, padding=23, style="Display.TFrame", width=250)
frame_display.grid(row=0, column=2, padx=23, pady=23)

#a custom style for frames
style = ttk.Style()

#style for inputframe
style.configure("Input.TFrame", background="#87CEFB")

#style for displayframe
style.configure("Display.TFrame", background="#98FB99")

#creating widgets for inputframe
label_title = ttk.Label(frame_input, text="Welcome!!!", font=("Helvetica", 25, "bold"), foreground="blue")
label_title.grid(row=0, column=0, columnspan=3, pady=23)

label_name = ttk.Label(frame_input, text="Name:")
label_name.grid(row=2, column=0, sticky="w")

entry_name = ttk.Entry(frame_input)
entry_name.grid(row=2, column=2, pady=6)

label_color = ttk.Label(frame_input, text="Select Color:")
label_color.grid(row=3, column=0, sticky="w")

color_var = tk.StringVar(value="black")
dropdown_color = ttk.Combobox(frame_input, textvariable=color_var, values=["black", "red", "blue", "green"])
dropdown_color.grid(row=3, column=2, pady=6)

button_update = ttk.Button(frame_input, text="Update Greeting", command=update_greeting)
button_update.grid(row=4, column=0, columnspan=3, pady=23)

#label for displayframe
label_display = ttk.Label(frame_display, font=("Helvetica", 25), anchor="center")
label_display.pack(expand=True, fill="both")

root.mainloop()
```

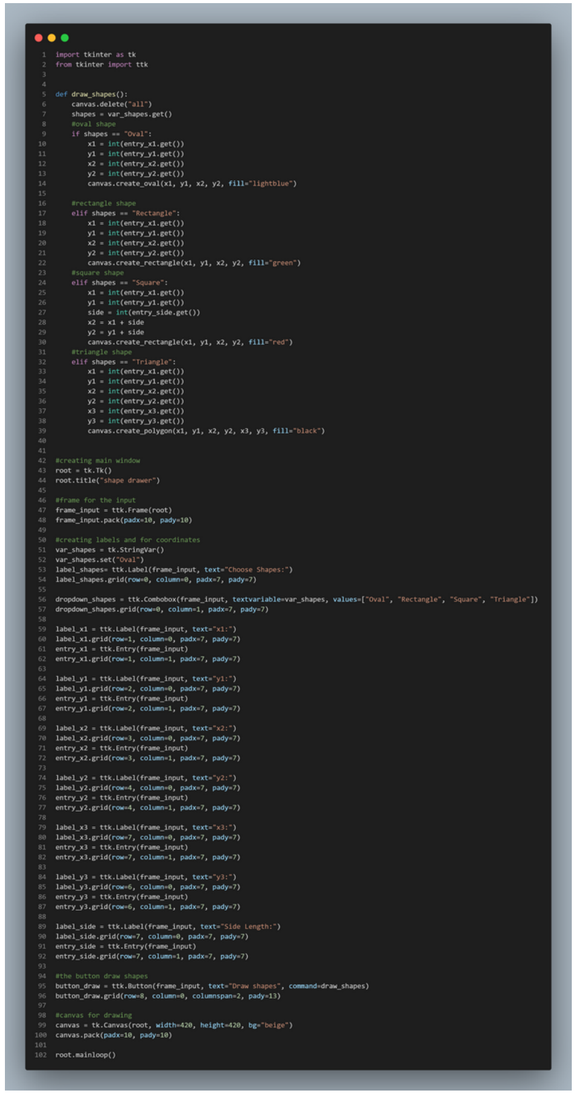
Exercise 2

```python
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from PIL import Image, ImageTk
#functions for coffee orders
def order_coffee():
    selected_coffee = coffee_var.get()
    options = []

    if milk_var.get():
        options.append("Milk")

    if sugar_var.get():
        options.append("Sugar")

    if chocolate_var.get():
        options.append("Chocolate")

    message = f"You ordered a {selected_coffee} coffee with {', '.join(options)}."
    messagebox.showinfo("Order Confirmation", message)

root = tk.Tk()
root.title("Coffee Vending Machine")
root.geometry("500x500")

#options for coffee
coffee_var = tk.StringVar(value="Espresso")
coffee_label = ttk.Label(root, text="Select Coffee Type:")
coffee_label.pack()

coffee_options = ttk.Combobox(root, textvariable=coffee_var, values=["Espresso", "Spanish Latte", "Cappuccino"])
coffee_options.pack()

#additional of options
sugar_var = tk.BooleanVar(value=False)
milk_var = tk.BooleanVar(value=False)
chocolate_var = tk.BooleanVar(value=False)

check_boxsugars = ttk.Checkbutton(root, text="Sugar", variable=sugar_var)
check_boxsugars.pack()

checkbox_milks = ttk.Checkbutton(root, text="Milk", variable=milk_var)
checkbox_milks.pack()

checkbox_chocolates = ttk.Checkbutton(root, text="Chocolate", variable=chocolate_var)
checkbox_chocolates.pack()

#loading and resizing images
image_espresso = Image.open("espresso.jpg")
image_espresso = image_espresso.resize((200, 100))
photo_espresso = ImageTk.PhotoImage(image_espresso)
label_espresso = ttk.Label(root, image=photo_espresso)
label_espresso.pack()

image_cappucinno = Image.open("capuccino.jpg")
image_cappucinno = image_cappucinno.resize((200, 100))
photo_cappucinno = ImageTk.PhotoImage(image_cappucinno)
label_cappucinno = ttk.Label(root, image=photo_cappucinno)
label_cappucinno.pack()

image_spanishlatte = Image.open("spanish latte.jpg")
image_spanishlatte = image_spanishlatte.resize((200, 100))
spanishlatte = ImageTk.PhotoImage(image_spanishlatte)
label_spanishlatte = ttk.Label(root, image=spanishlatte)
label_spanishlatte.pack()



