ASSIGNMENT-4

COURIER MANAGEMENT SYSTEM

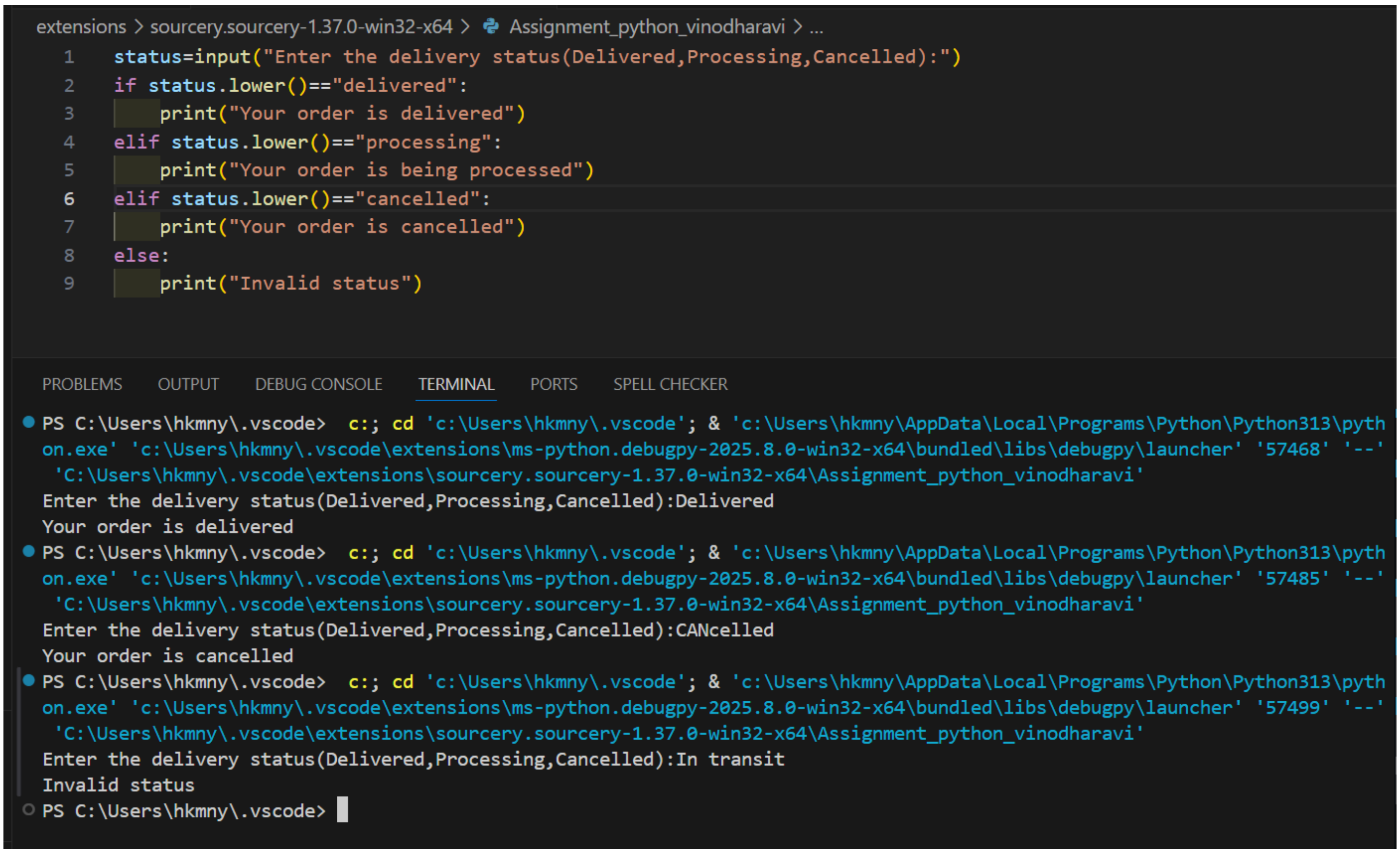
(Coding Part)

BY

VINODHA R

Coding
Task 1:
Control Flow Statements
1. Write a program that checks whether a given order is delivered or not based on its status (e.g., "Processing," "Delivered," "Cancelled"). Use if-else statements for this.
Code:

```
status=input("Enter the delivery status(Delivered,Processing,Cancelled):")
if status.lower()=="delivered":
   print("Your order is delivered")
elif status.lower()=="processing":
   print("Your order is being processed")
elif status.lower()=="cancelled":
   print("Your order is cancelled")
else:
   print("Invalid status")
```

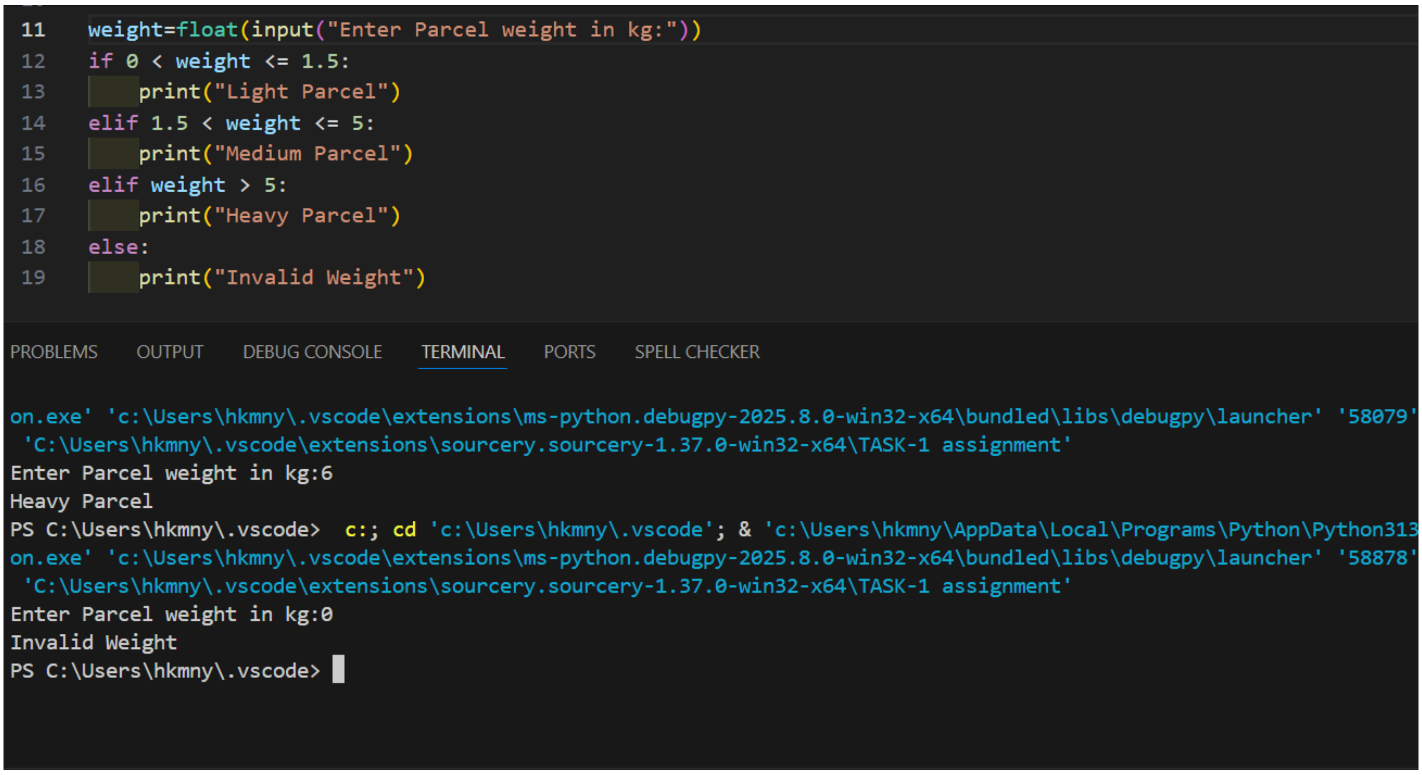

extensions > sourcery.sourcery-1.37.0-win32-x64 > Assignment_python_vinodharavi > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SPELL CHECKER

```
status=input("Enter the delivery status(Delivered,Processing,Cancelled):")
if status.lower()=="delivered":
    print("Your order is delivered")
elif status.lower()=="processing":
    print("Your order is being processed")
elif status.lower()=="cancelled":
    print("Your order is cancelled")
else:
    print("Invalid status")
```

```
PS C:\Users\hkmny\.vscode>  c:; cd 'c:\Users\hkmny\.vscode'; & 'c:\Users\hkmny\AppData\Local\Programs\Python\Python313\pyth
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '57468' '--'
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\Assignment_python_vinodharavi'
Enter the delivery status(Delivered,Processing,Cancelled):Delivered
Your order is delivered
PS C:\Users\hkmny\.vscode>  c:; cd 'c:\Users\hkmny\.vscode'; & 'c:\Users\hkmny\AppData\Local\Programs\Python\Python313\pyth
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '57485' '--'
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\Assignment_python_vinodharavi'
Enter the delivery status(Delivered,Processing,Cancelled):CANcelled
Your order is cancelled
PS C:\Users\hkmny\.vscode>  c:; cd 'c:\Users\hkmny\.vscode'; & 'c:\Users\hkmny\AppData\Local\Programs\Python\Python313\pyth
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '57499' '--'
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\Assignment_python_vinodharavi'
Enter the delivery status(Delivered,Processing,Cancelled):In transit
Invalid status
PS C:\Users\hkmny\.vscode> 
```

2. Implement a switch-case statement to categorize parcels based on their weight into "Light," "Medium," or "Heavy."
Code:

```
weight=float(input("Enter Parcel weight in kg:"))
if 0 < weight <= 1.5:
    print("Light Parcel")
elif 1.5 < weight <= 5:
    print("Medium Parcel")
elif weight > 5:
    print("Heavy Parcel")
else:
    print("Invalid Weight")
```



```
weight=float(input("Enter Parcel weight in kg:"))
if 0 < weight <= 1.5:
    print("Light Parcel")
elif 1.5 < weight <= 5:
    print("Medium Parcel")
elif weight > 5:
    print("Heavy Parcel")
else:
    print("Invalid Weight")
```

