Code:

```
#Code1
import socket


def server_program():
    # get the hostname
    host = socket.gethostname()
    port = 5000  # initiate port no above 1024

    server_socket = socket.socket()  # get instance
    # look closely. The bind() function takes tuple as argument
    server_socket.bind((host, port))  # bind host address and port together

    # configure how many client the server can listen simultaneously
    server_socket.listen(2)
    conn, address = server_socket.accept()  # accept new connection
    print("Connection from: " + str(address))
    while True:
        # receive data stream. it won't accept data packet greater than 1024 bytes
        data = conn.recv(1024).decode()
        if not data:
            # if data is not received break
            break
        print("from connected user: " + str(data))
        data = input(' -> ')
        conn.send(data.encode())  # send data to the client

    conn.close()  # close the connection


if __name__ == '__main__':
    server_program()


#Code2
import socket


def client_program():
    host = socket.gethostname()  # as both code is running on same pc
    port = 5000  # socket server port number
```

```
    client_socket = socket.socket()  # instantiate
    client_socket.connect((host, port))  # connect to the server

    message = input(" -> ")  # take input

    while message.lower().strip() != 'bye':
        client_socket.send(message.encode())  # send message
        data = client_socket.recv(1024).decode()  # receive response

        print('Received from server: ' + data)  # show in terminal

        message = input(" -> ")  # again take input

    client_socket.close()  # close the connection


if __name__ == '__main__':
    client_program()
```

Outputs:

```
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python> python -u "c
:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python\Program7a.py"
Connection from: ('192.168.56.1', 62364)
from connected user: Hello
 -> Hello
from connected user: Hi
 -> Hi
from connected user: This is a message from client to server
 -> This is a message from server to client.
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python>
```

```
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python> python .\Pro
gram7b.py
 -> Hello
Received from server: Hello
 -> Hi
Received from server: Hi
 -> This is a message from client to server
Received from server: This is a message from server to client.
 -> bye
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python>
```

Code:

```
#Code1
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User


# Create your forms here.

class NewUserForm(UserCreationForm):
	email = forms.EmailField(required=True)

	class Meta:
		model = User
		fields = ("username", "email", "password1", "password2")

	def save(self, commit=True):
		user = super(NewUserForm, self).save(commit=False)
		user.email = self.cleaned_data['email']
		if commit:
			user.save()
		return user

#Code2
from django.urls import path
from . import views

app_name = "main"


urlpatterns = [
   path("", views.homepage, name="homepage"),
   path("register", views.register_request, name="register")
]

#Code3
from django.shortcuts import  render, redirect
from .forms import NewUserForm
from django.contrib.auth import login
from django.contrib import messages

def register_request(request):
	if request.method == "POST":
```

```
                form = NewUserForm(request.POST)
                if form.is_valid():
                        user = form.save()
                        login(request, user)
                        messages.success(request, "Registration successful." )
                        return redirect("main:homepage")
                messages.error(request, "Unsuccessful registration. Invalid information.")
        form = NewUserForm()
        return render (request=request, template_name="main/register.html",
context={"register_form":form})

#Code4
from django.urls import path
from . import views

app_name = "main"


urlpatterns = [
   path("", views.homepage, name="homepage"),
   path("register", views.register_request, name="register"),
   path("login", views.login_request, name="login")
]

#Code5
from django.shortcuts import  render, redirect
from .forms import NewUserForm
from django.contrib.auth import login, authenticate #add this
from django.contrib import messages
from django.contrib.auth.forms import AuthenticationForm #add this

def register_request(request):
        ...

def login_request(request):
        if request.method == "POST":
                form = AuthenticationForm(request, data=request.POST)
                if form.is_valid():
                        username = form.cleaned_data.get('username')
                        password = form.cleaned_data.get('password')
                        user = authenticate(username=username, password=password)
                        if user is not None:
                                login(request, user)
                                messages.info(request, f"You are now logged in as {username}.")
                                return redirect("main:homepage")
```

```
                        else:
                                messages.error(request,"Invalid username or password.")
                else:
                        messages.error(request,"Invalid username or password.")
        form = AuthenticationForm()
        return render(request=request, template_name="main/login.html",
context={"login_form":form})
```

HTML Files:

#File1: register.html

```
{% extends "main/header.html" %}

{% block content %}

{% load crispy_forms_tags %}

<!--Register-->
<div class="container py-5">
        <h1>Register</h1>
        <form method="POST">
                {% csrf_token %}
                {{ register_form|crispy }}
                <button class="btn btn-primary" type="submit">Register</button>
        </form>
        <p class="text-center">If you already have an account, <a href="/login">login</a>
instead.</p>
</div>

{% endblock %}
```

#File2: login.html

```
{% extends "main/header.html" %}

{% block content %}

{% load crispy_forms_tags %}

<!--Login-->
<div class="container py-5">
  <h1>Login</h1>
  <form method="POST">
    {% csrf_token %}
```

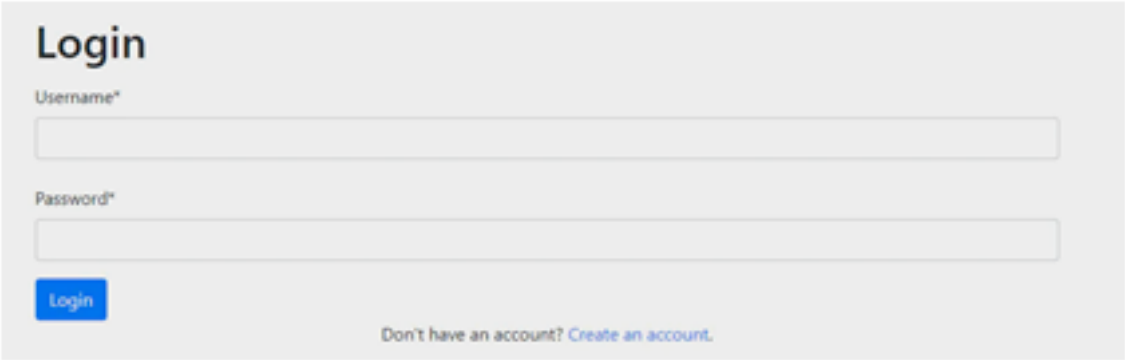
```
  {{ login_form|crispy }}
  <button class="btn btn-primary" type="submit">Login</button>
 </form>
 <p class="text-center">Don't have an account? <a href="/register">Create an account</a>.</p>
</div>

{% endblock %}
```

Outputs:


```
(env) C:\Users\Owner\Desktop\Code\env\mysite>py manage.py createsuperuser
Username (leave blank to use 'owner'): owner
Email address:
Password: *****
Password (again): *****
Superuser created successfully.

(env) C:\Users\Owner\Desktop\Code\env\mysite>py manage.py runserver
```



Code:

```
#Code1
# Python program to illustrate the concept
# of threading
# importing the threading module
import threading

def print_cube(num):
   # function to print cube of given num
   print("Cube: {}" .format(num * num * num))

def print_square(num):
   # function to print square of given num
   print("Square: {}" .format(num * num))

if __name__ =="__main__":
   # creating thread
   t1 = threading.Thread(target=print_square, args=(10,))
   t2 = threading.Thread(target=print_cube, args=(10,))
   # starting thread 1
   t1.start()
   # starting thread 2
   t2.start()
   # wait until thread 1 is completely executed
   t1.join()
   # wait until thread 2 is completely executed
   t2.join()
   # both threads completely executed
   print("Done!")


#Code2
# Python program to illustrate the concept
# of threading
import threading
import os
def task1():
   print("Task 1 assigned to thread: {}".format(threading.current_thread().name))
   print("ID of process running task 1: {}".format(os.getpid()))
def task2():
   print("Task 2 assigned to thread: {}".format(threading.current_thread().name))
   print("ID of process running task 2: {}".format(os.getpid()))
if __name__ == "__main__":
```

```
# print ID of current process
print("ID of process running main program: {}".format(os.getpid()))
# print name of main thread
print("Main thread name: {}".format(threading.current_thread().name))
# creating threads
t1 = threading.Thread(target=task1, name='t1')
t2 = threading.Thread(target=task2, name='t2')
# starting threads
t1.start()
t2.start()
# wait until all threads finish
t1.join()
t2.join()
```

Output:

```
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python> python -u "c:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python\Program9a.py"
Square: 100
Cube: 1000
Done!
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python> python -u "c:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python\Program9b.py"
ID of process running main program: 6032
Main thread name: MainThread
Task 1 assigned to thread: t1
Task 2 assigned to thread: t2ID of process running task 1: 6032

ID of process running task 2: 6032
```