#order button
order_button = ttk.Button(root, text="Order Coffee", command=order_coffee)
order_button.pack()

root.mainloop()
```

Exercise 3

```python
import tkinter as tk
from tkinter import ttk
import math
#to calculate and display the area of the circle
def calculate_circle_area2():
    radius1 = float(entry_circle_radius1.get())
    area2 = math.pi * (radius1 ** 2)
    result_circle_label.config(text=f"area of the Circle: {area2:.2f}")

#to calculate and display the area of the square
def calculate_square_area2():
    sides = float(entry_squares_sides.get())
    area2 = sides ** 2
    result_square_label.config(text=f"area of the Square: {area2:.2f}")
#to calculate and display the area of the rectangle
def calculate_rectangle_area2():
    length = float(entry_rectangle_length.get())
    width = float(entry_rectangle_width.get())
    area2 = length * width
    label_rectangle_result.config(text=f"area of the Rectangle: {area2:.2f}")

#main window
root = tk.Tk()
root.title("Area Calculator")
root.geometry("450x350")

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

#circle tab
circles_tab = ttk.Frame(notebook)
notebook.add(circles_tab, text="Circle")

label_circle_radius1 = ttk.Label(circles_tab, text="Enter radius1:")
label_circle_radius1.grid(row=0, column=0, padx=7, pady=7)

entry_circle_radius1 = ttk.Entry(circles_tab)
entry_circle_radius1.grid(row=0, column=1, padx=7, pady=7)

button_circle_calculate = ttk.Button(circles_tab, text="Calculate", command=calculate_circle_area2)
button_circle_calculate.grid(row=1, column=0, columnspan=2, pady=10)

result_circle_label = ttk.Label(circles_tab, text="area of the Circle: ")
result_circle_label.grid(row=2, column=0, columnspan=2)

#square tab
square_tab = ttk.Frame(notebook)
notebook.add(square_tab, text="Square")

label_square_sides = ttk.Label(square_tab, text="Enter side length:")
label_square_sides.grid(row=0, column=0, padx=7, pady=7)

entry_squares_sides = ttk.Entry(square_tab)
entry_squares_sides.grid(row=0, column=1, padx=7, pady=7)

button_square_calculate = ttk.Button(square_tab, text="Calculate", command=calculate_square_area2)
button_square_calculate.grid(row=1, column=0, columnspan=2, pady=10)

result_square_label = ttk.Label(square_tab, text="area of the Square: ")
result_square_label.grid(row=2, column=0, columnspan=2)

#rectangle tab
tab_rectangles = ttk.Frame(notebook)
notebook.add(tab_rectangles, text="Rectangle")

label_rectangle_length = ttk.Label(tab_rectangles, text="Enter length:")
label_rectangle_length.grid(row=0, column=0, padx=7, pady=7)

entry_rectangle_length = ttk.Entry(tab_rectangles)
entry_rectangle_length.grid(row=0, column=1, padx=7, pady=7)

label_rectangle_width = ttk.Label(tab_rectangles, text="Enter width:")
label_rectangle_width.grid(row=1, column=0, padx=7, pady=7)

entry_rectangle_width = ttk.Entry(tab_rectangles)
entry_rectangle_width.grid(row=1, column=1, padx=7, pady=7)

button_rectangle_calculate = ttk.Button(tab_rectangles, text="Calculate", command=calculate_rectangle_area2)
button_rectangle_calculate.grid(row=2, column=0, columnspan=2, pady=10)

label_rectangle_result = ttk.Label(tab_rectangles, text="area of the Rectangle: ")
label_rectangle_result.grid(row=3, column=0, columnspan=2)


notebook.pack(padx=10, pady=10, fill='both', expand=True)

