1. Password Generator

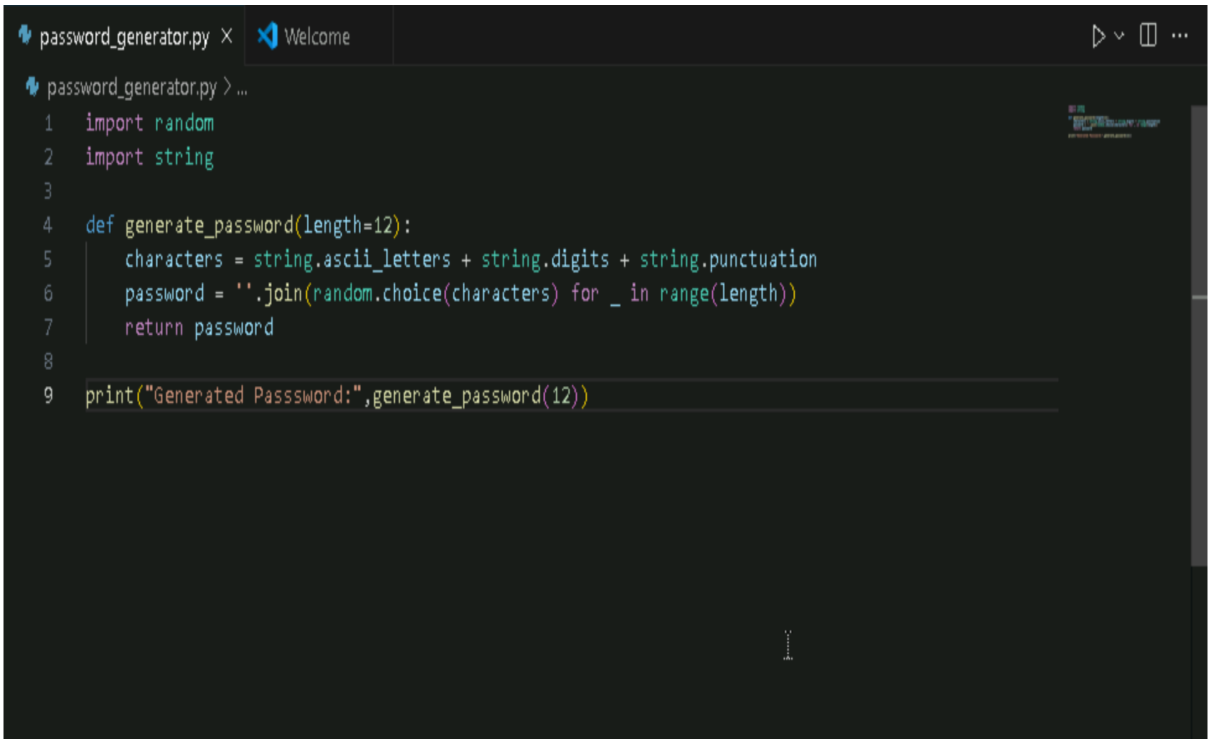
1. Import required libraries: random and string.

2. Define password length.

3. Create a character pool (uppercase, lowercase, digits, symbols).

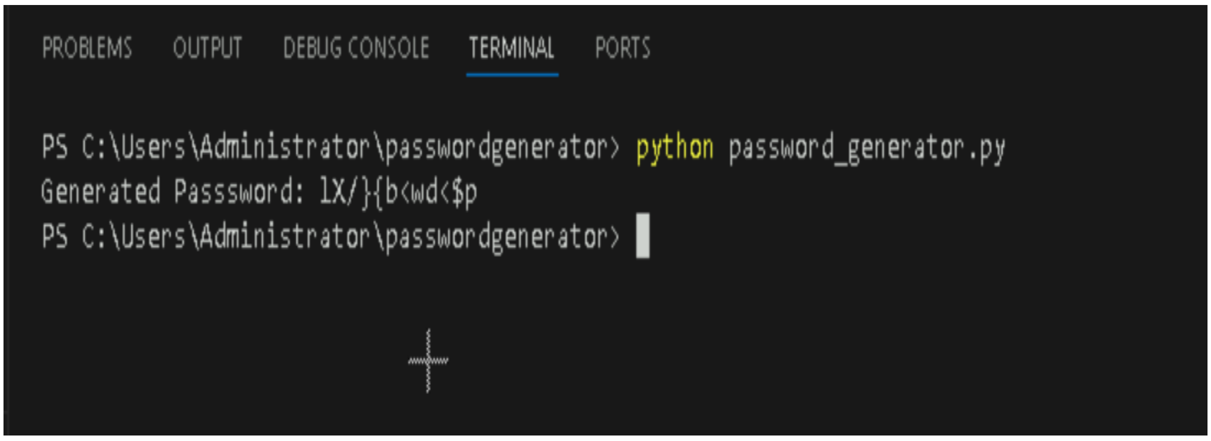
4. Randomly select characters from the pool.