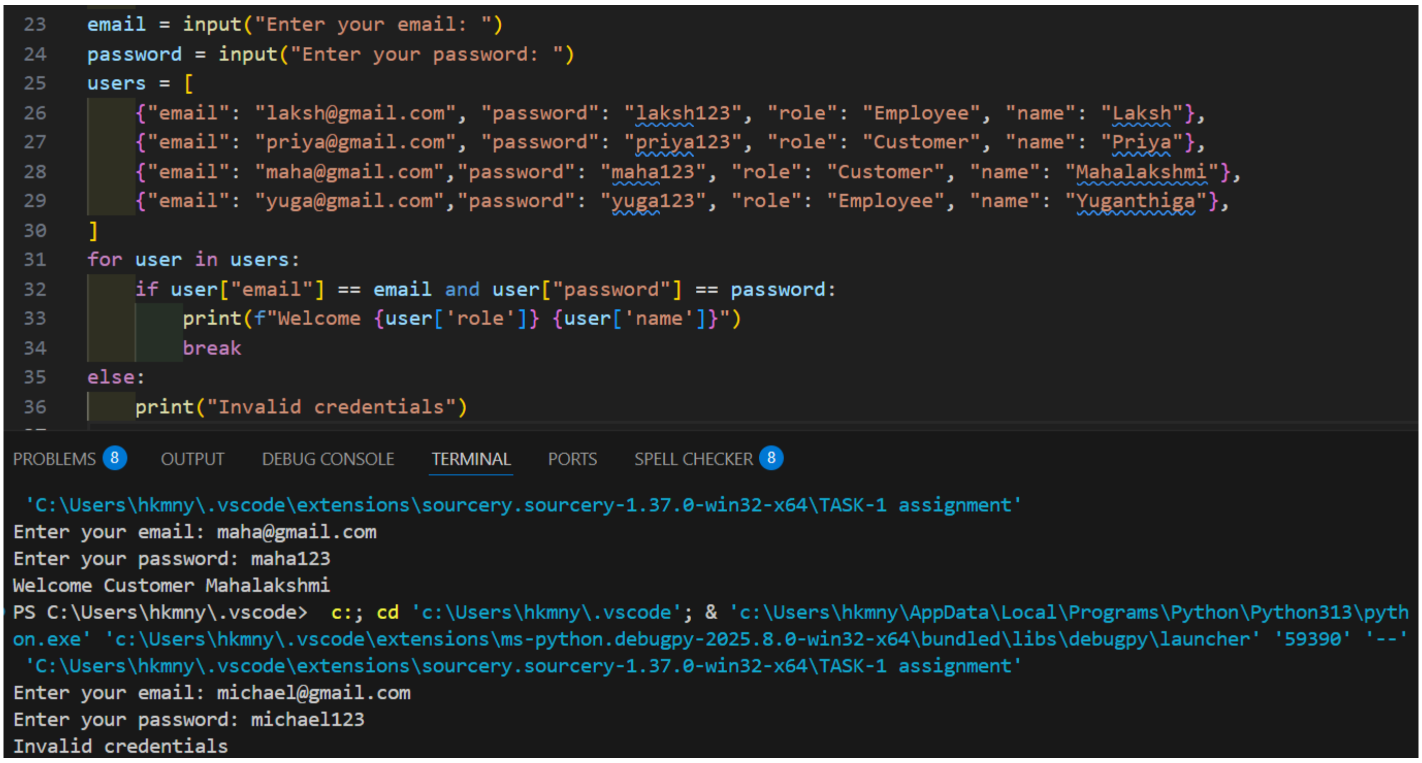
```
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '58079'
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TASK-1 assignment'
Enter Parcel weight in kg:6
Heavy Parcel
PS C:\Users\hkmny\.vscode>  c:; cd 'c:\Users\hkmny\.vscode'; & 'c:\Users\hkmny\AppData\Local\Programs\Python\Python313
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '58878'
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TASK-1 assignment'
Enter Parcel weight in kg:0
Invalid Weight
PS C:\Users\hkmny\.vscode> 
```

3. Implement User Authentication 1. Create a login system for employees and customers using Java control flow statements.

Code:

```
email = input("Enter your email: ")
password = input("Enter your password: ")
users = [
    {"email": "laksh@gmail.com", "password": "laksh123", "role": "Employee", "name":
"Laksh"},
    {"email": "priya@gmail.com", "password": "priya123", "role": "Customer", "name":
"Priya"},
    {"email": "maha@gmail.com","password": "maha123", "role": "Customer", "name":
"Mahalakshmi"},
    {"email": "yuga@gmail.com","password": "yuga123", "role": "Employee", "name":
"Yuganthiga"},
]
```

```
for user in users:
    if user["email"] == email and user["password"] == password:
        print(f"Welcome {user['role']} {user['name']}")
        break
else:
    print("Invalid credentials")
```



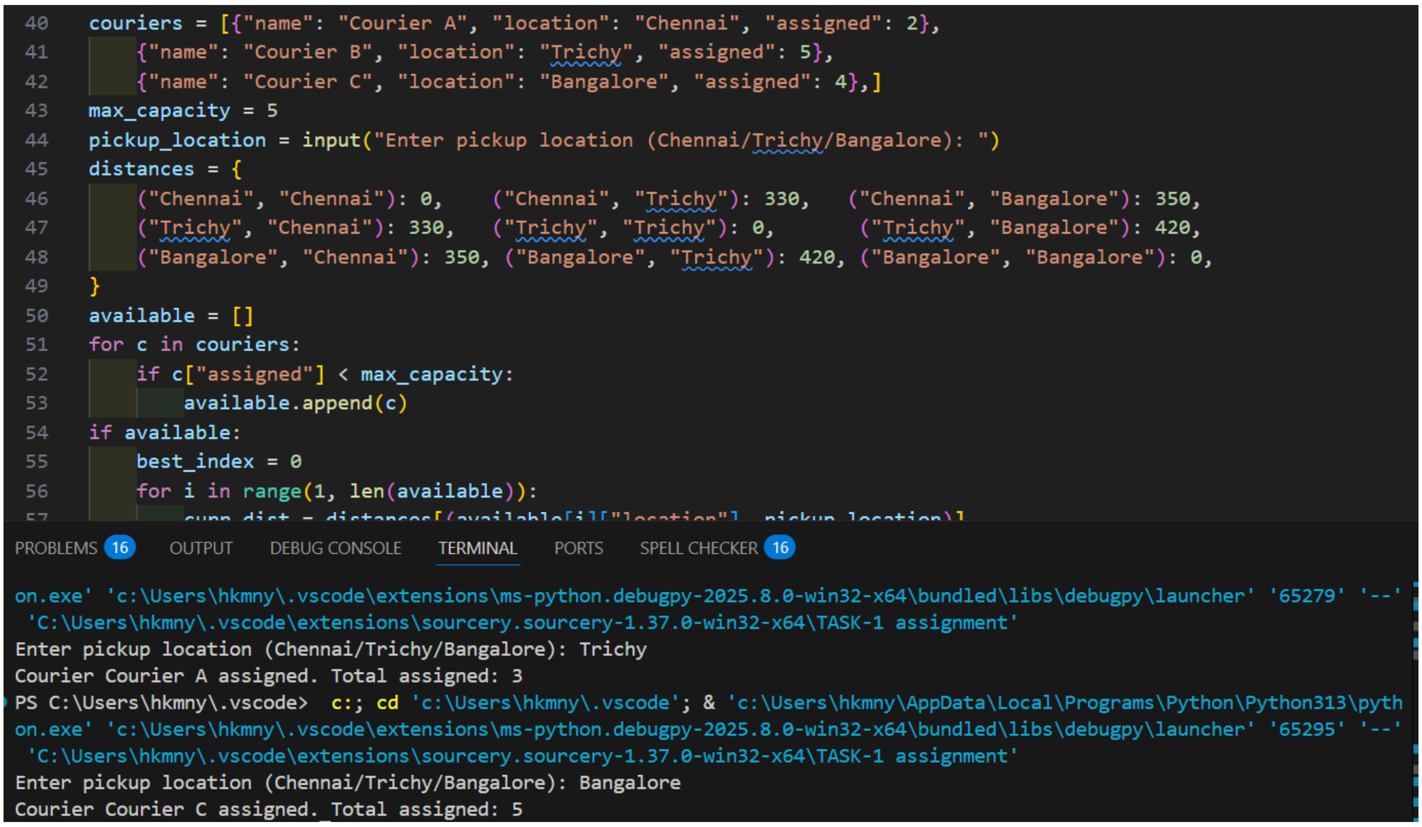
```
email = input("Enter your email: ")
password = input("Enter your password: ")
users = [
    {"email": "laksh@gmail.com", "password": "laksh123", "role": "Employee", "name": "Laksh"},
    {"email": "priya@gmail.com", "password": "priya123", "role": "Customer", "name": "Priya"},
    {"email": "maha@gmail.com","password": "maha123", "role": "Customer", "name": "Mahalakshmi"},
    {"email": "yuga@gmail.com","password": "yuga123", "role": "Employee", "name": "Yuganthiga"},
]
for user in users:
    if user["email"] == email and user["password"] == password:
        print(f"Welcome {user['role']} {user['name']}")
        break
else:
    print("Invalid credentials")
```

```
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TASK-1 assignment'
Enter your email: maha@gmail.com
Enter your password: maha123
Welcome Customer Mahalakshmi
PS C:\Users\hkmny\.vscode>  c:; cd 'c:\Users\hkmny\.vscode'; & 'c:\Users\hkmny\AppData\Local\Programs\Python\Python313\pyth
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '59390' '--'
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TASK-1 assignment'
Enter your email: michael@gmail.com
Enter your password: michael123
Invalid credentials
```

4. Implement Courier Assignment Logic 1. Develop a mechanism to assign couriers to shipments based on predefined criteria (e.g., proximity, load capacity) using loops.

Code:

```
couriers = [{"name": "Courier A", "location": "Chennai", "assigned": 2},
    {"name": "Courier B", "location": "Trichy", "assigned": 5},
    {"name": "Courier C", "location": "Bangalore", "assigned": 4},]
max_capacity = 5
pickup_location = input("Enter pickup location (Chennai/Trichy/Bangalore): ")
distances = {
    ("Chennai", "Chennai"): 0,    ("Chennai", "Trichy"): 330,   ("Chennai", "Bangalore"): 350,
    ("Trichy", "Chennai"): 330,   ("Trichy", "Trichy"): 0,      ("Trichy", "Bangalore"): 420,
    ("Bangalore", "Chennai"): 350, ("Bangalore", "Trichy"): 420, ("Bangalore", "Bangalore"):
0,
}
available = []
for c in couriers:
    if c["assigned"] < max_capacity:
        available.append(c)
if available:
    best_index = 0
    for i in range(1, len(available)):
        curr_dist = distances[(available[i]["location"], pickup_location)]
        best_dist = distances[(available[best_index]["location"], pickup_location)]
        if curr_dist < best_dist:
            best_index = i
    selected = available[best_index]
    selected["assigned"] += 1
    print(f"Courier {selected['name']} assigned. Total assigned: {selected['assigned']}")
else:
    print("No available couriers.")
```



```
couriers = [{"name": "Courier A", "location": "Chennai", "assigned": 2},
    {"name": "Courier B", "location": "Trichy", "assigned": 5},
    {"name": "Courier C", "location": "Bangalore", "assigned": 4},]
max_capacity = 5
pickup_location = input("Enter pickup location (Chennai/Trichy/Bangalore): ")
distances = {
    ("Chennai", "Chennai"): 0,    ("Chennai", "Trichy"): 330,   ("Chennai", "Bangalore"): 350,
    ("Trichy", "Chennai"): 330,   ("Trichy", "Trichy"): 0,       ("Trichy", "Bangalore"): 420,
    ("Bangalore", "Chennai"): 350, ("Bangalore", "Trichy"): 420, ("Bangalore", "Bangalore"): 0,
}
available = []
for c in couriers:
    if c["assigned"] < max_capacity:
        available.append(c)
if available:
    best_index = 0
    for i in range(1, len(available)):
        curr_dist = distances[(available[i]["location"], pickup_location)]
```