root.mainloop()
```

Exercise 4



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


def draw_shapes():
    canvas.delete("all")
    shapes = var_shapes.get()
    #oval shape
    if shapes == "Oval":
        x1 = int(entry_x1.get())
        y1 = int(entry_y1.get())
        x2 = int(entry_x2.get())
        y2 = int(entry_y2.get())
        canvas.create_oval(x1, y1, x2, y2, fill="lightblue")

    #rectangle shape
    elif shapes == "Rectangle":
        x1 = int(entry_x1.get())
        y1 = int(entry_y1.get())
        x2 = int(entry_x2.get())
        y2 = int(entry_y2.get())
        canvas.create_rectangle(x1, y1, x2, y2, fill="green")
    #square shape
    elif shapes == "Square":
        x1 = int(entry_x1.get())
        y1 = int(entry_y1.get())
        side = int(entry_side.get())
        x2 = x1 + side
        y2 = y1 + side
        canvas.create_rectangle(x1, y1, x2, y2, fill="red")
    #triangle shape
    elif shapes == "Triangle":
        x1 = int(entry_x1.get())
        y1 = int(entry_y1.get())
        x2 = int(entry_x2.get())
        y2 = int(entry_y2.get())
        x3 = int(entry_x3.get())
        y3 = int(entry_y3.get())
        canvas.create_polygon(x1, y1, x2, y2, x3, y3, fill="black")


#creating main window
root = tk.Tk()
root.title("shape drawer")

#frame for the input
frame_input = ttk.Frame(root)
frame_input.pack(padx=10, pady=10)

#creating labels and for coordinates
var_shapes = tk.StringVar()
var_shapes.set("Oval")
label_shapes= ttk.Label(frame_input, text="Choose Shapes:")
label_shapes.grid(row=0, column=0, padx=7, pady=7)

dropdown_shapes = ttk.Combobox(frame_input, textvariable=var_shapes, values=["Oval", "Rectangle", "Square", "Triangle"])
dropdown_shapes.grid(row=0, column=1, padx=7, pady=7)

label_x1 = ttk.Label(frame_input, text="x1:")
label_x1.grid(row=1, column=0, padx=7, pady=7)
entry_x1 = ttk.Entry(frame_input)
entry_x1.grid(row=1, column=1, padx=7, pady=7)

label_y1 = ttk.Label(frame_input, text="y1:")
label_y1.grid(row=2, column=0, padx=7, pady=7)
entry_y1 = ttk.Entry(frame_input)
entry_y1.grid(row=2, column=1, padx=7, pady=7)

label_x2 = ttk.Label(frame_input, text="x2:")
label_x2.grid(row=3, column=0, padx=7, pady=7)
entry_x2 = ttk.Entry(frame_input)
entry_x2.grid(row=3, column=1, padx=7, pady=7)

label_y2 = ttk.Label(frame_input, text="y2:")
label_y2.grid(row=4, column=0, padx=7, pady=7)
entry_y2 = ttk.Entry(frame_input)
entry_y2.grid(row=4, column=1, padx=7, pady=7)

label_x3 = ttk.Label(frame_input, text="x3:")
label_x3.grid(row=7, column=0, padx=7, pady=7)
entry_x3 = ttk.Entry(frame_input)
entry_x3.grid(row=7, column=1, padx=7, pady=7)

label_y3 = ttk.Label(frame_input, text="y3:")
label_y3.grid(row=6, column=0, padx=7, pady=7)
entry_y3 = ttk.Entry(frame_input)
entry_y3.grid(row=6, column=1, padx=7, pady=7)

label_side = ttk.Label(frame_input, text="Side Length:")
label_side.grid(row=7, column=0, padx=7, pady=7)
entry_side = ttk.Entry(frame_input)
entry_side.grid(row=7, column=1, padx=7, pady=7)

#the button draw shapes
button_draw = ttk.Button(frame_input, text="Draw shapes", command=draw_shapes)
button_draw.grid(row=8, column=0, columnspan=2, pady=11)

#canvas for drawing
canvas = tk.Canvas(root, width=420, height=420, bg="beige")
canvas.pack(padx=10, pady=10)

root.mainloop()
```

Chapter 4

Exercise 1

```python
import tkinter as tk

def save_to_file():
    name = entry_name.get()
    age = entry_age.get()
    hometown = entry_hometown.get()

    with open("bio.txt", "w") as file:
        file.write(f"Name: {name}\nAge: {age}\nHometown: {hometown}")

def read_from_file():
    with open("bio.txt", "r") as file:
        content = file.read()
        label_output.config(text=content)

# Create main window
root = tk.Tk()
root.title("Bio Information")
root.geometry("500x400")

# Create labels and entry fields
label_name = tk.Label(root, text="Name:")
label_name.pack()

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

label_age = tk.Label(root, text="Age:")
label_age.pack()

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

label_hometown = tk.Label(root, text="Hometown:")
label_hometown.pack()

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

# Create buttons
button_saves = tk.Button(root, text="Save to your File", command=save_to_file)
button_saves.pack()

button_reads = tk.Button(root, text="Read from your File", command=read_from_file)
button_reads.pack()

# Create label to display output
label_output = tk.Label(root, text="", wraplength=350)
label_output.pack()

# Start the tkinter event loop
root.mainloop()
```

Exercise 2

```python
import tkinter as tk
#using sentences.txt to count how many sentences
def count_occurrences():
    search_string = entry.get().strip()
    if search_string:
        with open("sentences.txt", "r") as file:
            content = file.read()
            occurrences = content.lower().count(search_string.lower())
            label_results.config(text=f"The string appears {occurrences} times.")
    else:
        label_results.config(text="Please enter a search string.")


root = tk.Tk()
root.title("String Occurrence Counter")
root.geometry("400x300")

#labels and the entry field
label_entry = tk.Label(root, text="Enter the string:")
label_entry.pack()

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

#creating a button
button_counts = tk.Button(root, text="Count the Occurrences", command=count_occurrences)
button_counts.pack()

#label to display the result
label_results = tk.Label(root, text="", wraplength=320)
label_results.pack()


root.mainloop()
```

Exercise 3

```python
#reading nums from the file and creating a list of a num
with open('numbers.txt', 'r') as file:
    numbers = [int(line.strip()) for line in file]

#outputing the values
print("List of integers:")
for num in numbers:
    print(num)
```

Exercise 4

```python
import tkinter as tk
#counting how many characters are there in sentences.txt
def count_occurrences():
    char = entry.get()
    with open('sentences.txt', 'r') as file:
        content = file.read()
    counts = content.count(char)
    label_result.config(text=f"The characters '{char}' occurs {counts} times.")

#main window
root = tk.Tk()
root.title("Character Occurrence Counter")
root.geometry("400x300")

#creating the label and entry for the characters
label_char = tk.Label(root, text="Enter the characters:")
label_char.pack(pady=12)
entry = tk.Entry(root)
entry.pack(pady=12)

# button to trigger thee counting
count_buttons = tk.Button(root, text="Count Occurrences", command=count_occurrences)
count_buttons.pack(pady=12)

#creating a label display the results
label_result = tk.Label(root, text="")
label_result.pack()

#starting the tkinter event loop
root.mainloop()
```

Exercise 5

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

def validate_password():
    attempts_remaining = 5
    #loop to for maximum attempts
    while attempts_remaining > 0:
        password = entry_password.get()