Code:

```
# Python program to demonstrate
# array creation techniques
import numpy as np
# Creating array from list with type float
a = np.array([[1, 2, 4], [5, 8, 7]], dtype = 'float')
print ("Array created using passed list:\n", a)
# Creating array from tuple
b = np.array((1 , 3, 2))
print ("\nArray created using passed tuple:\n", b)
# Creating a 3X4 array with all zeros
c = np.zeros((3, 4))
print ("\nAn array initialized with all zeros:\n", c)
# Create a constant value array of complex type
d = np.full((3, 3), 6, dtype = 'complex')
print ("\nAn array initialized with all 6s."
"Array type is complex:\n", d)
# Create an array with random values
e = np.random.random((2, 2))
print("\nA random array:\n", e)
# Create a sequence of integers
# from 0 to 30 with steps of 5
f = np.arange(0, 30, 5)
print("\nA sequential array with steps of 5:\n", f)
# Create a sequence of 10 values in range 0 to 5
g = np.linspace(0, 5, 10)
print ("\nA sequential array with 10 values between"
"0 and 5:\n", g)
# Reshaping 3X4 array to 2X2X3 array
arr = np.array([[1, 2, 3, 4],
[5, 2, 4, 2],
[1, 2, 0, 1]])
newarr = arr.reshape(2, 2, 3)
print ("\nOriginal array:\n", arr)
print ("Reshaped array:\n", newarr)
# Flatten array
arr = np.array([[1, 2, 3], [4, 5, 6]])
flarr = arr.flatten()
print("\nOriginal array:\n", arr)
print("Flattened array:\n", flarr)
```

Output:

```
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python> python -u "c:\Users\USER\Deskt
op\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python\Program10.py"
Array created using passed list:
 [[1. 2. 4.]
 [5. 8. 7.]]

Array created using passed tuple:
 [1 3 2]

An array initialized with all zeros:
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]

An array initialized with all 6s.Array type is complex:
 [[6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]]

A random array:
 [[0.91118821 0.37851706]
 [0.45198546 0.25834059]]

A sequential array with steps of 5:
 [ 0  5 10 15 20 25]

A sequential array with 10 values between0 and 5:
 [0.         0.55555556 1.11111111 1.66666667 2.22222222 2.77777778
 3.33333333 3.88888889 4.44444444 5.        ]

Original array:
 [[1 2 3 4]
 [5 2 4 2]
  [1 2 0 1]]
 Reshaped array:
  [[[1 2 3]
   [4 5 2]]

  [[4 2 1]
   [2 0 1]]]

 Original array:
  [[1 2 3]
  [4 5 6]]
 Flattened array:
  [1 2 3 4 5 6]
```

Code:

```
# Python program to demonstrate
# basic operations on single array
import numpy as np

# Defining Array 1
a = np.array([[1, 2],[3, 4]])

# Defining Array 2cls

b = np.array([[4, 3],[2, 1]])

# Adding 1 to every element
print("Original Array: ",a)
print ("Adding 1 to every element:", a + 1)

# Subtracting 2 from each element
print("\nOriginal Array: ",b)
print ("Subtracting 2 from each element:", b - 2)

# sum of array elements
# Performing Unary operations
print("\nGiven Array: ",a)
print ("Sum of all array elements: ", a.sum())

# Adding two arrays
# Performing Binary operations
print("\n",a)
print("+")
print(b)
print ("\nArray sum:\n", a + b)
```

Output:

```
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python> python -u "c:\Users\USER\Deskt
op\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python\Program11.py"
Original Array:  [[1 2]
 [3 4]]
Adding 1 to every element: [[2 3]
 [4 5]]

Original Array:  [[4 3]
 [2 1]]
Subtracting 2 from each element: [[ 2  1]
 [ 0 -1]]

Given Array:  [[1 2]
 [3 4]]
Sum of all array elements:  10

 [[1 2]
 [3 4]]
+
[[4 3]
 [2 1]]

Array sum:
 [[5 5]
 [5 5]]
```

Code:

```
import pandas
##### INTIALIZATION #####
#STRING SERIES
fruits = pandas.Series(["apples", "oranges", "bananas"])
print("Fruit series:")
print(fruits)
#FLOAT SERIES
temperature = pandas.Series([32.6, 34.1, 28.0, 35.9])
print("\nTemperature series:")
print(temperature)
#INTEGER SERIES
factors_of_12 = pandas.Series([1,2,4,6,12])
print("\nFactors of 12 series:")
print(factors_of_12)
print("Type of this data structure is:", type(factors_of_12))

#FLOAT & INTEGER FRAME
temp_fact = {'col1':factors_of_12, 'col2':temperature}
result = pandas.DataFrame(data = temp_fact)
print("\nTemperature & Factors of 12 series combined in a frame: ")
print(result)
print("Type of this data structure is:", type(result))
```