PROBLEMS 16   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   SPELL CHECKER 16

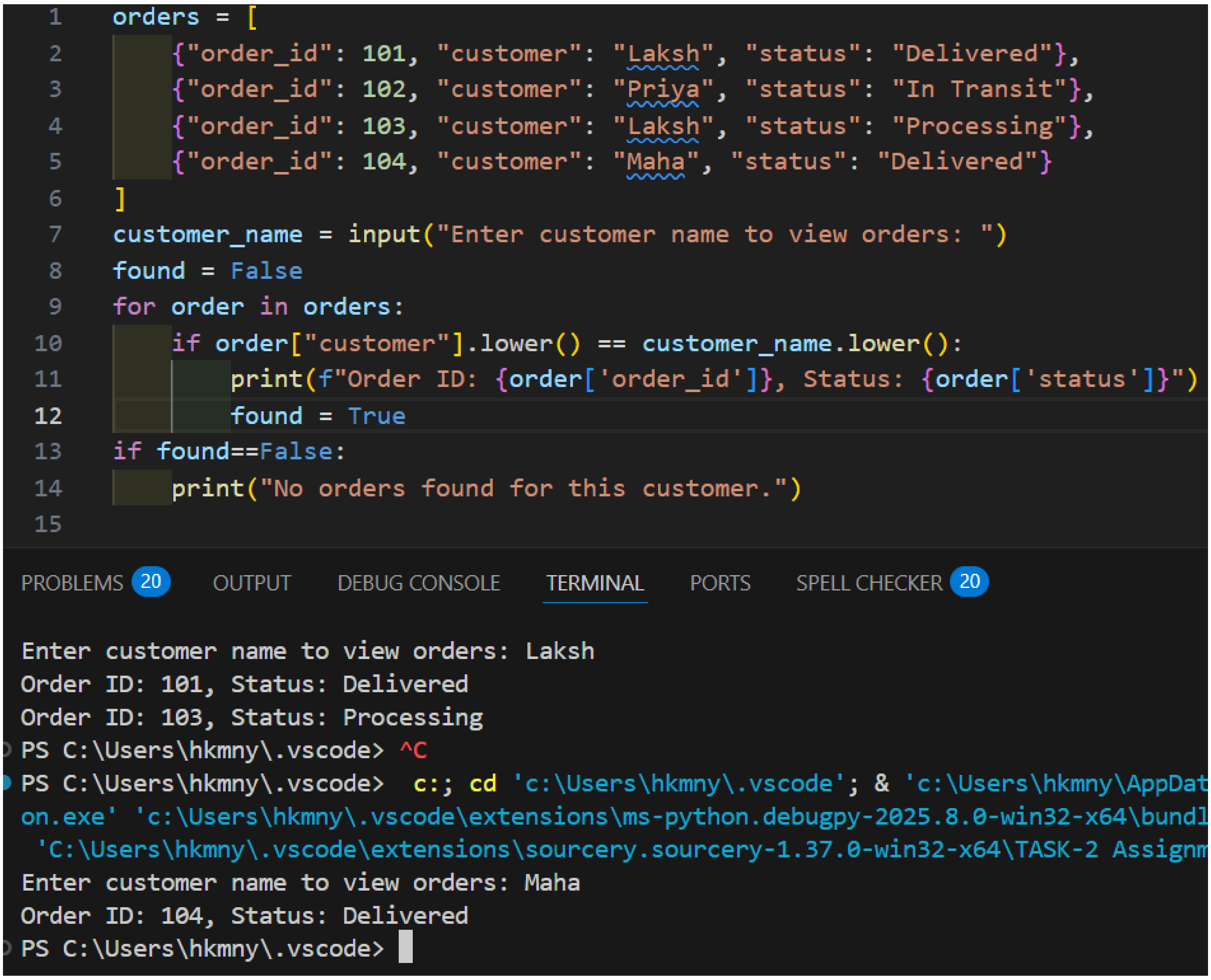
```
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '65279' '--'
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TASK-1 assignment'
Enter pickup location (Chennai/Trichy/Bangalore): Trichy
Courier Courier A assigned. Total assigned: 3
PS C:\Users\hkmny\.vscode>  c:; cd 'c:\Users\hkmny\.vscode'; & 'c:\Users\hkmny\AppData\Local\Programs\Python\Python313\pyth
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '65295' '--'
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TASK-1 assignment'
Enter pickup location (Chennai/Trichy/Bangalore): Bangalore
Courier Courier C assigned. Total assigned: 5
```

Task 2: Loops and Iteration

5. Write a Java program that uses a for loop to display all the orders for a specific customer.

Code:

```
orders = [
    {"order_id": 101, "customer": "Laksh", "status": "Delivered"},
    {"order_id": 102, "customer": "Priya", "status": "In Transit"},
    {"order_id": 103, "customer": "Laksh", "status": "Processing"},
    {"order_id": 104, "customer": "Maha", "status": "Delivered"}
]
customer_name = input("Enter customer name to view orders: ")
found = False
for order in orders:
    if order["customer"].lower() == customer_name.lower():
        print(f"Order ID: {order['order_id']}, Status: {order['status']}")
        found = True
if found==False:
    print("No orders found for this customer.")
```



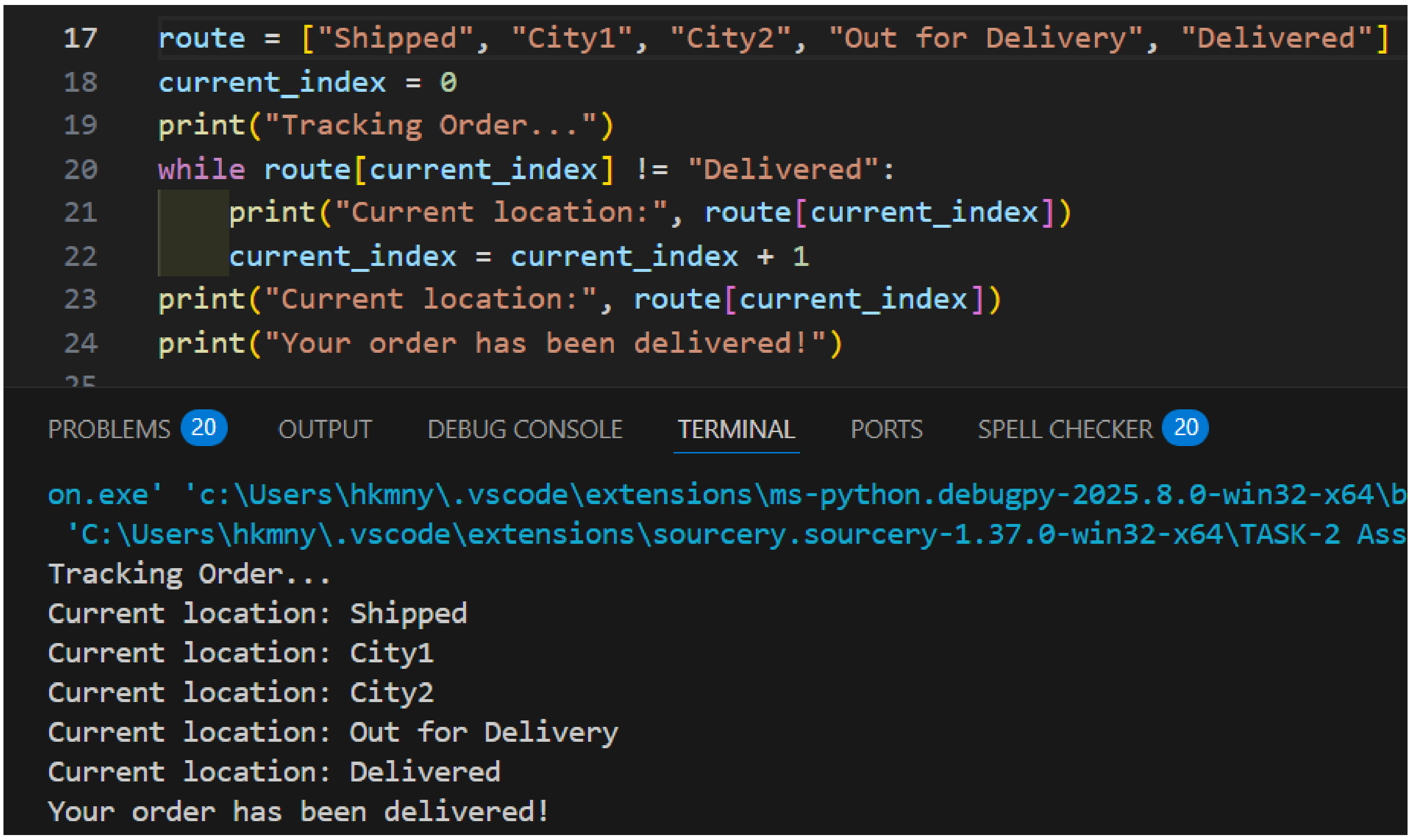
```python
orders = [
    {"order_id": 101, "customer": "Laksh", "status": "Delivered"},
    {"order_id": 102, "customer": "Priya", "status": "In Transit"},
    {"order_id": 103, "customer": "Laksh", "status": "Processing"},
    {"order_id": 104, "customer": "Maha", "status": "Delivered"}
]
customer_name = input("Enter customer name to view orders: ")
found = False
for order in orders:
    if order["customer"].lower() == customer_name.lower():
        print(f"Order ID: {order['order_id']}, Status: {order['status']}")
        found = True
if found==False:
    print("No orders found for this customer.")
```

```
Enter customer name to view orders: Laksh
Order ID: 101, Status: Delivered
Order ID: 103, Status: Processing
PS C:\Users\hkmny\.vscode> ^C
PS C:\Users\hkmny\.vscode>  c:; cd 'c:\Users\hkmny\.vscode'; & 'c:\Users\hkmny\AppDat
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundl
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TASK-2 Assignm
Enter customer name to view orders: Maha
Order ID: 104, Status: Delivered
PS C:\Users\hkmny\.vscode> 
```

6. Implement a while loop to track the real-time location of a courier until it reaches its destination.

Code:

```
route = ["Shipped", "City1", "City2", "Out for Delivery", "Delivered"]
current_index = 0
print("Tracking Order...")
while route[current_index] != "Delivered":
    print("Current location:", route[current_index])
    current_index = current_index + 1
print("Current location:", route[current_index])
print("Your order has been delivered!")
```



```
route = ["Shipped", "City1", "City2", "Out for Delivery", "Delivered"]
current_index = 0
print("Tracking Order...")
while route[current_index] != "Delivered":
    print("Current location:", route[current_index])
    current_index = current_index + 1
print("Current location:", route[current_index])
print("Your order has been delivered!")
```

```
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\b
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TASK-2 Ass
Tracking Order...
Current location: Shipped
Current location: City1
Current location: City2
Current location: Out for Delivery
Current location: Delivered
Your order has been delivered!
```