5. Generate and display the password.



```python
import random
import string

def generate_password(length=12):
    characters = string.ascii_letters + string.digits + string.punctuation
    password = ''.join(random.choice(characters) for _ in range(length))
    return password

print("Generated Passsword:",generate_password(12))
```

Output:



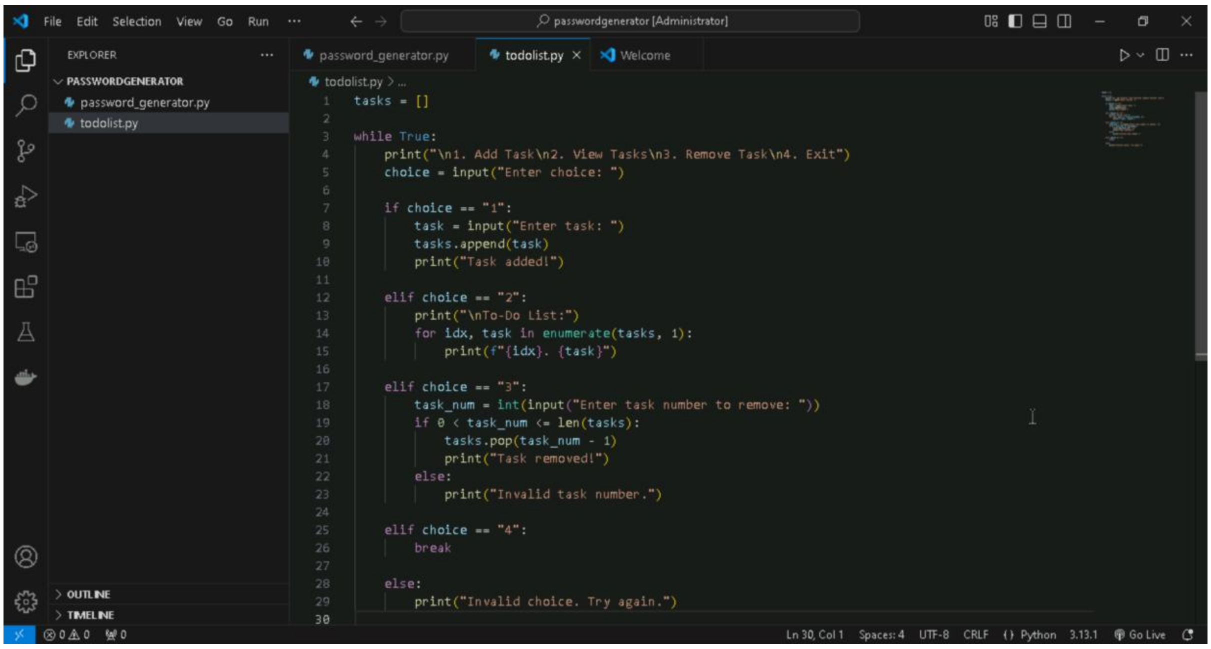
```
PS C:\Users\Administrator\passwordgenerator> python password_generator.py
Generated Passsword: lX/}{b<wd<$p
PS C:\Users\Administrator\passwordgenerator>
```

2. To-Do List (CLI)

1. Create a list to store tasks.

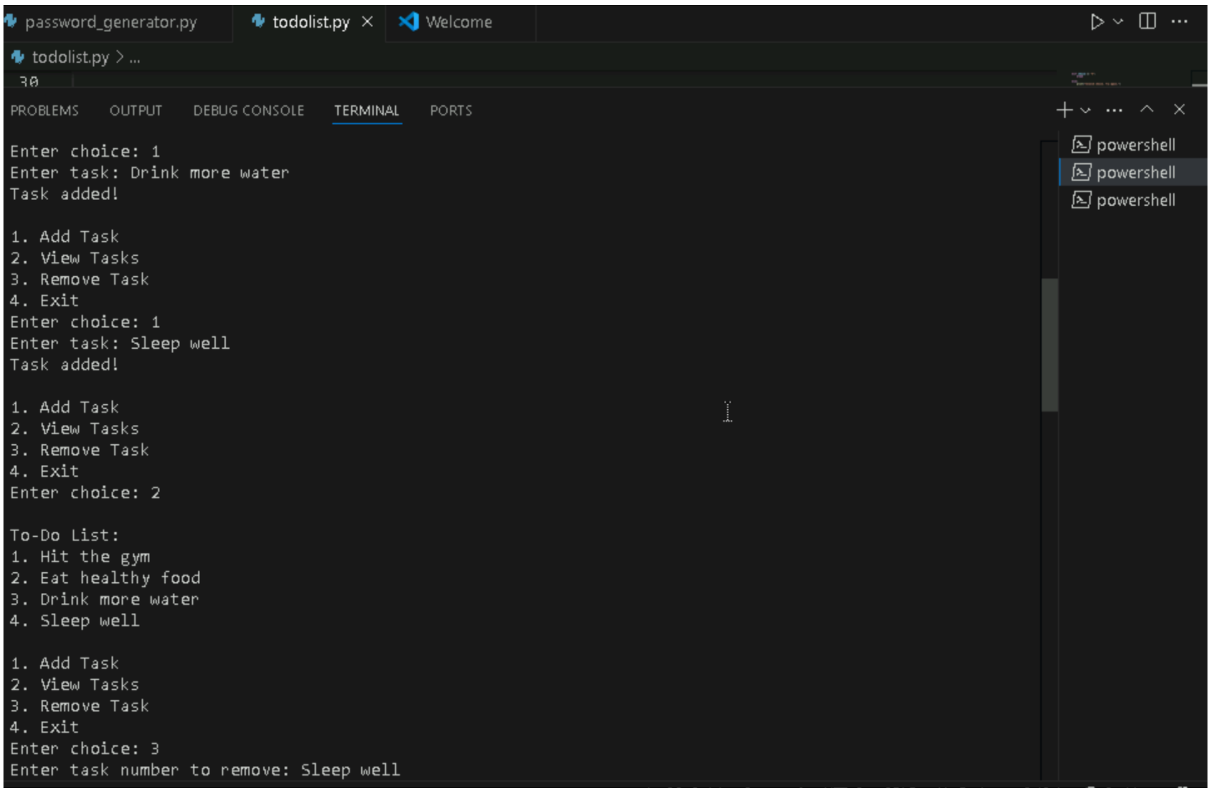
2. Provide options: Add, View, Remove, Exit.

