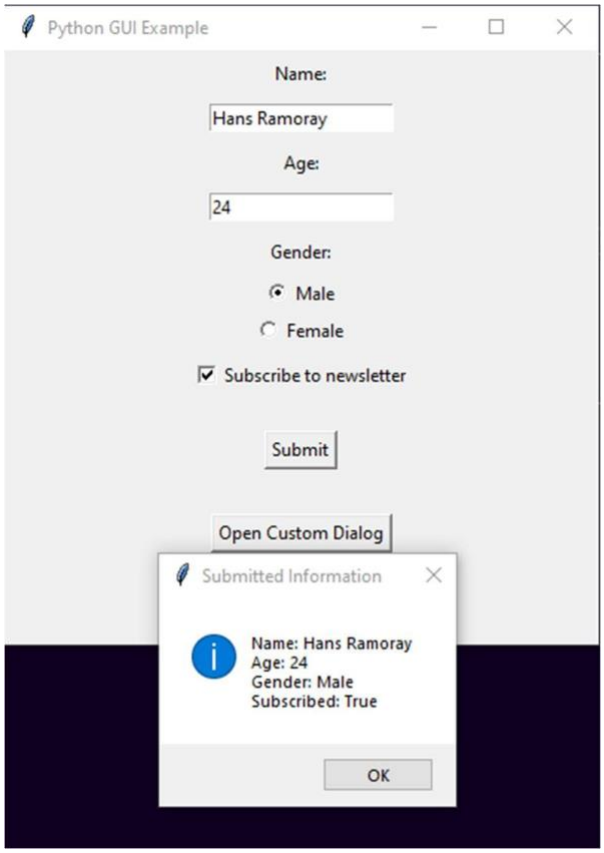
Ep-8

Anas Tanvar, 47, Batch-3

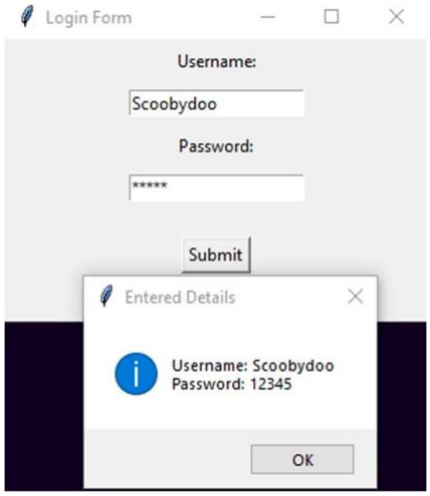
```
import socket import tkinter as tk
from tkinter import messagebox
def start_client():
    try:
        client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)         client_socket.connect(("127.0.0.1", 12345))
message = client_socket.recv(1024)         print(f"Received from server:
{message.decode()}")         client_socket.send("Hello from the
client!".encode())         client_socket.close()     except Exception as e:
print(f"Error: {e}")
def on_submit():
    name = entry_name.get()     age =
entry_age.get()     gender =
var_gender.get()     is_subscribed =
var_subscribe.get()
    messagebox.showinfo("Submitted Information", f"Name: {name}\nAge: {age}\nGender: {gender}\nSubscribed:
{is_subscribed}")
def custom_dialog():
    response = messagebox.askyesno("Custom Dialog", "Do you want to continue?")
messagebox.showinfo("Response", "You chose 'Yes'" if response else "You chose 'No'") root
= tk.Tk() root.title("Python GUI Example") root.geometry("400x400") label_name =
tk.Label(root, text="Name:") label_name.pack(pady=5) entry_name = tk.Entry(root)
entry_name.pack(pady=5)
label_age = tk.Label(root, text="Age:")
label_age.pack(pady=5) entry_age =
tk.Entry(root) entry_age.pack(pady=5)
label_gender = tk.Label(root, text="Gender:")
label_gender.pack(pady=5) var_gender =
tk.StringVar(value="Not selected")
radio_male = tk.Radiobutton(root, text="Male", variable=var_gender, value="Male")
radio_male.pack() radio_female = tk.Radiobutton(root, text="Female", variable=var_gender,
value="Female") radio_female.pack() var_subscribe = tk.BooleanVar(value=False)
checkbox_subscribe = tk.Checkbutton(root, text="Subscribe to newsletter", variable=var_subscribe)
checkbox_subscribe.pack(pady=5) button_submit = tk.Button(root, text="Submit",
command=on_submit) button_submit.pack(pady=20) button_dialog = tk.Button(root, text="Open
Custom Dialog", command=custom_dialog) button_dialog.pack(pady=10) if __name__ ==
'__main__':
    start_client()
```

root.mainloop()