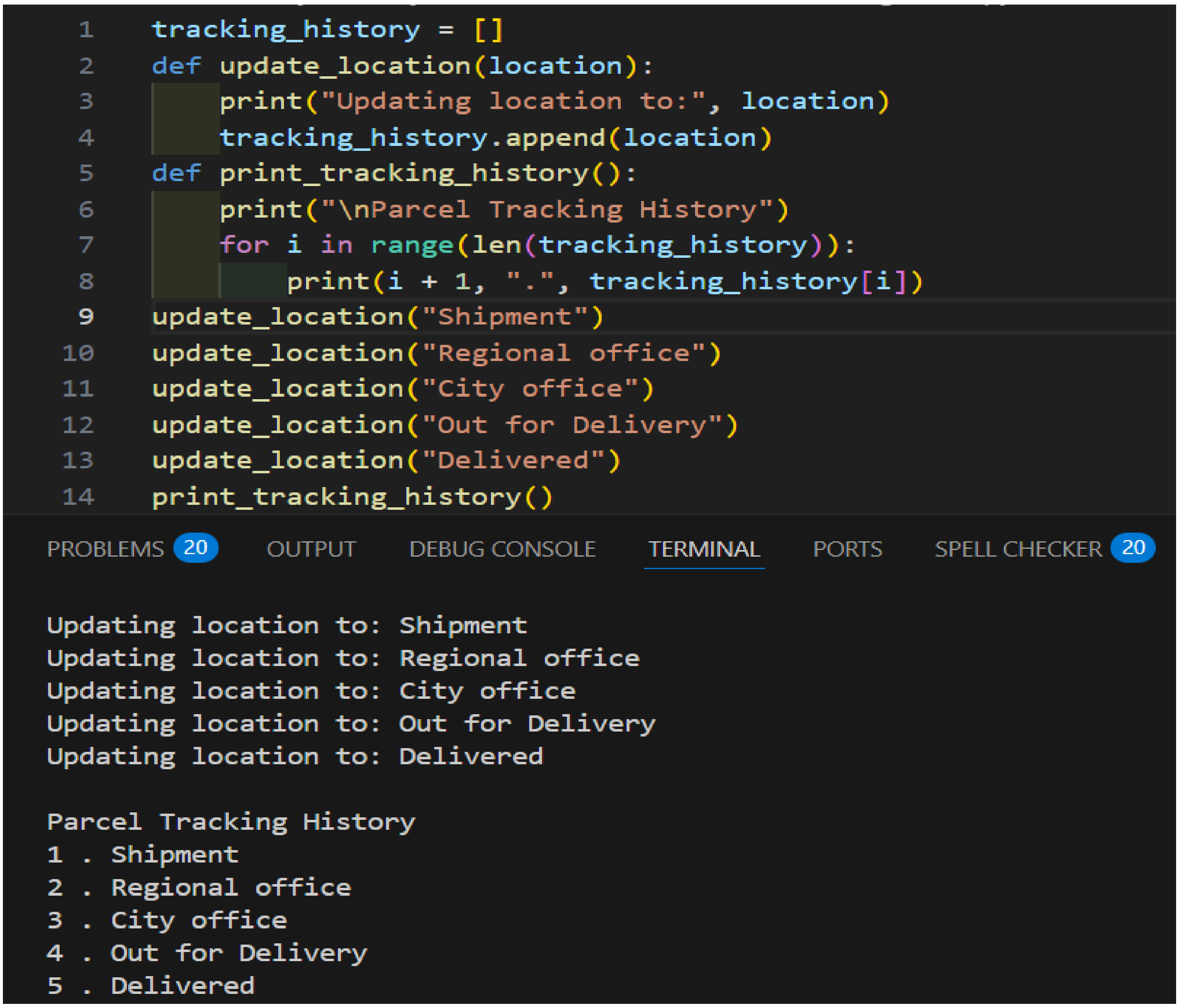
Task 3: Arrays and Data Structures

7. Create an array to store the tracking history of a parcel, where each entry represents a location update.

Code:

```
tracking_history = []
def update_location(location):
    print("Updating location to:", location)
    tracking_history.append(location)
```

```
def print_tracking_history():
    print("\nParcel Tracking History")
    for i in range(len(tracking_history)):
        print(i + 1, ".", tracking_history[i])
update_location("Shipment")
update_location("Regional office")
update_location("City office")
update_location("Out for Delivery")
update_location("Delivered")
print_tracking_history()
```



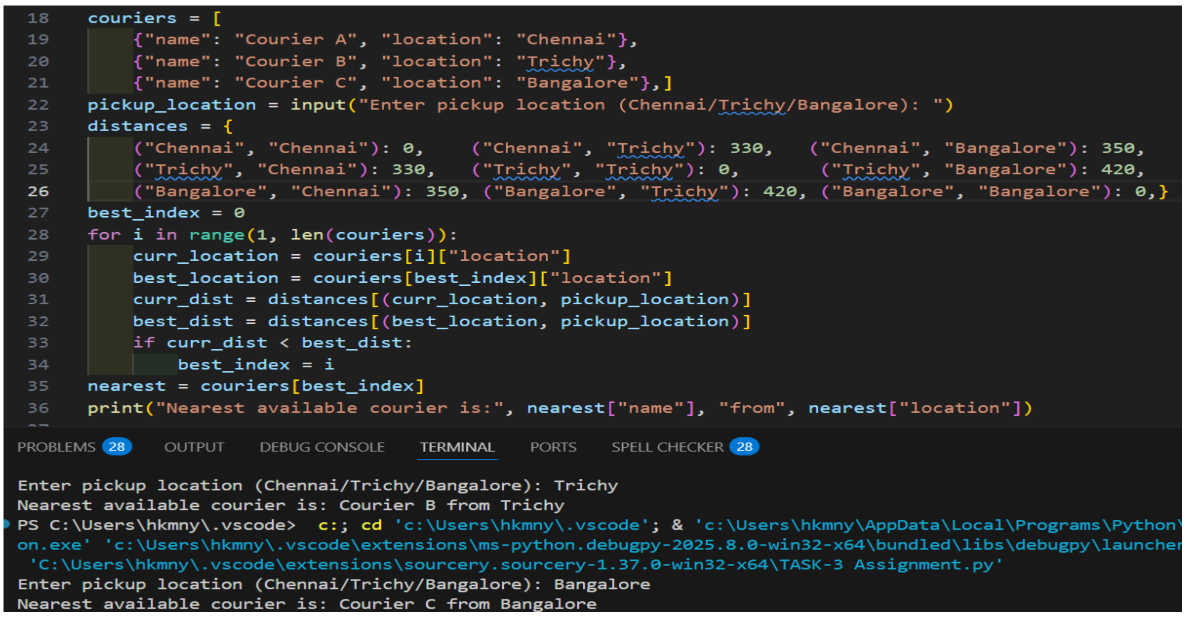
```
tracking_history = []
def update_location(location):
    print("Updating location to:", location)
    tracking_history.append(location)
def print_tracking_history():
    print("\nParcel Tracking History")
    for i in range(len(tracking_history)):
        print(i + 1, ".", tracking_history[i])
update_location("Shipment")
update_location("Regional office")
update_location("City office")
update_location("Out for Delivery")
update_location("Delivered")
print_tracking_history()
```

```
Updating location to: Shipment
Updating location to: Regional office
Updating location to: City office
Updating location to: Out for Delivery
Updating location to: Delivered

Parcel Tracking History
1 . Shipment
2 . Regional office
3 . City office
4 . Out for Delivery
5 . Delivered
```

8. Implement a method to find the nearest available courier for a new order using an array of couriers.

Code:

```
couriers = [
    {"name": "Courier A", "location": "Chennai"},
    {"name": "Courier B", "location": "Trichy"},
    {"name": "Courier C", "location": "Bangalore"},]
pickup_location = input ("Enter pickup location (Chennai/Trichy/Bangalore): ")
distances = {
    ("Chennai", "Chennai"): 0, ("Chennai", "Trichy"): 330, ("Chennai", "Bangalore"): 350,
    ("Trichy", "Chennai"): 330, ("Trichy", "Trichy"): 0, ("Trichy", "Bangalore"): 420,
    ("Bangalore", "Chennai"): 350, ("Bangalore", "Trichy"): 420, ("Bangalore", "Bangalore"):
0,}
best_index = 0
for i in range (1, len(couriers)):
    curr_location = couriers[i]["location"]
    best_location = couriers[best_index]["location"]
    curr_dist = distances [(curr_location, pickup_location)]
    best_dist = distances [(best_location, pickup_location)]
    if curr_dist < best_dist:
        best_index = i
nearest = couriers[best_index]
print ("Nearest available courier is:", nearest["name"], "from", nearest["location"])
```

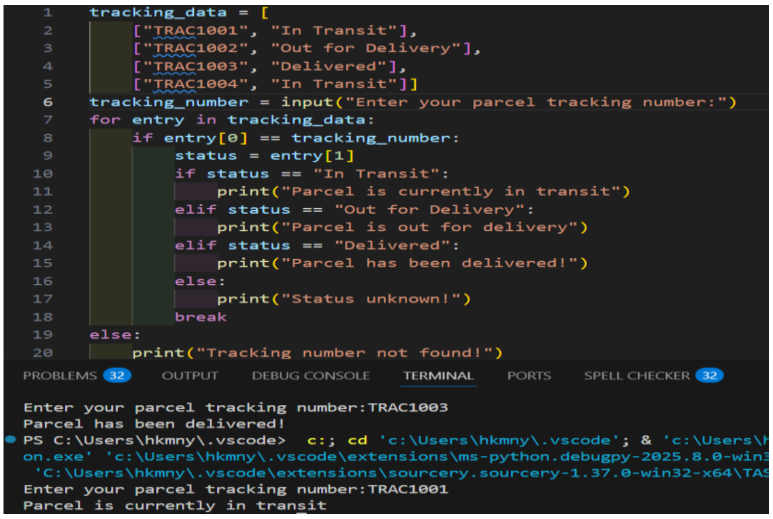


```
couriers = [
    {"name": "Courier A", "location": "Chennai"},
    {"name": "Courier B", "location": "Trichy"},
    {"name": "Courier C", "location": "Bangalore"},]
pickup_location = input("Enter pickup location (Chennai/Trichy/Bangalore): ")
distances = {
    ("Chennai", "Chennai"): 0,    ("Chennai", "Trichy"): 330,   ("Chennai", "Bangalore"): 350,
    ("Trichy", "Chennai"): 330,   ("Trichy", "Trichy"): 0,       ("Trichy", "Bangalore"): 420,
    ("Bangalore", "Chennai"): 350, ("Bangalore", "Trichy"): 420, ("Bangalore", "Bangalore"): 0,}
best_index = 0
for i in range(1, len(couriers)):
    curr_location = couriers[i]["location"]
    best_location = couriers[best_index]["location"]
    curr_dist = distances[(curr_location, pickup_location)]
    best_dist = distances[(best_location, pickup_location)]
    if curr_dist < best_dist:
        best_index = i
nearest = couriers[best_index]
print("Nearest available courier is:", nearest["name"], "from", nearest["location"])
```

```
Enter pickup location (Chennai/Trichy/Bangalore): Trichy
Nearest available courier is: Courier B from Trichy
PS C:\Users\hkmny\.vscode>  c:; cd 'c:\Users\hkmny\.vscode'; & 'c:\Users\hkmny\AppData\Local\Programs\Python\
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TASK-3 Assignment.py'
Enter pickup location (Chennai/Trichy/Bangalore): Bangalore
Nearest available courier is: Courier C from Bangalore
```