3. Loop until the user exits.


File Edit Selection View Go Run
passwordgenerator [Administrator]
EXPLORER
PASSWORDGENERATOR
password_generator.py
todolist.py
Welcome
OUTLINE
TIMELINE
Ln 30, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.1 Go Live

```python
tasks = []

while True:
    print("\n1. Add Task\n2. View Tasks\n3. Remove Task\n4. Exit")
    choice = input("Enter choice: ")

    if choice == "1":
        task = input("Enter task: ")
        tasks.append(task)
        print("Task added!")

    elif choice == "2":
        print("\nTo-Do List:")
        for idx, task in enumerate(tasks, 1):
            print(f"{idx}. {task}")

    elif choice == "3":
        task_num = int(input("Enter task number to remove: "))
        if 0 < task_num <= len(tasks):
            tasks.pop(task_num - 1)
            print("Task removed!")
        else:
            print("Invalid task number.")

    elif choice == "4":
        break

    else:
        print("Invalid choice. Try again.")
```

Output:



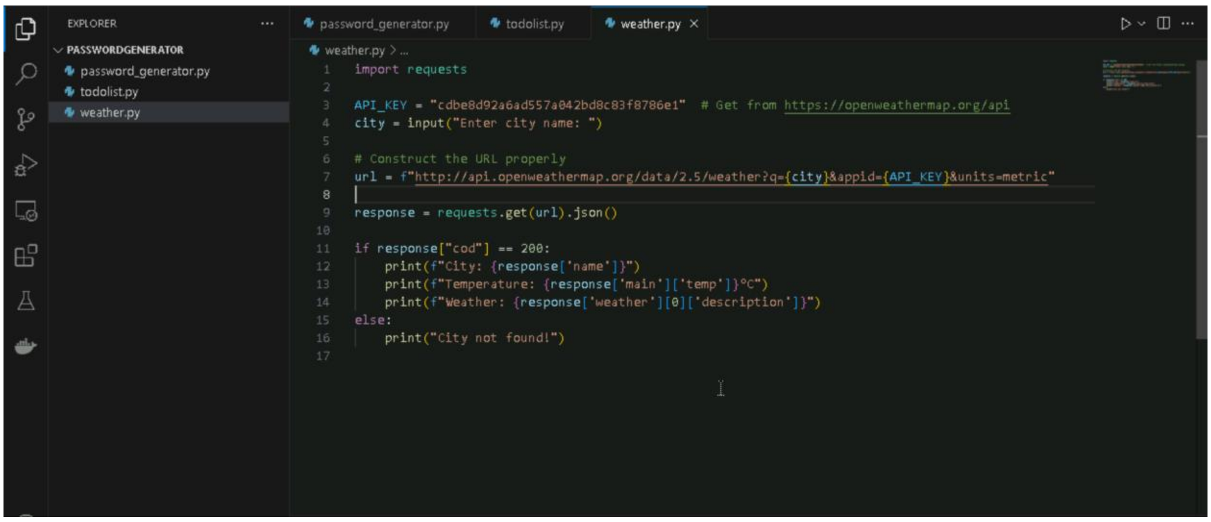
```
Enter choice: 1
Enter task: Drink more water
Task added!

1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Enter choice: 1
Enter task: Sleep well
Task added!

1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Enter choice: 2

To-Do List:
1. Hit the gym
2. Eat healthy food
3. Drink more water
4. Sleep well

1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Enter choice: 3
Enter task number to remove: Sleep well
```