        #password criteria
        if len(password) < 6 or len(password) > 12:
            messagebox.showwarning("Invalid Password", "Password length should be between 6 and 12 characters.")
        elif not any(charac.islower() for charac in password):
            messagebox.showwarning("Invalid Password", "Password should contain at least 1 lowercase letter (a-z).")
        elif not any(charac.isupper() for charac in password):
            messagebox.showwarning("Invalid Password", "Password should contain at least 1 uppercase letter (A-Z).")
        elif not any(charac.isdigit() for charac in password):
            messagebox.showwarning("Invalid Password", "Password should contain at least 1 digit (0-9).")
        elif not any(charac in "$#@ " for charac in password):
            messagebox.showwarning("Invalid Password", "Password should contain at least one of '$', '#', '@'.")
        else:
            messagebox.showinfo("Valid Password", "Password is valid.")
            break  #brakingg the loop if the password is valid

        attempts_remaining -= 1
        #handing the attempts count and alerting
        if attempts_remaining > 0:
            messagebox.showwarning("Attempts Remaining", f"{attempts_remaining} attempts remaining.")
        else:
            messagebox.showerror("Alert!", "Authorities have been alerted!")
            break  #breaking the loop if attempts reach the limit

#main window
root = tk.Tk()
root.title("Password Validator")
root.geometry("300x300")

#widget for the password input
label_password = tk.Label(root, text="Enter Password:")
label_password.pack(pady=11)

entry_password = tk.Entry(root, show="*")
entry_password.pack(pady=11)

#button for checking the password validity
validate_button = tk.Button(root, text="Check Password", command=validate_password)
validate_button.pack(pady=21)

root.mainloop()

```

Chapter 5

Exercise 1

```python
import tkinter as tk

class Dog:
    def __init__(self, name, age, breed):
        self.name = name
        self.age = age
        self.breed = breed
#details of the dogs
    def description(self):
        return f"Name: {self.name}, Age: {self.age}, Breed: {self.breed}"
    #figuring out who is the older dog
    @classmethod
    def old_dog_woof(cls, dog1st, dog2nd):
        old_dog = dog1st if dog1st.age > dog2nd.age else dog2nd
        print(f"{old_dog.name} says: Woof!")

#creating two dog objects and assigning data to their attributes
dog1st = Dog("Browny", 5, "Chihuahua")
dog2nd = Dog("Wilson", 7, "German Shepherd")

#the data for each dog
print("Dog 1:", dog1st.description())
print("Dog 2:", dog2nd.description())

#method to make the oldest dog "woof"
Dog.old_dog_woof(dog1st, dog2nd)

#simple Tkinter GUI to display dog information
root = tk.Tk()
root.title("Dog Information")

dog1st_label = tk.Label(root, text=dog1st.description())
dog1st_label.pack()

dog2nd_label = tk.Label(root, text=dog2nd.description())
dog2nd_label.pack()

root.mainloop()

```

Exercise 2

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

class Students:
    def __init__(self, name, mark1, mark2, mark3):
        self.name = name
        self.mark1 = mark1
        self.mark2 = mark2
        self.mark3 = mark3

    def calcGrade(self):
        #calculating the average of the three marks
        return (self.mark1 + self.mark2 + self.mark3) / 3

    def display(self):
        #displaying the student name and calculated grade average
        average = self.calcGrade()
        return f"Student Name: {self.name}\nAverage Grade: {average:.2f}"

#creating a students object and display the details
def process_student():
    #getting the input values from the user
    name = name_entry.get()
    mark1 = int(mark1_entry.get())
    mark2 = int(mark2_entry.get())
    mark3 = int(mark3_entry.get())

    #creating a students object with the input values
    student_obj = Students(name, mark1, mark2, mark3)

    #student details
    result = student_obj.display()
    messagebox.showinfo("Student Details", result)

#creating the main window
root = tk.Tk()
root.title("Student Details")

#labels and entry fields for student details
name_label = tk.Label(root, text="Enter Name:")
name_label.pack()

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

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

mark1_entry = tk.Entry(root)
mark1_entry.pack()

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

mark2_entry = tk.Entry(root)
mark2_entry.pack()

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

mark3_entry = tk.Entry(root)
mark3_entry.pack()

#the button to proceed to the student details
button_process = tk.Button(root, text="Process", command=process_student)
button_process.pack()

root.mainloop()