Task 4: Strings,2d Arrays, user defined functions,Hashmap
9. Parcel Tracking: Create a program that allows users to input a parcel tracking number.Store the tracking number and Status in 2d String Array. Initialize the array with values. Then, simulate the tracking process by displaying messages like "Parcel in transit," "Parcel out for delivery," or "Parcel delivered" based on the tracking number's status.
Code:

```
tracking_data = [
    ["TRAC1001", "In Transit"],
    ["TRAC1002", "Out for Delivery"],
    ["TRAC1003", "Delivered"],
    ["TRAC1004", "In Transit"]]
tracking_number = input("Enter your parcel tracking number:")
for entry in tracking_data:
    if entry[0] == tracking_number:
        status = entry[1]
        if status == "In Transit":
            print("Parcel is currently in transit")
        elif status == "Out for Delivery":
            print("Parcel is out for delivery")
        elif status == "Delivered":
            print("Parcel has been delivered!")
        else:
            print("Status unknown!")
        break
else:
    print("Tracking number not found!")
```



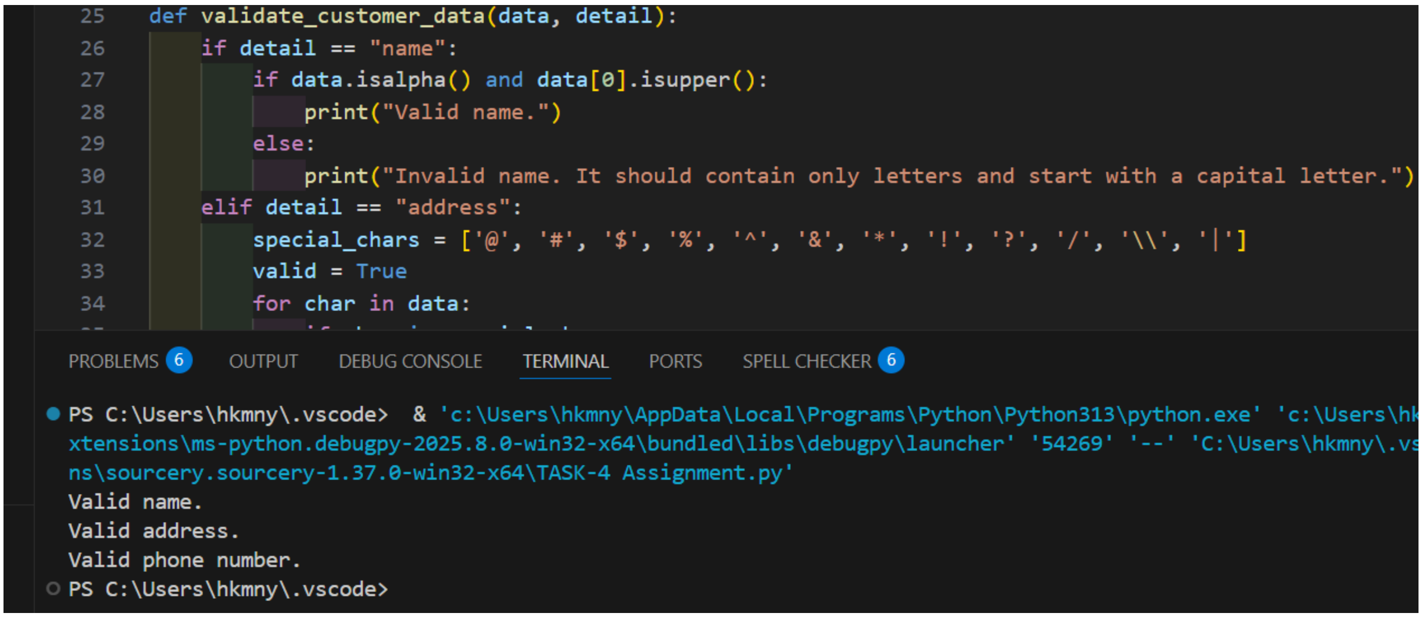
```
tracking_data = [
    ["TRAC1001", "In Transit"],
    ["TRAC1002", "Out for Delivery"],
    ["TRAC1003", "Delivered"],
    ["TRAC1004", "In Transit"]]
tracking_number = input("Enter your parcel tracking number:")
for entry in tracking_data:
    if entry[0] == tracking_number:
        status = entry[1]
        if status == "In Transit":
            print("Parcel is currently in transit")
        elif status == "Out for Delivery":
            print("Parcel is out for delivery")
        elif status == "Delivered":
            print("Parcel has been delivered!")
        else:
            print("Status unknown!")
        break
else:
    print("Tracking number not found!")
```

```
Enter your parcel tracking number:TRAC1003
Parcel has been delivered!
PS C:\Users\hkmny\.vscode>  c:; cd 'c:\Users\hkmny\.vscode'; & 'c:\Users\h
on.exe' 'c:\Users\hkmny\.vscode\extensions\ms-python.debugpy-2025.8.0-win3
 'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TAS
Enter your parcel tracking number:TRAC1001
Parcel is currently in transit
```

10. Customer Data Validation: Write a function which takes 2 parameters, data-denotes the data and detail-denotes if it is name addtress or phone number.Validate customer information based on following critirea. Ensure that names contain only letters and are properly capitalized, addresses do not contain special characters, and phone numbers follow a specific format (e.g., ###-###-####).

Code:

```
def validate_customer_data(data, detail):
    if detail == "name":
        if data.isalpha() and data[0].isupper():
            print("Valid name.")
        else:
            print("Invalid name. It should contain only letters and start with a capital letter.")
    elif detail == "address":
        special_chars = ['@', '#', '$', '%', '^', '&', '*', '!', '?', '/', '\\', '|']
        valid = True
        for char in data:
            if char in special_chars:
                valid = False
                break
        if valid:
            print("Valid address.")
        else:
            print("Invalid address. It should not contain special characters.")
    elif detail == "phone":
        if len(data) == 12 and data[3] == '-' and data[7] == '-' and \
           data[0:3].isdigit() and data[4:7].isdigit() and data[8:12].isdigit():
            print("Valid phone number.")
        else:
            print("Invalid phone number. Format must be ###-###-####")
    else:
        print("Invalid detail type. Use 'name', 'address', or 'phone'.")
validate_customer_data("Laksh", "name")
validate_customer_data("320 KK Nagar", "address")
validate_customer_data("987-654-3210", "phone")
```



```python
def validate_customer_data(data, detail):
    if detail == "name":
        if data.isalpha() and data[0].isupper():
            print("Valid name.")
        else:
            print("Invalid name. It should contain only letters and start with a capital letter.")
    elif detail == "address":
        special_chars = ['@', '#', '$', '%', '^', '&', '*', '!', '?', '/', '\\', '|']
        valid = True
        for char in data:
```

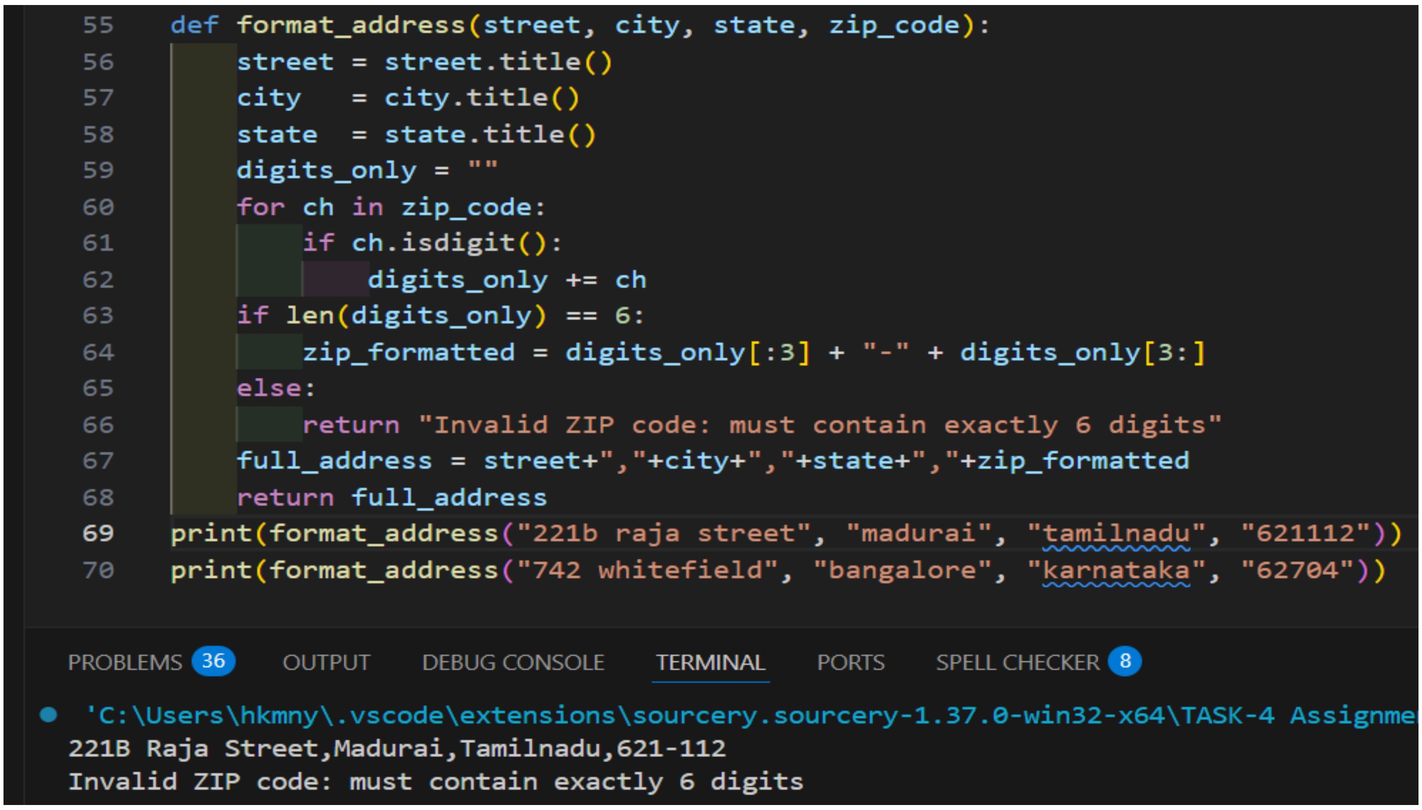
PROBLEMS 6   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   SPELL CHECKER 6

```
PS C:\Users\hkmny\.vscode>  & 'c:\Users\hkmny\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\hk
xtensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '54269' '--' 'C:\Users\hkmny\.vs
ns\sourcery.sourcery-1.37.0-win32-x64\TASK-4 Assignment.py'
Valid name.
Valid address.
Valid phone number.
PS C:\Users\hkmny\.vscode>
```

11. Address Formatting: Develop a function that takes an address as input (street, city, state, zip code) and formats it correctly, including capitalizing the first letter of each word and properly formatting the zip code.