3. Weather App (API-based)

1. Sign up for OpenWeatherMap API and get an API key.

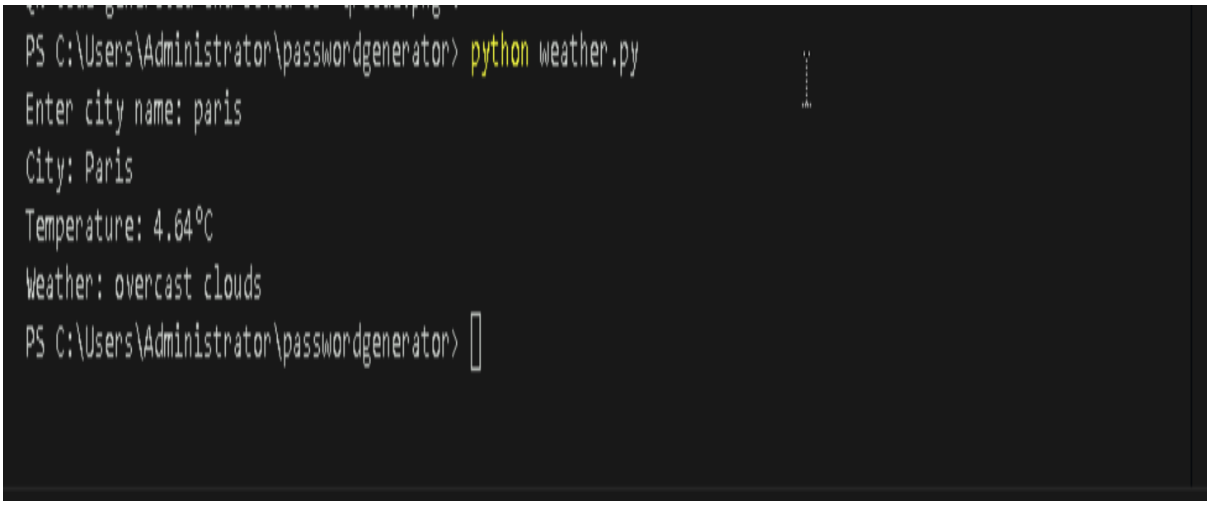
2. Use requests to fetch weather data.

3. Display temperature, weather condition, and city name.



```python
import requests

API_KEY = "cdbe8d92a6ad557a042bd8c83f8786e1"  # Get from https://openweathermap.org/api
city = input("Enter city name: ")

# Construct the URL properly
url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}&units=metric"

response = requests.get(url).json()

if response["cod"] == 200:
    print(f"City: {response['name']}")
    print(f"Temperature: {response['main']['temp']}°C")
    print(f"Weather: {response['weather'][0]['description']}")
else:
    print("City not found!")
```

Output:



```
PS C:\Users\Administrator\passwordgenerator> python weather.py
Enter city name: paris
City: Paris
Temperature: 4.64°C
Weather: overcast clouds
PS C:\Users\Administrator\passwordgenerator> 
```