Anas Tanvar, 47, Batch-3

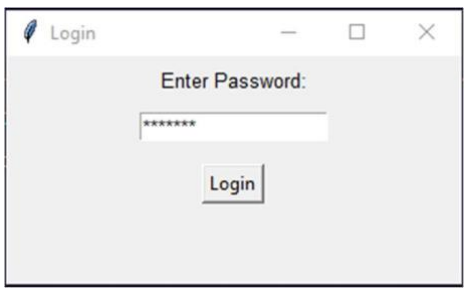
```
import tkinter as tk from tkinter
import messagebox def
display_text():
    username = entry_username.get()     password = entry_password.get()
messagebox.showinfo("Entered Details", f"Username: {username}\nPassword: {password}")
root = tk.Tk() root.title("Login Form") root.geometry("300x200") tk.Label(root,
text="Username:").pack(pady=5) entry_username = tk.Entry(root)
entry_username.pack(pady=5) tk.Label(root, text="Password:").pack(pady=5) entry_password
= tk.Entry(root, show="*") entry_password.pack(pady=5) tk.Button(root, text="Submit",
command=display_text).pack(pady=20)
root.mainloop()
```

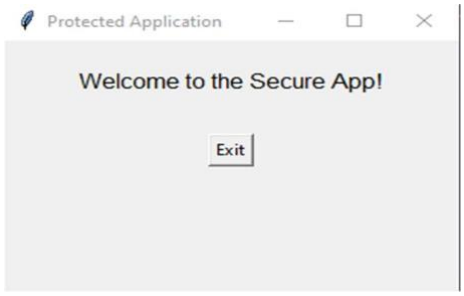


Anas Tanvar, 47, Batch-3

```
import tkinter as tk from tkinter
import messagebox def
check_password():
    if entry_password.get() == "admin123":
        login_window.destroy()
open_main_app()
    else:
        messagebox.showerror("Access Denied", "Incorrect Password!") def open_main_app():
main_app = tk.Tk()     main_app.title("Protected Application")
main_app.geometry("300x200")     tk.Label(main_app, text="Welcome to the Secure App!",
font=("Arial", 12)).pack(pady=20)     tk.Button(main_app, text="Exit",
command=main_app.destroy).pack(pady=10)     main_app.mainloop()

login_window = tk.Tk() login_window.title("Login")
login_window.geometry("300x150") tk.Label(login_window, text="Enter
Password:", font=("Arial", 10)).pack(pady=5) entry_password =
tk.Entry(login_window, show="*") entry_password.pack(pady=5)
tk.Button(login_window, text="Login", command=check_password).pack(pady=10)
login_window.mainloop()
```





Exp-9:

Anas Tanvar, 47,

Batch-3

```
class Stack:     def
__init__(self):
        self.st = []
  def isempty(self):
        return len(self.st) == 0
def push(self, element):
self.st.append(element)     def
pop(self):
        return -1 if self.isempty() else self.st.pop()
def peep(self):
        return -1 if self.isempty() else self.st[-1]
def search(self, element):
        if self.isempty():
            return -1
try:
            index = self.st.index(element)
return len(self.st) - index          except
ValueError:
            return -2     def display(self):          return "Stack is empty" if self.isempty() else
"\n".join(map(str, reversed(self.st))) stack = Stack()
stack.push(10) stack.push(20) stack.push(30)
print("Stack contents:\n", stack.display())
print("\nTop element (Peep):", stack.peep())
print("\nPop element:", stack.pop())
print("\nStack after pop:\n", stack.display())
print("\nSearch for 10:", stack.search(10))
print("Search for 40:", stack.search(40))
stack.pop() stack.pop()
print("\nStack after popping all elements:\n", stack.display())
```

```
PS C:\Users\HP\Desktop\Python> python -u "c:\Users\HP\Desktop\Python\Ep-9.py"
Stack contents:
 30
20
10

Top element (Peep): 30

Pop element: 30

Stack after pop:
 20
10

Search for 10: 2
Search for 40: -2

Stack after popping all elements:
 Stack is empty
PS C:\Users\HP\Desktop\Python>
```