Code:

```python
def format_address(street, city, state, zip_code):
    street = street.title()
    city   = city.title()
    state  = state.title()
    digits_only = ""
    for ch in zip_code:
        if ch.isdigit():
            digits_only += ch
    if len(digits_only) == 6:
        zip_formatted = digits_only[:3] + "-" + digits_only[3:]
    else:
        return "Invalid ZIP code: must contain exactly 6 digits"
    full_address = street+","+city+","+state+","+zip_formatted
    return full_address
print(format_address("221b raja street", "madurai", "tamilnadu", "621112"))
print(format_address("742 whitefield", "bangalore", "karnataka", "62704"))
```



```
def format_address(street, city, state, zip_code):
    street = street.title()
    city   = city.title()
    state  = state.title()
    digits_only = ""
    for ch in zip_code:
        if ch.isdigit():
            digits_only += ch
    if len(digits_only) == 6:
        zip_formatted = digits_only[:3] + "-" + digits_only[3:]
    else:
        return "Invalid ZIP code: must contain exactly 6 digits"
    full_address = street+","+city+","+state+","+zip_formatted
    return full_address
print(format_address("221b raja street", "madurai", "tamilnadu", "621112"))
print(format_address("742 whitefield", "bangalore", "karnataka", "62704"))
```

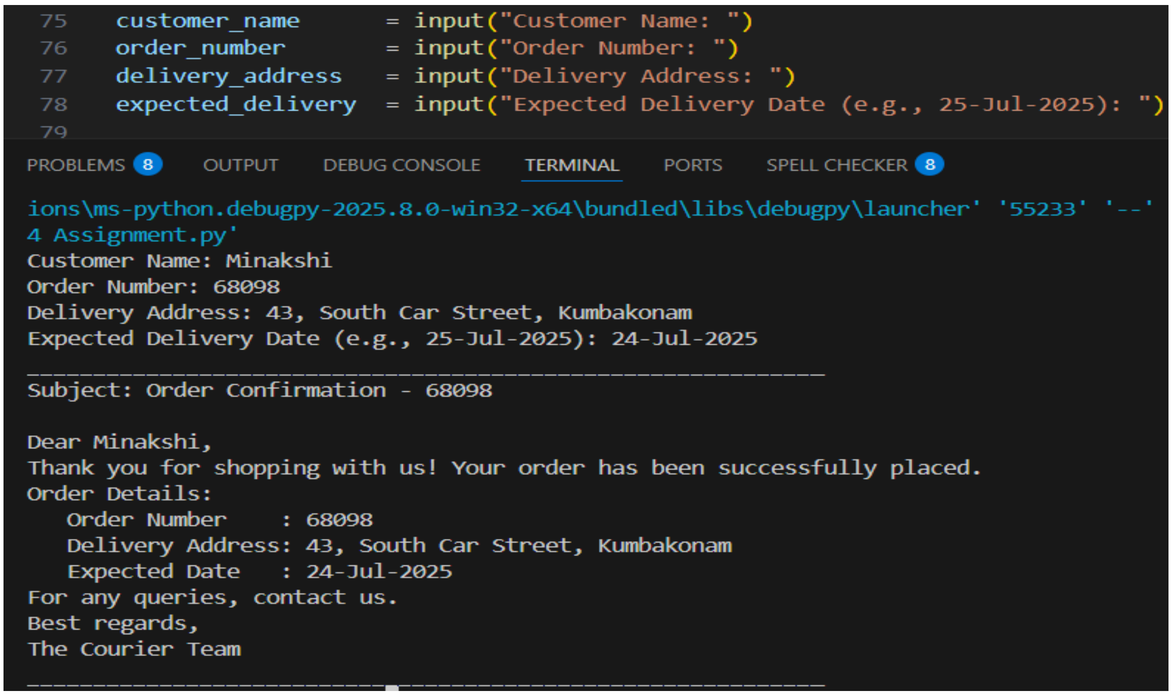
PROBLEMS 36   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   SPELL CHECKER 8

```
'C:\Users\hkmny\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\TASK-4 Assignme
221B Raja Street,Madurai,Tamilnadu,621-112
Invalid ZIP code: must contain exactly 6 digits
```

12. Order Confirmation Email: Create a program that generates an order confirmation email. The email should include details such as the customer's name, order number, delivery address, and expected delivery date.

Code:

```
customer_name     = input("Customer Name: ")
order_number      = input("Order Number: ")
delivery_address  = input("Delivery Address: ")
expected_delivery = input("Expected Delivery Date (e.g., 25-Jul-2025): ")

email_body = (
    "Subject: Order Confirmation - " + order_number + "\n"
    "\nDear " + customer_name + ",\n"
    "Thank you for shopping with us! Your order has been successfully placed.\n"
    "Order Details:\n"
    "   Order Number    : " + order_number + "\n"
    "   Delivery Address: " + delivery_address + "\n"
    "   Expected Date   : " + expected_delivery + "\n"
    "For any queries, contact us.\n"
    "Best regards,\n"
    "The Courier Team"
)
print("_"*60)
print(email_body)
print("_"*60)
```



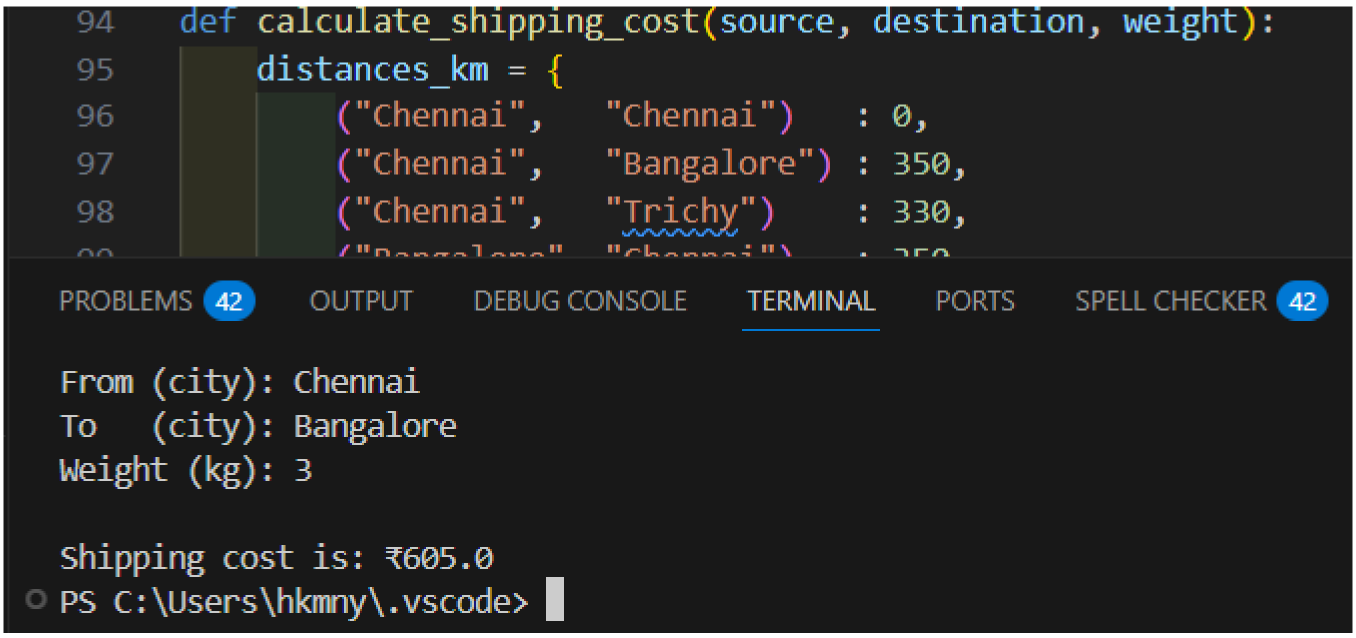
```
customer_name       = input("Customer Name: ")
order_number        = input("Order Number: ")
delivery_address    = input("Delivery Address: ")
expected_delivery   = input("Expected Delivery Date (e.g., 25-Jul-2025): ")
```

```
ions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '55233' '--'
4 Assignment.py'
Customer Name: Minakshi
Order Number: 68098
Delivery Address: 43, South Car Street, Kumbakonam
Expected Delivery Date (e.g., 25-Jul-2025): 24-Jul-2025

____________________________________________________________
Subject: Order Confirmation - 68098

Dear Minakshi,
Thank you for shopping with us! Your order has been successfully placed.
Order Details:
   Order Number    : 68098
   Delivery Address: 43, South Car Street, Kumbakonam
   Expected Date   : 24-Jul-2025
For any queries, contact us.
Best regards,
The Courier Team

____________________________________________________________
```

13. Calculate Shipping Costs: Develop a function that calculates the shipping cost based on the distance between two locations and the weight of the parcel. You can use string inputs for the source and destination addresses.

Code:

```
def calculate_shipping_cost(source, destination, weight):
    distances_km = {
        ("Chennai",   "Chennai")   : 0,
        ("Chennai",   "Bangalore") : 350,
        ("Chennai",   "Trichy")    : 330,
        ("Bangalore", "Chennai")   : 350,
        ("Bangalore", "Trichy")    : 420,
        ("Bangalore", "Bangalore") : 0,
        ("Trichy",    "Chennai")   : 330,
        ("Trichy",    "Bangalore") : 420,
        ("Trichy",    "Trichy")    : 0
    }
    key = (source, destination)
    if key not in distances_km:
        return "Sorry,Delivery not available for this Location."
    distance = distances_km[key]
    base_fee    = 50
    per_km_rate = 1.5
    per_kg_rate = 10
    cost = base_fee + (distance * per_km_rate) + (weight * per_kg_rate)
    return round(cost, 2)
src  = input("From (city): ")
dest = input("To   (city): ")
wt   = float(input("Weight (kg): "))
result = calculate_shipping_cost(src, dest, wt)
print()
if isinstance(result, str):
    print(result)
else:
    print("Shipping cost is: ₹", result, sep="")
```