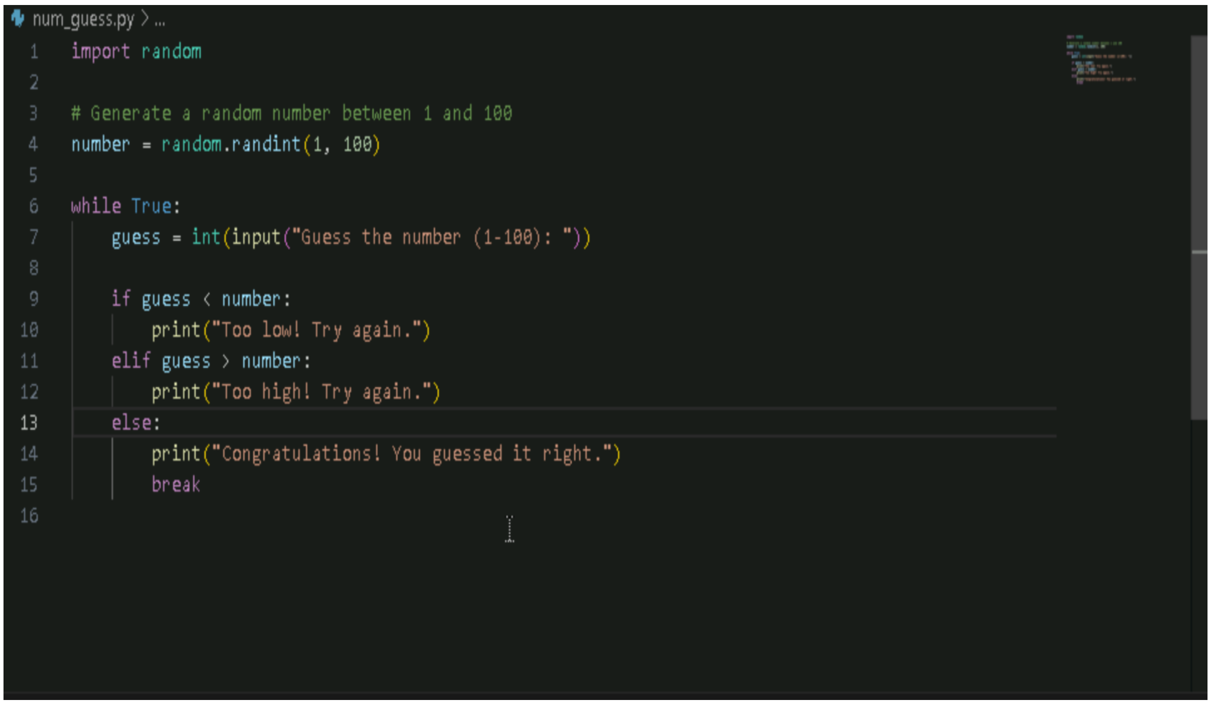
4. Number Guessing Game

1. Generate a random number between 1-100.

2. Ask the user to guess.

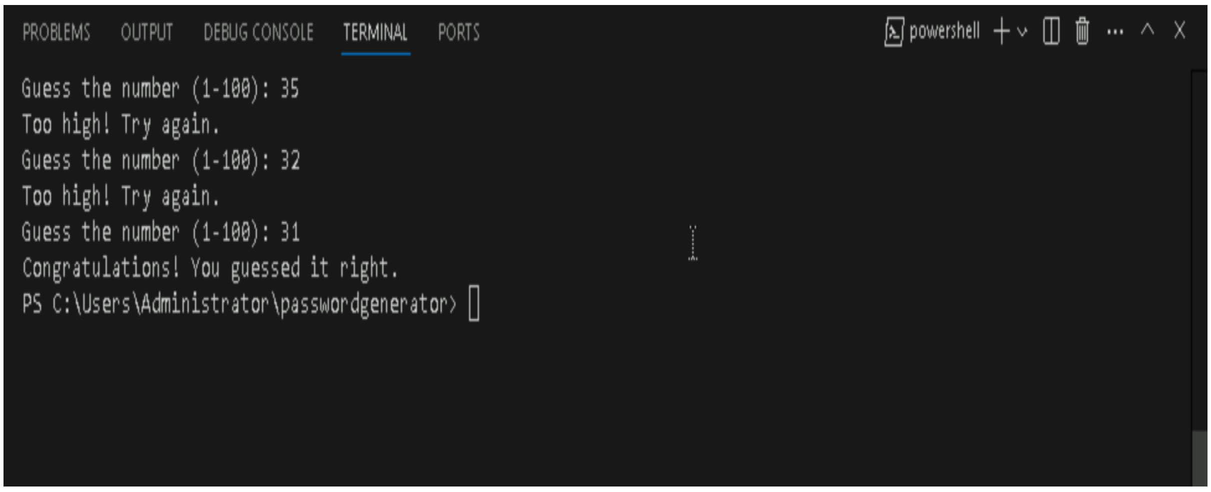
3. Give hints if the guess is too high/low.

4. Continue until guessed correctly.



```python
import random

# Generate a random number between 1 and 100
number = random.randint(1, 100)

while True:
    guess = int(input("Guess the number (1-100): "))

    if guess < number:
        print("Too low! Try again.")
    elif guess > number:
        print("Too high! Try again.")
    else:
        print("Congratulations! You guessed it right.")
        break

```

Output:



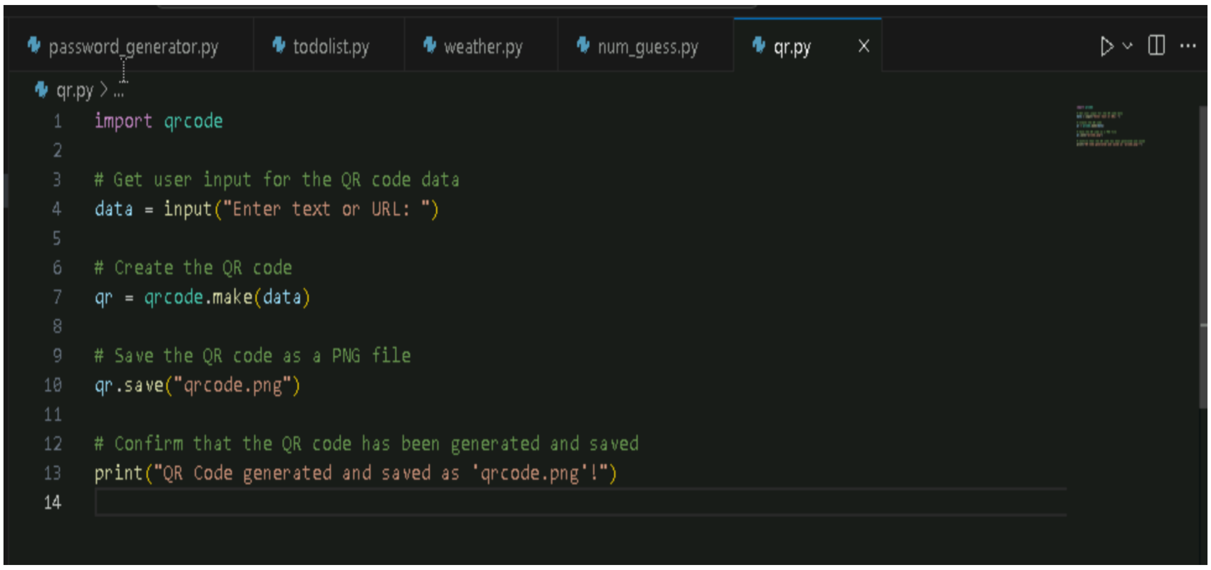
```
Guess the number (1-100): 35
Too high! Try again.
Guess the number (1-100): 32
Too high! Try again.
Guess the number (1-100): 31
Congratulations! You guessed it right.
PS C:\Users\Administrator\passwordgenerator>
```