Ep-9a:queue

Anas Tanvar, 47, Batch-3

```
class Queue:
    def __init__(self):
self.q = []
    def is_empty(self):
        return len(self.q) == 0
def add(self, element):
self.q.append(element)     def
delete(self):
        return -1 if self.is_empty() else self.q.pop(0)
def search(self, element):         if self.is_empty():
            return -1
        try:
            return self.q.index(element) + 1
        except ValueError:
            return -2
def display(self):
        return "Queue is empty" if self.is_empty() else " -> ".join(map(str, self.q))
q     =     Queue()
choice  =  0  while
choice != 4:
    print("\nQUEUE OPERATIONS\n1. Add Element\n2. Delete Element\n3. Search Element\n4. Exit")
    try:
        choice = int(input("Enter Your Choice: "))
except ValueError:
        print("Invalid input! Please enter a number between 1-4.")
        continue
if choice == 1:
        try:
            element = int(input("Enter Element: "))
            q.add(element)
except ValueError:
            print("Invalid input! Please enter a valid number.")
elif choice == 2:
        element = q.delete()         print("Queue is empty" if element == -1 else
f"Deleted Element: {element}")
    elif choice == 3:
        try:
```

```
            element = int(input("Enter Element: "))
pos = q.search(element)
            if pos == -1:
                print("Queue is empty")
            elif pos == -2:
                print("Element not found in the queue")
            else:
                print(f"Element found at position: {pos}")
except ValueError:
            print("Invalid input! Please enter a valid number.")
elif choice == 4:
        print("Exiting program...")
        break
else:
        print("Invalid choice! Please select a valid option.")
    print("Queue =", q.display())
```

```
QUEUE OPERATIONS
1. Add Element
2. Delete Element
3. Search Element
4. Exit
Enter Your Choice: 1
Enter Element: 2
Queue = 2

QUEUE OPERATIONS
1. Add Element
2. Delete Element
3. Search Element
4. Exit
Enter Your Choice: 1
Enter Element: 4
Queue = 2 -> 4
```

```
QUEUE OPERATIONS
1. Add Element
2. Delete Element
3. Search Element
4. Exit
Enter Your Choice: 3
Enter Element: 2
Element found at position: 1
Queue = 2 -> 4
```

Ep-9c:Linked list

Anas Tanvar, 47, Batch-3

```
class Node:     def
__init__(self, data):
        self.data       =       data
self.next    =    None    class
LinkedList:                  def
__init__(self):         self.head
=  None          def  insert(self,
data):               new_node  =
Node(data)
```

```
        if self.head is None:
self.head = new_node
        else:
            temp = self.head
while temp.next:                temp =
temp.next            temp.next =
new_node     def delete(self, key):
temp = self.head         if temp and
temp.data == key:
            self.head = temp.next
return True         prev = None
while temp and temp.data != key:
            prev = temp
            temp = temp.next
if temp is None:
return False
        prev.next = temp.next
return True     def
search(self, key):
temp = self.head
        pos = 1         while
temp:             if temp.data
== key:                 return
pos             temp =
temp.next
            pos += 1         return
-1     def display(self):
temp = self.head         if not
temp:             return "List is
empty"         result = []
        while temp:
            result.append(str(temp.data))
temp = temp.next         return " ->
".join(result) ll = LinkedList() choice
= 0 while choice != 4:
    print("\nLINKED LIST OPERATIONS\n1. Insert\n2. Delete\n3. Search\n4. Exit")
    try:
        choice = int(input("Enter Your Choice: "))
except ValueError:
```

```
        print("Invalid input! Please enter a number between 1-4.")
continue     if choice == 1:
        try:
            data = int(input("Enter Element: "))
            ll.insert(data)
        except ValueError:
            print("Invalid input! Please enter a valid number.")
elif choice == 2:
        try:
            key = int(input("Enter Element to Delete: "))            print("Element
deleted" if ll.delete(key) else "Element not found")        except ValueError:
            print("Invalid input! Please enter a valid number.")
elif choice == 3:
        try:
            key = int(input("Enter Element to Search: "))
pos = ll.search(key)
            print(f"Element found at position {pos}" if pos != -1 else "Element not found")
except ValueError:
            print("Invalid input! Please enter a valid number.")
elif choice == 4:
        print("Exiting program...")
        break
else:
        print("Invalid choice! Please select a valid option.")
    print("Linked List =", ll.display())
```

```
LINKED LIST OPERATIONS
1. Insert
2. Delete
3. Search
4. Exit
Enter Your Choice: 1
Enter Element: 2
Linked List = 2

LINKED LIST OPERATIONS
1. Insert
2. Delete
3. Search
4. Exit
Enter Your Choice: 1
Enter Element: 7
Linked List = 2 -> 7
```

```
LINKED LIST OPERATIONS
1. Insert
2. Delete
3. Search
4. Exit
Enter Your Choice: 2
Enter Element to Delete: 2
Element deleted
Linked List = 7
```