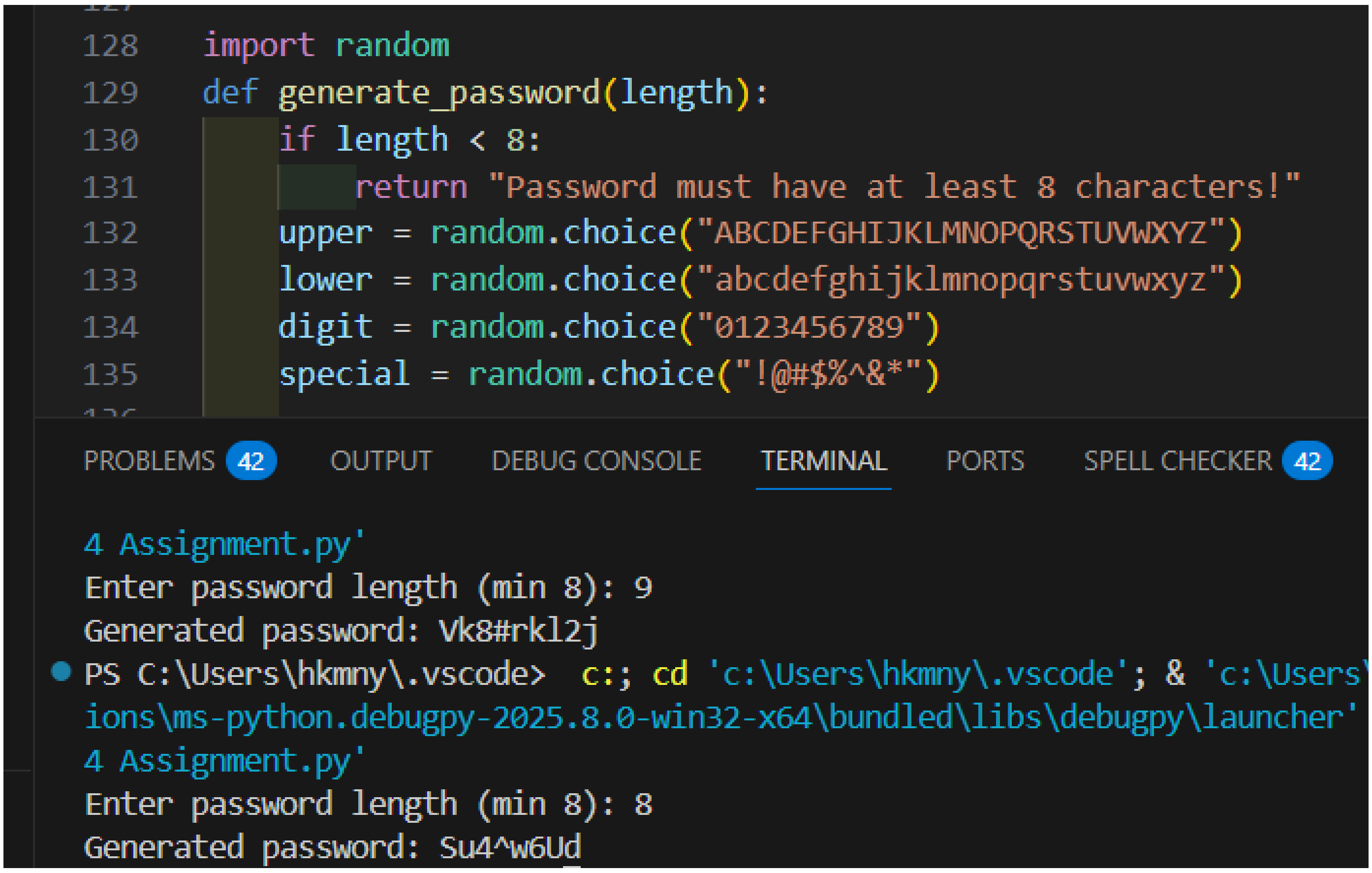
```
def calculate_shipping_cost(source, destination, weight):
    distances_km = {
        ("Chennai",    "Chennai")   : 0,
        ("Chennai",    "Bangalore") : 350,
        ("Chennai",    "Trichy")    : 330,
```

```
From (city): Chennai
To   (city): Bangalore
Weight (kg): 3

Shipping cost is: ₹605.0
PS C:\Users\hkmny\.vscode>
```

14. Password Generator: Create a function that generates secure passwords for courier system accounts. Ensure the passwords contain a mix of uppercase letters, lowercase letters, numbers, and special characters.

Code:

```
import random
def generate_password(length):
    if length < 8:
        return "Password must have at least 8 characters!"
    upper = random.choice("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
    lower = random.choice("abcdefghijklmnopqrstuvwxyz")
    digit = random.choice("0123456789")
    special = random.choice("!@#$%^&*")
    password = upper + lower + digit + special
    all_chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" +
"abcdefghijklmnopqrstuvwxyz"+ "0123456789" + "!@#$%^&*"
    for _ in range(length-4):
        password += random.choice(all_chars)
    return password

length = int(input("Enter password length (min 8): "))
print("Generated password:", generate_password(length))
```



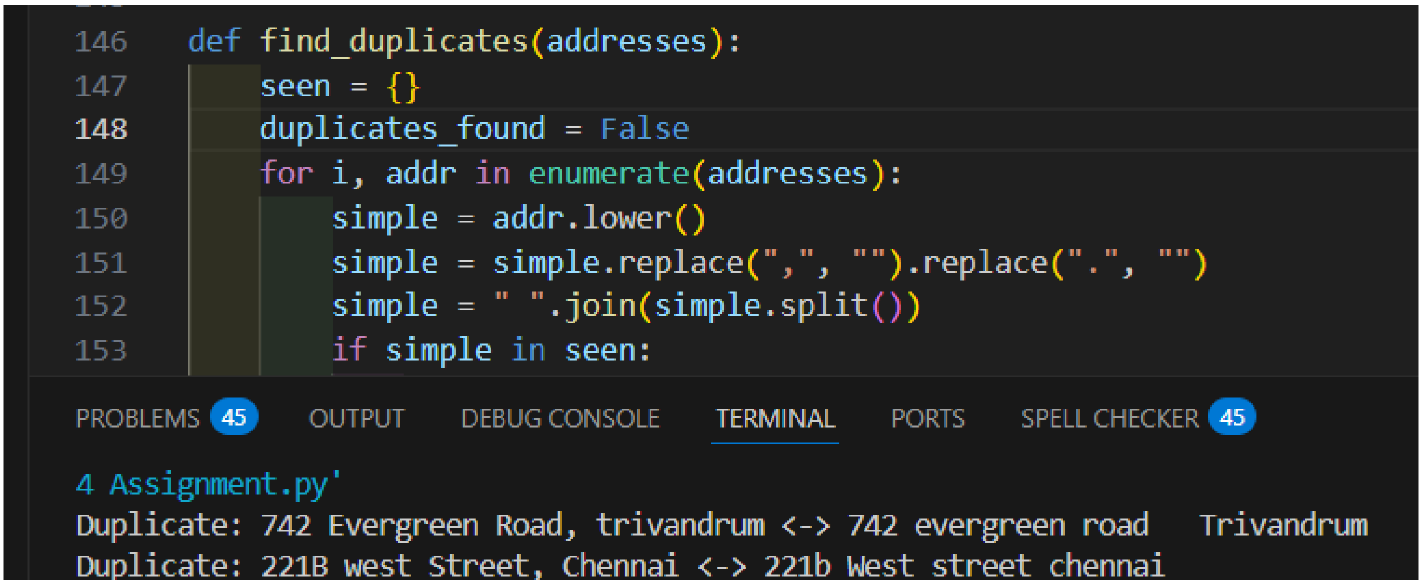
```
import random
def generate_password(length):
    if length < 8:
        return "Password must have at least 8 characters!"
    upper = random.choice("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
    lower = random.choice("abcdefghijklmnopqrstuvwxyz")
    digit = random.choice("0123456789")
    special = random.choice("!@#$%^&*")
```

```
4 Assignment.py'
Enter password length (min 8): 9
Generated password: Vk8#rkl2j
PS C:\Users\hkmny\.vscode>  c:; cd 'c:\Users\hkmny\.vscode'; & 'c:\Users
ions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher'
4 Assignment.py'
Enter password length (min 8): 8
Generated password: Su4^w6Ud
```

15. Find Similar Addresses: Implement a function that finds similar addresses in the system. This can be useful for identifying duplicate customer entries or optimizing delivery routes.Use string functions to implement this.

Code:

```
def find_duplicates(addresses):
    seen = {}
    duplicates_found = False
    for i, addr in enumerate(addresses):
        simple = addr.lower()
        simple = simple.replace(",", "").replace(".", "")
        simple = " ".join(simple.split())
        if simple in seen:
            j = seen[simple]
            print("Duplicate:", addresses[j], "<->", addresses[i])
            duplicates_found = True
        else:
            seen[simple] = i
    if not duplicates_found:
        print("No duplicate addresses found.")
addrs = [
    "742 Evergreen Road, trivandrum",
    "742 evergreen road   Trivandrum",
    "221B west Street, Chennai",
    "221b West street chennai",
    "1600 Kanyakumari"
]
find_duplicates(addrs)
```



```
def find_duplicates(addresses):
    seen = {}
    duplicates_found = False
    for i, addr in enumerate(addresses):
        simple = addr.lower()
        simple = simple.replace(",", "").replace(".", "")
        simple = " ".join(simple.split())
        if simple in seen:
```

```
4 Assignment.py'
Duplicate: 742 Evergreen Road, trivandrum <-> 742 evergreen road   Trivandrum
Duplicate: 221B west Street, Chennai <-> 221b West street chennai
```

TASK-5
Object Oriented Programming
Scope: Entity classes/Models/POJO, Abstraction/Encapsulation
Create the following model/entity classes within package entities with variables declared private, constructors (default and parametrized, getters, setters and toString())

1. **User Class:**
   Code:
```
class User:
    def __init__(self, user_id=None, user_name="", email="", password="",
                 contact_number="", address=""):
        self.__user_id = user_id
        self.__user_name = user_name
        self.__email = email
        self.__password = password
        self.__contact_number = contact_number
        self.__address = address
    def get_user_id(self):
        return self.__user_id
    def set_user_id(self, v):
        self.__user_id = v
    def __str__(self):
        return f"User[{self.__user_id}] {self.__user_name}"
```
2. **Courier Class:**
   Code:
```
import datetime
class Courier:
    __track_seed = 10000
    def __init__(self, courier_id=None, sender_name="", sender_address="",
                 receiver_name="", receiver_address="", weight=0.0,
                 status="YetToDispatch", delivery_date=None, user_id=None):
        self.__courier_id = courier_id
        self.__sender_name = sender_name
        self.__sender_address = sender_address
        self.__receiver_name = receiver_name
        self.__receiver_address = receiver_address
        self.__weight = weight
        self.__status = status
        self.__tracking_number = f"TRK{Courier.__track_seed}"
        Courier.__track_seed += 1
        self.__delivery_date = delivery_date or datetime.date.today()
        self.__user_id = user_id

    def get_tracking_number(self): return self.__tracking_number
    def get_status(self):          return self.__status
    def set_status(self, s):       self.__status = s
    def __str__(self):
        return f"Courier[{self.__courier_id}] {self.__tracking_number}"
```

3. **Employee Class:**

   Code:

```
class Employee:
    def __init__(self, employee_id=None, employee_name="", email="",
                 contact_number="", role="", salary=0.0):
        self.__employee_id = employee_id
        self.__employee_name = employee_name
        self.__email = email
        self.__contact_number = contact_number
        self.__role = role
        self.__salary = salary
    def __str__(self):
        return f"Emp[{self.__employee_id}] {self.__employee_name}"
```