5. QR Code Generator

1. Install qrcode library (pip install qrcode).

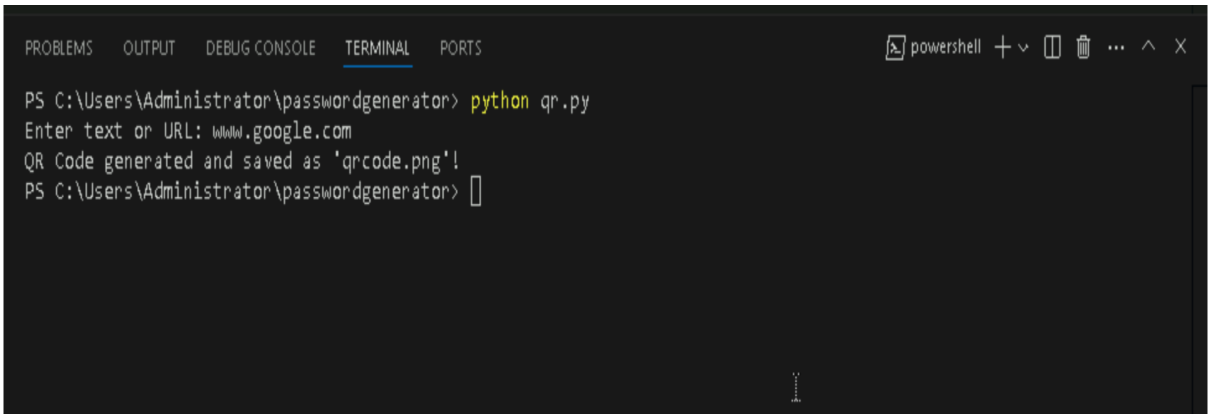
2. Take user input (text/link) to convert.

3. Generate and save the QR code.

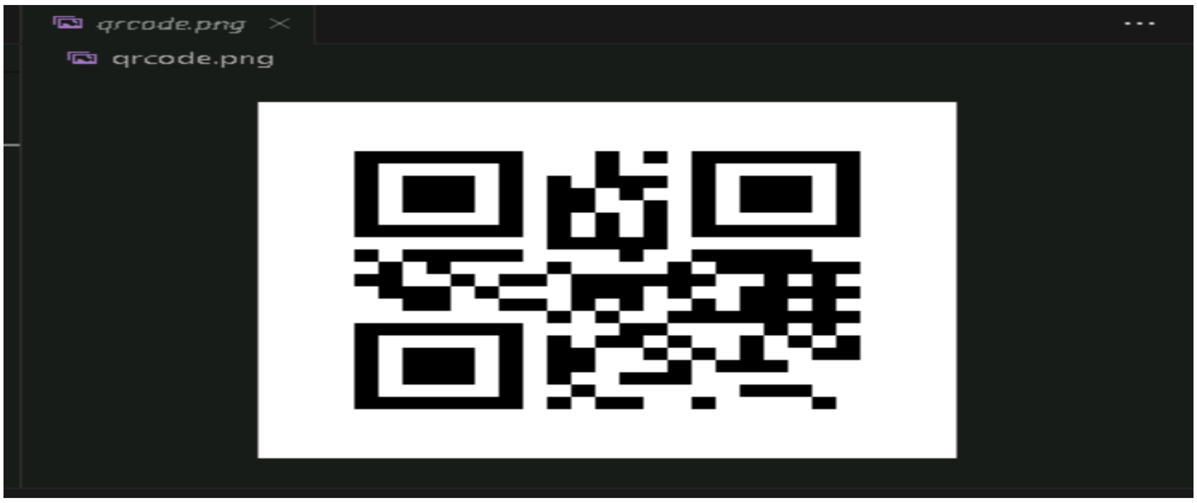


```python
import qrcode

# Get user input for the QR code data
data = input("Enter text or URL: ")

# Create the QR code
qr = qrcode.make(data)

# Save the QR code as a PNG file
qr.save("qrcode.png")

# Confirm that the QR code has been generated and saved
print("QR Code generated and saved as 'qrcode.png'!")
```

Output:



```
PS C:\Users\Administrator\passwordgenerator> python qr.py
Enter text or URL: www.google.com
QR Code generated and saved as 'qrcode.png'!
PS C:\Users\Administrator\passwordgenerator>
```