```

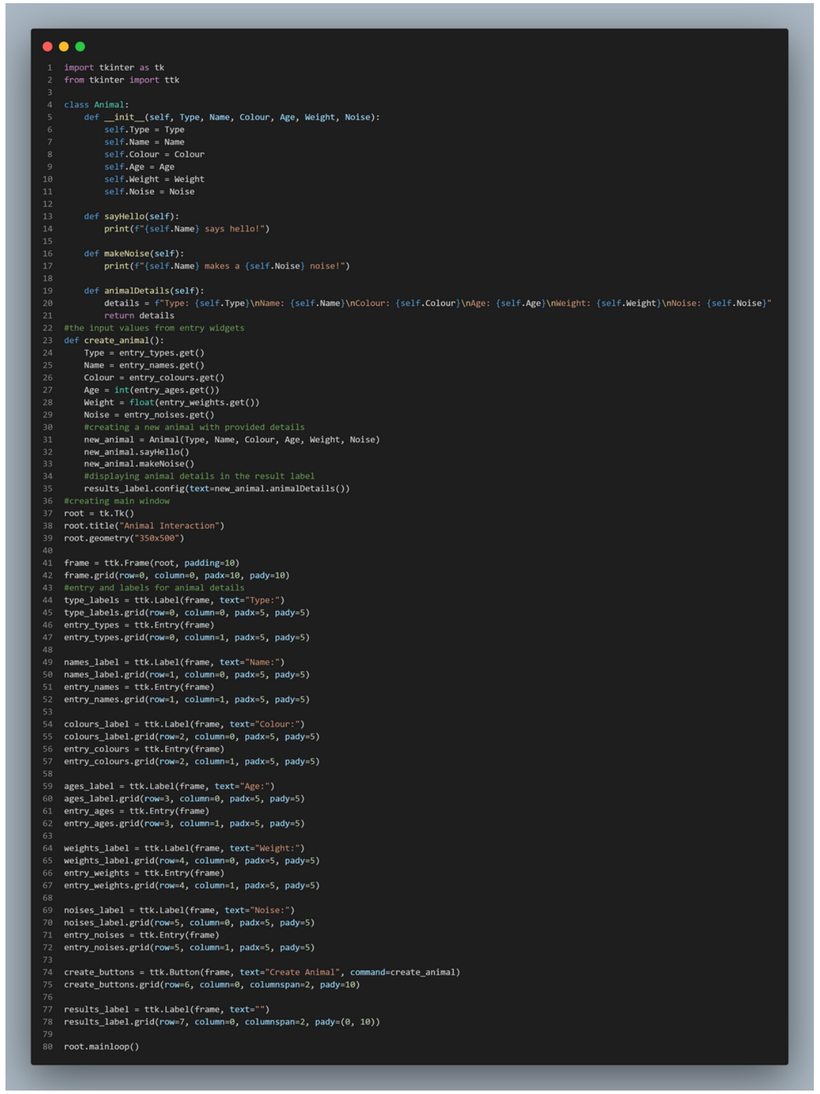
Exercise 3

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

class Employee:
    def __init__(self):
        self.name = ""
        self.age = 0
        self.id = 0
        self.salary = 0

    def setData(self, name, age, employee_id, salary):
        self.name = name
        self.age = age
        self.id = employee_id
        self.salary = salary

    def getData(self):
        return self.name, self.age, self.id, self.salary

#function for adding employee details
def add_employee():
    #input values from the entry fields
    name = name_entry.get()
    age = int(age_entry.get())
    employee_id = int(id_entry.get())
    salary = float(salary_entry.get())

    #creating an employee object and set the data
    emp = Employee()
    emp.setData(name, age, employee_id, salary)

    #append the employee data to the employees list
    employees.append(emp)

    #the entry fields for the next input
    name_entry.delete(0, tk.END)
    age_entry.delete(0, tk.END)
    id_entry.delete(0, tk.END)
    salary_entry.delete(0, tk.END)

#display employee details
def display_employee():
    #creating a text box to display the output
    output = tk.Toplevel()
    output.title("Employee Details")

    #create headers for the table
    tk.Label(output, text="Name").grid(row=0, column=0)
    tk.Label(output, text="Position").grid(row=0, column=1)
    tk.Label(output, text="Salary").grid(row=0, column=2)
    tk.Label(output, text="ID").grid(row=0, column=3)

    #displaying employee data in a table format
    for i, emp in enumerate(employees, start=1):
        name, age, employee_id, salary = emp.getData()
        position = "Position" + str(i)
        tk.Label(output, text=name).grid(row=i, column=0)
        tk.Label(output, text=position).grid(row=i, column=1)
        tk.Label(output, text=salary).grid(row=i, column=2)
        tk.Label(output, text=employee_id).grid(row=i, column=3)


root = tk.Tk()
root.title("Employee Details")

#creating empty list to store employee objects
employees = []

#labels and entry fields for user input
name_label = tk.Label(root, text="Name:")
name_label.pack()
name_entry = tk.Entry(root)
name_entry.pack()

age_label = tk.Label(root, text="Age:")
age_label.pack()
age_entry = tk.Entry(root)
age_entry.pack()

id_label = tk.Label(root, text="ID:")
id_label.pack()
id_entry = tk.Entry(root)
id_entry.pack()

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

#buttons to add employee details
add_button = tk.Button(root, text="Add Employee", command=add_employee)
add_button.pack()

#buttons to display employee details
display_button = tk.Button(root, text="Display Employees", command=display_employee)
display_button.pack()