4. **Location Class:**

   Code:

```
class Location:
    def __init__(self, location_id=None, location_name="", address=""):
        self.__location_id = location_id
        self.__location_name = location_name
        self.__address = address
    def __str__(self):
        return f"Loc[{self.__location_id}] {self.__location_name}"
```

5. **CourierCompany Class:**

   Code:

```
class CourierCompany:
    def __init__(self, company_name=""):
        self.__company_name = company_name
        self.__courier_details  = []
        self.__employee_details = []
        self.__location_details = []
    def add_courier(self, c):  self.__courier_details.append(c)
    def add_employee(self, e): self.__employee_details.append(e)
    def add_location(self, l): self.__location_details.append(l)
    def _couriers(self):  return self.__courier_details
    def _employees(self): return self.__employee_details
```

6. **Payment Class:**

   Code:

```
import datetime
class Payment:
    def __init__(self, payment_id=None, courier_id=None,
                amount=0.0, payment_date=None):
        self.__payment_id = payment_id
        self.__courier_id = courier_id
        self.__amount = amount
        self.__payment_date = payment_date or datetime.date.today()
    def __str__(self):
        return f"Pay[{self.__payment_id}] ₹{self.__amount}"
```

TASK-6

ICourierUserService

Code:

```
from abc import ABC, abstractmethod

class ICourierUserService(ABC):

    @abstractmethod

    def place_order(self, courier_obj): ...

    @abstractmethod

    def get_order_status(self, tracking_number): ...

    @abstractmethod

    def cancel_order(self, tracking_number): ...

    @abstractmethod

    def get_assigned_order(self, courier_staff_id): ...
```

ICourierAdminService

Code:

```
from abc import ABC, abstractmethod
class ICourierAdminService(ABC):
    @abstractmethod
    def add_courier_staff(self, employee_obj): ...
```

TASK 7:

Exception Handling

1. **TrackingNumberNotFoundException:**
   Code:
   ```
   class TrackingNumberNotFoundException(Exception):
       pass
   ```

2. **InvalidEmployeeIdException:**
   Code:
   ```
   class InvalidEmployeeIdException(Exception):
       pass
   ```

TASK-8:

Service implementation

CourierUserServiceImpl:

Code:

```
from service.courier_user_service import ICourierUserService
from exception.tracking_number_not_found import TrackingNumberNotFoundException


class CourierUserServiceImpl(ICourierUserService):
    def __init__(self, company_obj):
```

```
        self.company = company_obj

    def place_order(self, courier_obj):
        self.company.add_courier(courier_obj)
        return courier_obj.get_tracking_number()

    def get_order_status(self, tracking_number):
        for c in self.company._couriers():
            if c.get_tracking_number() == tracking_number:
                return c.get_status()
        raise TrackingNumberNotFoundException("Tracking number not found")

    def cancel_order(self, tracking_number):
        for c in self.company._couriers():
            if c.get_tracking_number() == tracking_number:
                c.set_status("Cancelled")
                return True
        return False

    def get_assigned_order(self, courier_staff_id):
        return self.company._couriers()
```

CourierAdminServiceImpl:

Code:

```
from service.courier_admin_service import ICourierAdminService
from service.courier_user_service_impl import CourierUserServiceImpl
class CourierAdminServiceImpl(CourierUserServiceImpl, ICourierAdminService):
    def add_courier_staff(self, employee_obj):
        self.company.add_employee(employee_obj)
        return employee_obj._Employee__employee_id
```

CourierAdminServiceCollectionImpl:

Code:

```
from service.courier_admin_service_impl import CourierAdminServiceImpl
class CourierAdminServiceCollectionImpl(CourierAdminServiceImpl):
    pass
```

CourierUserServiceColectionImpl:

Code:

```
from service.courier_user_service_impl import CourierUserServiceImpl
class CourierUserServiceCollectionImpl(CourierUserServiceImpl):
    pass
```

TASK-9

Database Interaction

1. **Write code to establish a connection to your SQL database.**
   Code:

```
import configparser, pathlib, mysql.connector, mysql.connector.errors

_prop = pathlib.Path(__file__).with_name("db.properties")
cfg   = configparser.ConfigParser()
cfg.read(_prop)

def get_connection():
    try:
        params = dict(cfg["mysql"])
        params["port"] = int(params.get("port", 3306))
        return mysql.connector.connect(**params)
    except mysql.connector.Error as e:
        print("DB connection failed:", e)
        raise
```

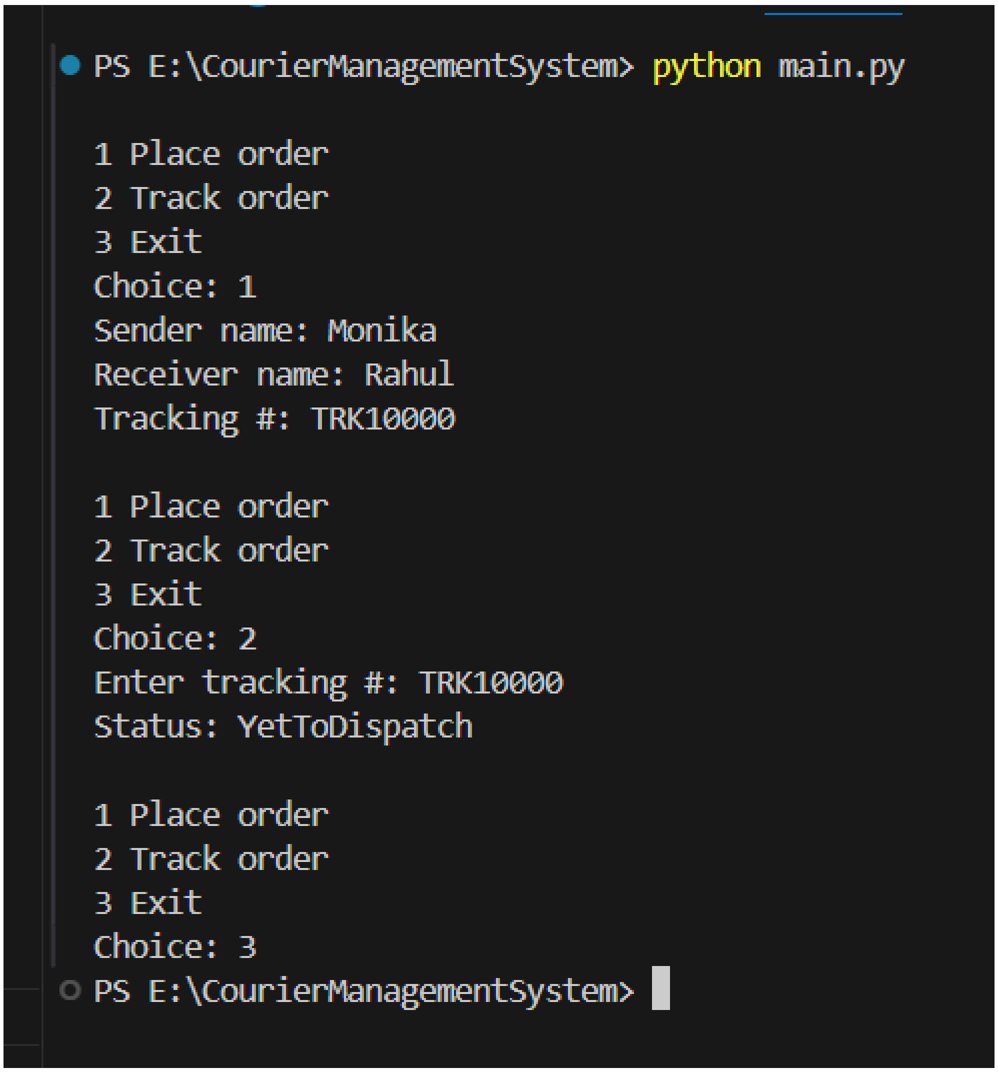
2. **Create a Service class CourierServiceDb in dao with a static variable named connection of type Connection which can be assigned in the constructor by invoking the method in DBConnection Class.**
   Code:

```
from db.db_connection import get_connection
from mysql.connector import Error

class CourierServiceDb:

    def __init__(self):
        self.cnx = get_connection()
        self._ensure_tables()

    def _ensure_tables(self):
        ddl = """
        CREATE TABLE IF NOT EXISTS courier(
            id INT AUTO_INCREMENT PRIMARY KEY,
            tracking   VARCHAR(20) UNIQUE,
            status     VARCHAR(50),
            user_id    INT,
            delivery_date DATE
        );
        CREATE TABLE IF NOT EXISTS payment(
            id INT AUTO_INCREMENT PRIMARY KEY,
            courier_id INT,
            amount DECIMAL(10,2),
            pay_date DATE
        );
        """
        cur = self.cnx.cursor()
        for stmt in ddl.strip().split(";"):
            if stmt.strip():
                cur.execute(stmt)
        self.cnx.commit()

    def insert_courier(self, tracking, status, user_id=None, date=None):
        sql = "INSERT INTO courier(tracking,status,user_id,delivery_date)
VALUES(%s,%s,%s,%s)"
        cur = self.cnx.cursor()
        cur.execute(sql, (tracking, status, user_id, date))
        self.cnx.commit()

    def update_status(self, tracking, new_status):
        cur = self.cnx.cursor()
        cur.execute("UPDATE courier SET status=%s WHERE tracking=%s",
                    (new_status, tracking))
        self.cnx.commit()

    def get_status(self, tracking):
        cur = self.cnx.cursor()
        cur.execute("SELECT status FROM courier WHERE tracking=%s", (tracking,))
        row = cur.fetchone()
        return row[0] if row else None
```

The main code to run the application:

Code:

```
from entity.courier_company import CourierCompany

from entity.courier import Courier

from service.courier_user_service_impl import CourierUserServiceImpl

company = CourierCompany("FastShip")

user_service = CourierUserServiceImpl(company)

def menu():

    print("\n1 Place order\n2 Track order\n3 Exit")

    return input("Choice: ")

while True:

    choice = menu()
    if choice == "1":
        sender = input("Sender name: ")
        receiver = input("Receiver name: ")
        c = Courier(courier_id=len(company._couriers())+1,
                    sender_name=sender,
                    sender_address="Src",
                    receiver_name=receiver,
                    receiver_address="Dest",
                    weight=1.0)
        print("Tracking #:", user_service.place_order(c))
    elif choice == "2":
        tn = input("Enter tracking #: ")
        try:
            print("Status:", user_service.get_order_status(tn))
        except Exception as e:
            print(e)
    else:
        break
#python main.py
```



```
PS E:\CourierManagementSystem> python main.py

1 Place order
2 Track order
3 Exit
Choice: 1
Sender name: Monika
Receiver name: Rahul
Tracking #: TRK10000

1 Place order
2 Track order
3 Exit
Choice: 2
Enter tracking #: TRK10000
Status: YetToDispatch

1 Place order
2 Track order
3 Exit
Choice: 3
PS E:\CourierManagementSystem>
```