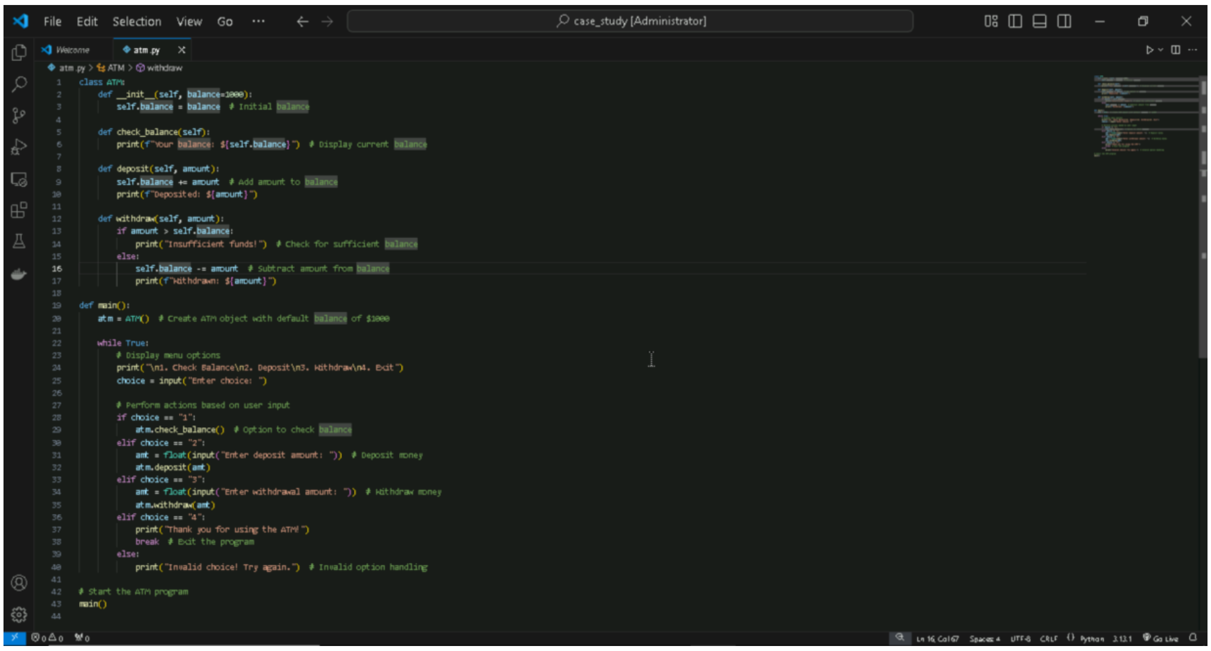


PYTHON CASE STUDIES WITH SOLUTIONS

1. Case Study: ATM Simulation System

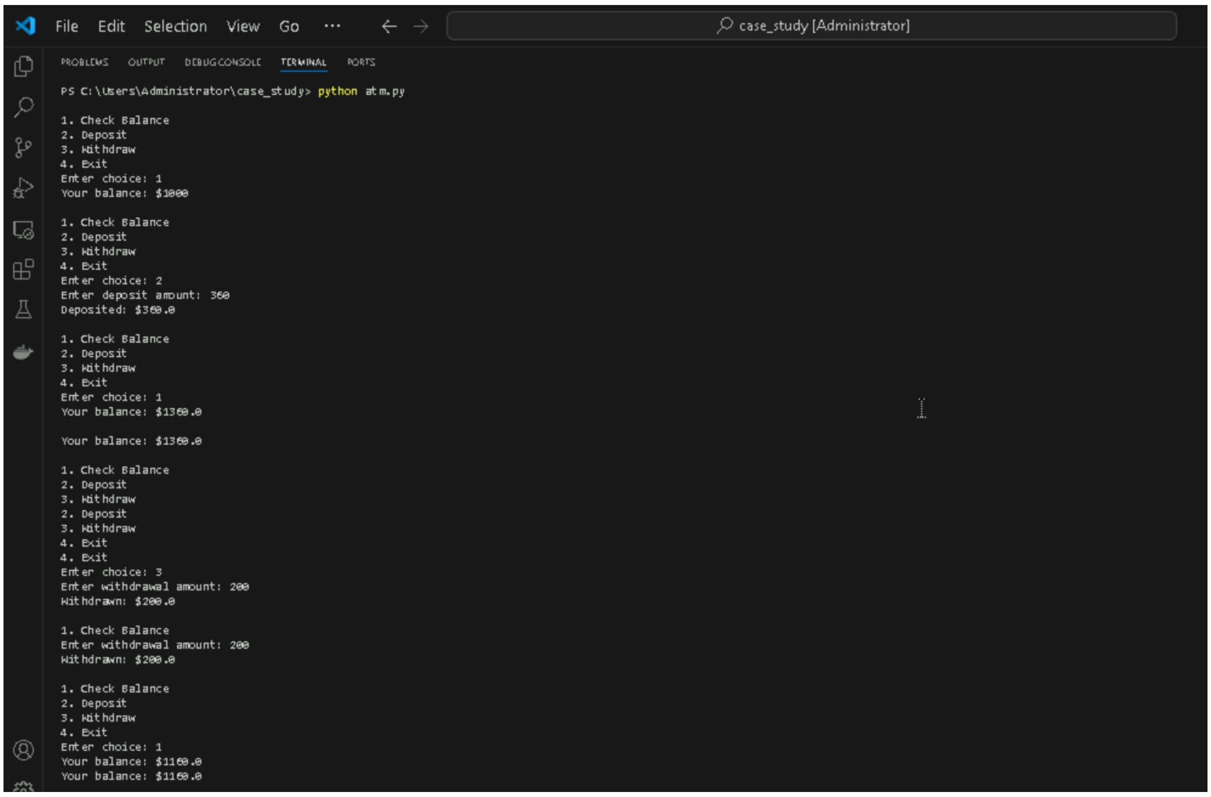
Problem Statement Develop an ATM simulation that allows users to: • Check balance • Deposit money • Withdraw money • Exit

Code:



```python
class ATM:
    def __init__(self, balance=1000):
        self.balance = balance  # Initial balance

    def check_balance(self):
        print(f"Your balance: ${self.balance}")  # Display current balance

    def deposit(self, amount):
        self.balance += amount  # Add amount to balance
        print(f"Deposited: ${amount}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient funds!")  # Check for sufficient balance
        else:
            self.balance -= amount  # Subtract amount from balance
            print(f"Withdrawn: ${amount}")

def main():
    atm = ATM()  # Create ATM object with default balance of $1000

    while True:
        # Display menu options
        print("\n1. Check Balance\n2. Deposit\n3. Withdraw\n4. Exit")
        choice = input("Enter choice: ")

        # Perform actions based on user input
        if choice == "1":
            atm.check_balance()  # Option to check balance
        elif choice == "2":
            amt = float(input("Enter deposit amount: "))  # Deposit money
            atm.deposit(amt)
        elif choice == "3":
            amt = float(input("Enter withdrawal amount: "))  # Withdraw money
            atm.withdraw(amt)
        elif choice == "4":
            print("Thank you for using the ATM!")
            break  # Exit the program
        else:
            print("Invalid choice! Try again.")  # Invalid option handling

# Start the ATM program
main()
```