root.mainloop()

```

Exercise 4

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

class Shape:
    def __init__(self):
        self.sides = []

    def inputSides(self):
        pass
#getting the radius of circle
class Circle(Shape):
    def inputSides(self):
        radius = float(radius_entry.get())
        self.sides.append(radius)

    def area(self):
        return pi * (self.sides[0] ** 2)
#getting lenghts and widths of rectangle
class Rectangle(Shape):
    def inputSides(self):
        lenghts = float(lenghts_entry.get())
        widths = float(widths_entry.get())
        self.sides.extend([lenghts, widths])

    def area(self):
        return self.sides[0] * self.sides[1]
#getting the base and height of the triangle
class Triangle(Shape):
    def inputSides(self):
        base = float(base_entry.get())
        height = float(height_entry.get())
        self.sides.extend([base, height])

    def area(self):
        return 0.6 * self.sides[0] * self.sides[1]
#function for displaying the area
def calculate_area():
    shape = shapes_combobox.get()
    if shape == "Circle":
        circles = Circle()
        circles.inputSides()
        label_result.config(text=f"Area: {circles.area()} square units")
    elif shape == "Rectangle":
        rectangles = Rectangle()
        rectangles.inputSides()
        label_result.config(text=f"Area: {rectangles.area()} square units")
    elif shape == "Triangle":
        triangles = Triangle()
        triangles.inputSides()
        label_result.config(text=f"Area: {triangles.area()} square units")

root = tk.Tk()
root.title("Shape Area Calculator")

frame = ttk.Frame(root, padding=10)
frame.grid(row=0, column=0, padx=10, pady=10)

shape_label = ttk.Label(frame, text="Select Shape:")
shape_label.grid(row=0, column=0, padx=6, pady=6)

shapes = ["Circle", "Rectangle", "Triangle"]
shapes_combobox = ttk.Combobox(frame, values=shapes, state="readonly")
shapes_combobox.grid(row=0, column=1, padx=6, pady=6)
shapes_combobox.current(0)

buttons_calcuates = ttk.Button(frame, text="Calculate Area", command=calculate_area)
buttons_calcuates.grid(row=1, column=0, columnspan=2, pady=10)

#circle columns and row and its spacing
label_radius = ttk.Label(frame, text="Radius:")
label_radius.grid(row=2, column=0, padx=6, pady=6)
radius_entry = ttk.Entry(frame)
radius_entry.grid(row=2, column=1, padx=6, pady=6)

#rectangle columns and row and its spacing
label_lenghts = ttk.Label(frame, text="Lenght:")
label_lenghts.grid(row=3, column=0, padx=6, pady=6)
lenghts_entry = ttk.Entry(frame)
lenghts_entry.grid(row=3, column=1, padx=6, pady=6)

label_widths = ttk.Label(frame, text="Width:")
label_widths.grid(row=4, column=0, padx=6, pady=6)
widths_entry = ttk.Entry(frame)
widths_entry.grid(row=4, column=1, padx=6, pady=6)

#triangle columns and row and its spacing
label_base = ttk.Label(frame, text="Base:")
label_base.grid(row=6, column=0, padx=6, pady=6)
base_entry = ttk.Entry(frame)
base_entry.grid(row=6, column=1, padx=6, pady=6)

label_height = ttk.Label(frame, text="Height:")
label_height.grid(row=6, column=0, padx=6, pady=6)
height_entry = ttk.Entry(frame)
height_entry.grid(row=6, column=1, padx=6, pady=6)

label_result = ttk.Label(frame, text="Area:")
label_result.grid(row=7, column=0, columnspan=2, pady=(10, 0))

root.mainloop()
```

Exercise 5



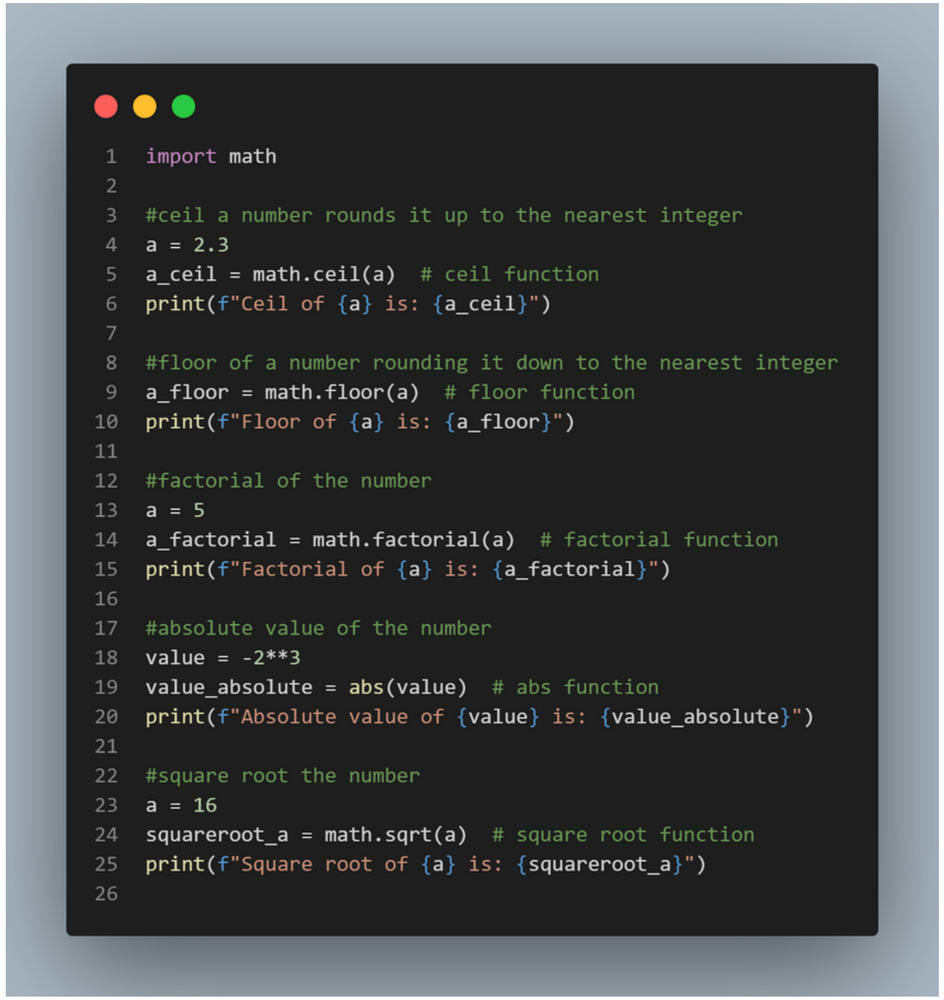
```python
import tkinter as tk
from tkinter import ttk

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

    def sayHello(self):
        print(f"{self.Name} says hello!")

    def makeNoise(self):
        print(f"{self.Name} makes a {self.Noise} noise!")

    def animalDetails(self):
        details = f"Type: {self.Type}\nName: {self.Name}\nColour: {self.Colour}\nAge: {self.Age}\nWeight: {self.Weight}\nNoise: {self.Noise}"
        return details
#the input values from entry widgets
def create_animal():
    Type = entry_types.get()
    Name = entry_names.get()
    Colour = entry_colours.get()
    Age = int(entry_ages.get())
    Weight = float(entry_weights.get())
    Noise = entry_noises.get()
    #creating a new animal with provided details
    new_animal = Animal(Type, Name, Colour, Age, Weight, Noise)
    new_animal.sayHello()
    new_animal.makeNoise()
    #displaying animal details in the result label
    results_label.config(text=new_animal.animalDetails())
#creating main window
root = tk.Tk()
root.title("Animal Interaction")
root.geometry("350x500")

frame = ttk.Frame(root, padding=10)
frame.grid(row=0, column=0, padx=10, pady=10)
#entry and labels for animal details
type_labels = ttk.Label(frame, text="Type:")
type_labels.grid(row=0, column=0, padx=5, pady=5)
entry_types = ttk.Entry(frame)
entry_types.grid(row=0, column=1, padx=5, pady=5)

names_label = ttk.Label(frame, text="Name:")
names_label.grid(row=1, column=0, padx=5, pady=5)
entry_names = ttk.Entry(frame)
entry_names.grid(row=1, column=1, padx=5, pady=5)

colours_label = ttk.Label(frame, text="Colour:")
colours_label.grid(row=2, column=0, padx=5, pady=5)
entry_colours = ttk.Entry(frame)
entry_colours.grid(row=2, column=1, padx=5, pady=5)

ages_label = ttk.Label(frame, text="Age:")
ages_label.grid(row=3, column=0, padx=5, pady=5)
entry_ages = ttk.Entry(frame)
entry_ages.grid(row=3, column=1, padx=5, pady=5)

weights_label = ttk.Label(frame, text="Weight:")
weights_label.grid(row=4, column=0, padx=5, pady=5)
entry_weights = ttk.Entry(frame)
entry_weights.grid(row=4, column=1, padx=5, pady=5)

noises_label = ttk.Label(frame, text="Noise:")
noises_label.grid(row=5, column=0, padx=5, pady=5)
entry_noises = ttk.Entry(frame)
entry_noises.grid(row=5, column=1, padx=5, pady=5)

create_buttons = ttk.Button(frame, text="Create Animal", command=create_animal)
create_buttons.grid(row=6, column=0, columnspan=2, pady=10)

results_label = ttk.Label(frame, text="")
results_label.grid(row=7, column=0, columnspan=2, pady=(0, 10))

root.mainloop()
```