Post-Lab

Reverse stack

Anas Tanvar, 47,

Batch-3

```
class Stack:     def
__init__(self):
self.stack = []
    def is_empty(self):
        return len(self.stack) == 0
    def push(self, item):
self.stack.append(item)     def
pop(self):
        return self.stack.pop() if not self.is_empty() else None
def peek(self):
        return self.stack[-1] if not self.is_empty() else None
def display(self):
        return self.stack[::-1] if not self.is_empty() else "Stack is empty"
def insert_at_bottom(stack, item):     if stack.is_empty():
stack.push(item)
    else:
        temp = stack.pop()
insert_at_bottom(stack, item)
stack.push(temp) def
reverse_stack(stack):     if not
stack.is_empty():         temp =
stack.pop()
reverse_stack(stack)
insert_at_bottom(stack, temp) s =
Stack() elements = [1, 2, 3, 4, 5] for
el in elements:
    s.push(el) print("Original
Stack:", s.display())
reverse_stack(s) print("Reversed
Stack:", s.display())
```

```
Original Stack: [5, 4, 3, 2, 1]
Reversed Stack: [1, 2, 3, 4, 5]

=== Code Execution Successful ===
```

Circular queue

Anas Tanvar, 47, Batch-3

```
class CircularQueue:     def
__init__(self, size):
        self.size = size
        self.queue = [None] * size
self.front = self.rear = -1
    def is_full(self):
        return (self.rear + 1) % self.size == self.front
def is_empty(self):
        return self.front == -1
def enqueue(self, item):
        if self.is_full():
            print("Queue is
full!")             return         if
self.is_empty():
            self.front = self.rear = 0
        else:
            self.rear = (self.rear + 1) %
self.size         self.queue[self.rear] = item
def dequeue(self):         if self.is_empty():
print("Queue is empty!")              return
None         item = self.queue[self.front]
if self.front == self.rear:              self.front
= self.rear = -1
        else:
            self.front = (self.front + 1) %
self.size         return item     def
display(self):         if self.is_empty():
return "Queue is empty"         elements = []
i = self.front         while True:
            elements.append(str(self.queue[i]))
            if i == self.rear:
                break             i =
(i + 1) % self.size
        return      "      ->      ".join(elements)      size      =
int(input("Enter the size of Circular Queue: ")) cq =
CircularQueue(size) choice = 0 while choice != 4:
    print("\nCIRCULAR QUEUE OPERATIONS\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit")
    try:
        choice = int(input("Enter Your Choice: "))
except ValueError:
```

```
        print("Invalid input! Please enter a number between 1-4.")
        continue
if choice == 1:
        try:
            element = int(input("Enter Element: "))
cq.enqueue(element)         except ValueError:
            print("Invalid input! Please enter a valid number.")
elif choice == 2:
        element = cq.dequeue()         print("Dequeued Element:", element if
element is not None else "")
    elif choice == 3:
        print("Circular Queue =", cq.display())
elif choice == 4:
        print("Exiting program...")         break     else:
print("Invalid choice! Please select a valid option.")
```

```
Enter the size of Circular Queue: 1

CIRCULAR QUEUE OPERATIONS
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter Your Choice: 1
Enter Element: 3

CIRCULAR QUEUE OPERATIONS
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter Your Choice: 3
Circular Queue = 3
```