Output:

```
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python> python -u "c:\Users\USER\Deskt
op\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python\Program12.py"
Fruit series:
0     apples
1    oranges
2    bananas
dtype: object

Temperature series:
0    32.6
1    34.1
2    28.0
3    35.9
dtype: float64

Factors of 12 series:
0     1
1     2
2     4
3     6
4    12
dtype: int64
Type of this data structure is: <class 'pandas.core.series.Series'>

Temperature & Factors of 12 series combined in a frame:
   col1  col2
0     1  32.6
1     2  34.1
2     4  28.0
3     6  35.9
4    12   NaN
Type of this data structure is: <class 'pandas.core.frame.DataFrame'>
```

Code:

```
#Code 1
import smtplib, ssl
import getpass as gp
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

# Defining HTML Doc:
html = """
This is an e-mail message to be sent in HTML format
<html>
<body>
   <b>This is an HTML message.</b>
   <h1>This is a heading.</h1>
</body>
</html>
"""

# Defining Required Details
smtp_server = "smtp.gmail.com"
port = 465

sender = "2021ca06f@sigce.edu.in"
receiver = "2021ca21f@sigce.edu.in"
password = gp.getpass("Enter your password (2021ca06f@sigce.edu.in): ")

# Create a MIMEMultipart class, and set up the From, To, Subject fields
email_message = MIMEMultipart()
email_message['From'] = sender
email_message['To'] = receiver
email_message['Subject'] = "SMTP HTML e-mail test"

# Attach the html doc defined earlier, as a MIMEText html content type to the MIME message
email_message.attach(MIMEText(html, "html"))
# Convert it as a string
email_string = email_message.as_string()

context = ssl.create_default_context()

with smtplib.SMTP_SSL(smtp_server, port, context = context) as server:
   server.login(sender, password)

   #sending the email:
```

```
    server.sendmail(sender, receiver, email_string)


#Code 2
import smtplib, ssl
import getpass as gp
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.application import MIMEApplication

# Open the attachment file for reading in binary mode (using 'rb'), and make it a MIMEApplication
class
def file_attacher(email_message, file_name):
    with open(file_name, 'rb') as f:
        file_attachment = MIMEApplication(f.read())
        # Add header/name to the attachments
        file_attachment.add_header("Content-Disposition",f"attachment; filename = {file_name}")
        # Attach the file to the message
        email_message.attach(file_attachment)

# Defining required details
smtp_server = "smtp.gmail.com"
port = "465"
html = """
This message contains an attachment, html enclosed text and simple text(this sentence).
<h1 style = "color:#045803;">Hello Prathamesh!</h1><br>
<p> This is the second Program file for <b>Python Experiment 13<b>. </p>
<p> The Python program used to send this email itself is the second program file. Do<b>reply if
you want the second program file.</b></p>
"""
sender = "2021ca06f@sigce.edu.in"
receiver = "2021ca69f@sigce.edu.in"
password = gp.getpass(f"Enter your app password for {sender}: ")

# Create a MIMEMultipart class, and set up the From, To, Subject fields
email_message = MIMEMultipart()
email_message['From'] = sender
email_message['To'] = receiver
email_message['Subject'] = "SMTP e-mail test (HTML, File Attachments)"

# Attach the html doc defined earlier, as a MIMEText html content type to the MIME message
email_message.attach(MIMEText(html, "html"))

# Attaching a file to the email message using function
file_attacher(email_message, "Program13a.py")
```

```
# Convert it as a string
email_string = email_message.as_string()

context = ssl.create_default_context()

with smtplib.SMTP_SSL(smtp_server, port, context = context) as server:
    server.login(sender, password)

    #sending the email:
    server.sendmail(sender, receiver, email_string)
```

Outputs:

```
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python> python -u "c
:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python\Program13a.py"
Enter your password (2021ca06f@sigce.edu.in):
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python> python -u "c
:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python\Program13b.py"
Enter your app password for 2021ca06f@sigce.edu.in:
PS C:\Users\USER\Desktop\Sufi Folder\College Works\Important\Practicals & Projects (GH)\Programs in College\Python>
```