Exercise 6

```python
import tkinter as tk
from tkinter import ttk
#class for the arithmetic operation
class ArithmeticOperations:
    def __init__(self):
        self.result = 0
    #method of performing arithmetic operation
    def calculate(self, operation, num1, num2):
        if operation == "Addition":
            self.result = num1 + num2
        elif operation == "Subtraction":
            self.result = num1 - num2
        elif operation == "Multiplication":
            self.result = num1 * num2
        elif operation == "Division":
            self.result = num1 / num2
#function for performing the calculation based on the user input
def perform_calculation():
    operation = operations_combobox.get()
    num1 = float(entry_n1.get())
    num2 = float(entry_n2.get())

    calculator = ArithmeticOperations()
    calculator.calculate(operation, num1, num2)
    results_label.config(text=f"Result: {calculator.result}")
#creating the main window
root = tk.Tk()
root.title("Arithmetic Operation")
#creating frames for organizing widgets
frames= ttk.Frame(root, padding=13)
frames.grid(row=0, column=0, padx=13, pady=13)
#labels and entry widgets for the numbers and for the operations
operations_label = ttk.Label(frames, text="Operation:")
operations_label.grid(row=0, column=0, padx=7, pady=7)
options_operations = ["Addition", "Subtraction", "Multiplication", "Division"]
operations_combobox = ttk.Combobox(frames, values=options_operations)
operations_combobox.grid(row=0, column=1, padx=7, pady=7)

n1_label = ttk.Label(frames, text="Number 1:")
n1_label.grid(row=1, column=0, padx=7, pady=7)
entry_n1 = ttk.Entry(frames)
entry_n1.grid(row=1, column=1, padx=7, pady=7)

n2_label = ttk.Label(frames, text="Number 2:")
n2_label.grid(row=2, column=0, padx=7, pady=7)
entry_n2 = ttk.Entry(frames)
entry_n2.grid(row=2, column=1, padx=7, pady=7)
#buttons for calculate
buttons_calculates = ttk.Button(frames, text="Calculate", command=perform_calculation)
buttons_calculates.grid(row=3, column=0, columnspan=2, pady=13)
#label for display results
results_label = ttk.Label(frames, text="Results:")
results_label.grid(row=4, column=0, columnspan=2)

root.mainloop()
```

Chapter 6

Exercise 1

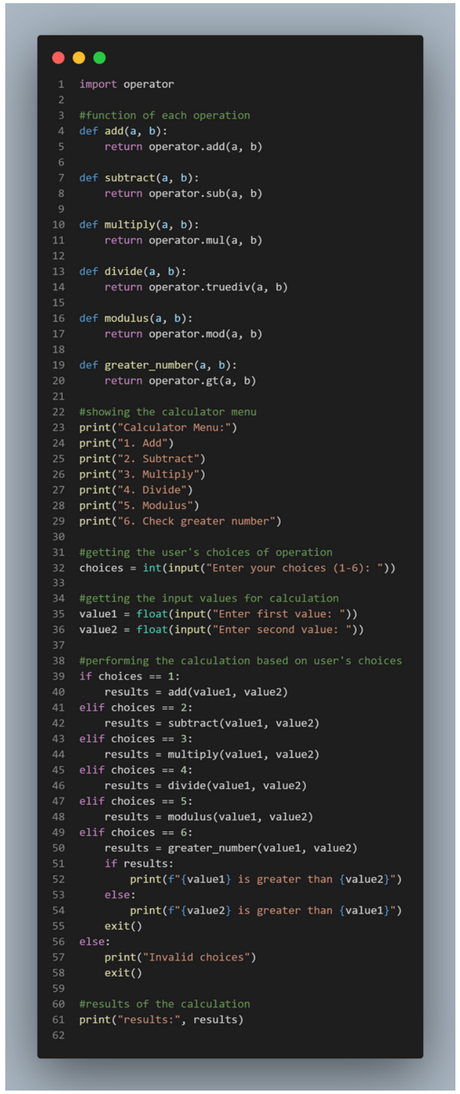


```python
import math

#ceil a number rounds it up to the nearest integer
a = 2.3
a_ceil = math.ceil(a)  # ceil function
print(f"Ceil of {a} is: {a_ceil}")

#floor of a number rounding it down to the nearest integer
a_floor = math.floor(a)  # floor function
print(f"Floor of {a} is: {a_floor}")

#factorial of the number
a = 5
a_factorial = math.factorial(a)  # factorial function
print(f"Factorial of {a} is: {a_factorial}")

#absolute value of the number
value = -2**3
value_absolute = abs(value)  # abs function
print(f"Absolute value of {value} is: {value_absolute}")

#square root the number
a = 16
squareroot_a = math.sqrt(a)  # square root function
print(f"Square root of {a} is: {squareroot_a}")

```

Exercise 2

```python
#listing the array
a = [20, 23, 82, 40, 32, 15, 67, 52]

#finding indices of even numbers
even_indice = [index for index, num in enumerate(a) if num % 2 == 0]
print("Indices of even numbers:", even_indice)

#sorting the arrays
sort_a = sorted(a)
print("Sorted array:", sort_a)

#slicing elements from index from 3 to the end of array
slicing_end = a[3:] #slicing the index from 3 to the end
print("Elements from index 3 to end:", slicing_end)

#slicing elements from index from 0 to the index 4
slicing_start_end = a[0:5]  #slicing the index from 0 to index 4 (exclusive)
print("Elements from index 0 to index 4:", slicing_start_end)

#printing [32, 15, 67] and using negative slicing
slice_negative = a[-5:-2]  #negative slicing to get the elements from index -5 to -2
print("Using negative slicing to get [32, 15, 67]:", slice_negative)

```

Exercise 3



```python
import operator

#function of each operation
def add(a, b):
    return operator.add(a, b)

def subtract(a, b):
    return operator.sub(a, b)

def multiply(a, b):
    return operator.mul(a, b)

def divide(a, b):
    return operator.truediv(a, b)

def modulus(a, b):
    return operator.mod(a, b)

def greater_number(a, b):
    return operator.gt(a, b)

#showing the calculator menu
print("Calculator Menu:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")
print("5. Modulus")
print("6. Check greater number")

#getting the user's choices of operation
choices = int(input("Enter your choices (1-6): "))

#getting the input values for calculation
value1 = float(input("Enter first value: "))
value2 = float(input("Enter second value: "))

#performing the calculation based on user's choices
if choices == 1:
    results = add(value1, value2)
elif choices == 2:
    results = subtract(value1, value2)
elif choices == 3:
    results = multiply(value1, value2)
elif choices == 4:
    results = divide(value1, value2)
elif choices == 5:
    results = modulus(value1, value2)
elif choices == 6:
    results = greater_number(value1, value2)
    if results:
        print(f"{value1} is greater than {value2}")
    else:
        print(f"{value2} is greater than {value1}")
    exit()
else:
    print("Invalid choices")
    exit()

#results of the calculation
print("results:", results)

```

Exercise 3(2)

```python
import matplotlib.pyplot as plt

#for the first line from (1, 2) to (6, 8)
x1_values = [1, 6]
y1_values = [2, 8]

#for the dotted line from (1, 3) to (2, 8) to (6, 1) to (8, 10)
values_x2 = [1, 2, 6, 8]
values_y2 = [3, 8, 1, 10]

#creating a new figure
plt.figure(figsize=(8, 6))

#plot of the first line (solid)
plt.plot(x1_values, y1_values, label='Line 1')

#plot of the dotted line
plt.plot(values_x2, values_y2, linestyle=':', marker='o', label='Dotted Line')

#x and y axis labels
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

#plot title
plt.title('Lines on a Diagram')

#display the legend
plt.legend()

# Display the plot
plt.show()

```

Exercise 4

```python
import json

#asking the user to enter their student details
name = input("Enter student name: ")
number_id = input("Enter student ID: ")
course = input("Enter student course: ")

#creating a dictionary with student details
details_student = {
    "Name": name,
    "ID": number_id,
    "course": course
}

#writing the student details to a JSON file
with open('StudentJson.json', 'w') as file:
    json.dump(details_student, file, indent=4)

#reading the contents from JSON
with open('StudentJson.json', 'r') as file:
    data_student = json.load(file)

#displaying the individual values
print("Details of the Student are")
print("\tName:", data_student['Name'])
print("\tID:", data_student['ID'])
print("\tcourse:", data_student['course'])

#updated coursedetails for student 1
details_course = {
    "Group": "A",
    "Year": 2
}

#updated students details dictionary with coursedetails
data_student["CourseDetails"] = details_course

#writing the updated data back to the JSON file
with open('StudentJson.json', 'w') as file:
    json.dump(data_student, file, indent=4)

#displaying individual values
print("\nDetails of the Student are")
print("\tName:", data_student['Name'])
print("\tID:", data_student['ID'])
print("\tcourse:", data_student['course'])
print("\tGroup:", data_student['CourseDetails']['Group'])
print("\tYear:", data_student['CourseDetails']['Year'])

```