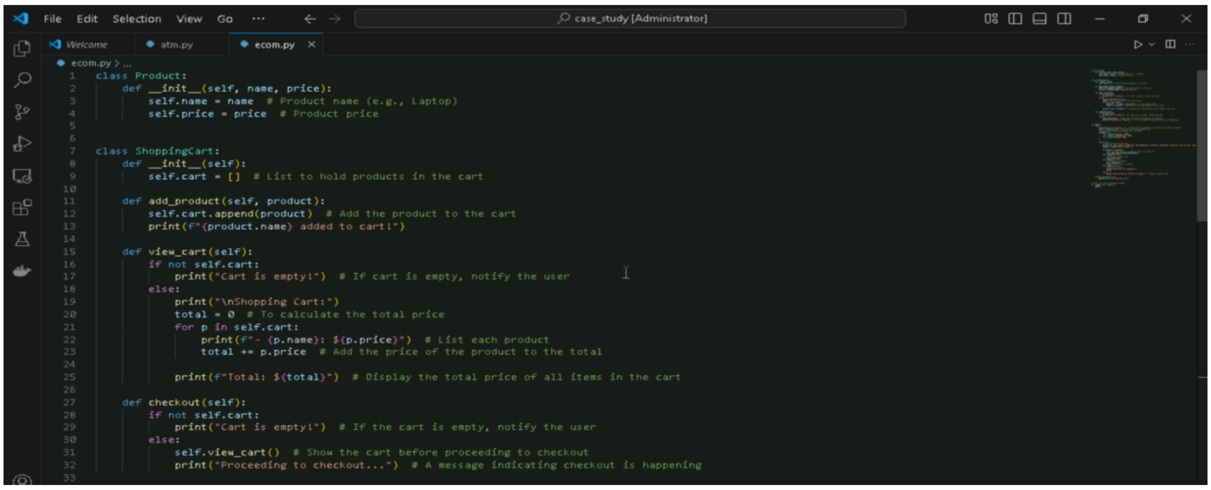
Output:



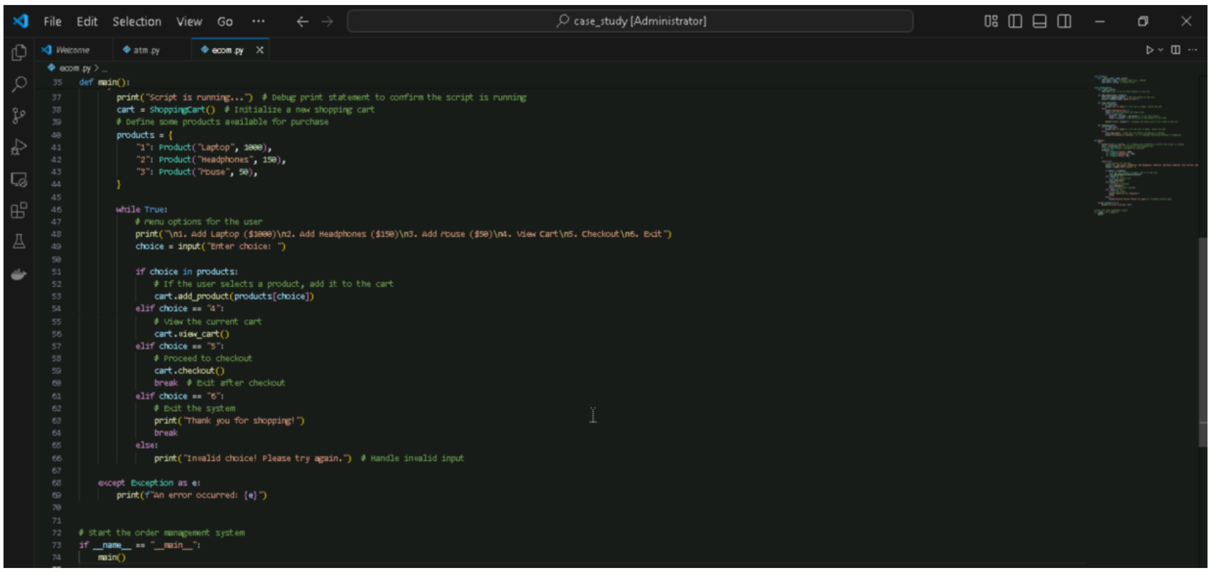
```
PS C:\Users\Administrator\case_study> python atm.py

1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 1
Your balance: $1000

1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 2
Enter deposit amount: 360
Deposited: $360.0

1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 1
Your balance: $1360.0

Your balance: $1360.0

1. Check Balance
2. Deposit
3. Withdraw
2. Deposit
3. Withdraw
4. Exit
4. Exit
Enter choice: 3
Enter withdrawal amount: 200
Withdrawn: $200.0

1. Check Balance
Enter withdrawal amount: 200
Withdrawn: $200.0

1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 1
Your balance: $1160.0
Your balance: $1160.0
```

2. Case Study: E-commerce Order Management

Problem Statement Create an Order Management System for an e-commerce platform. The system should allow: • Adding products to a cart • Viewing the cart • Checking out (calculating total price)

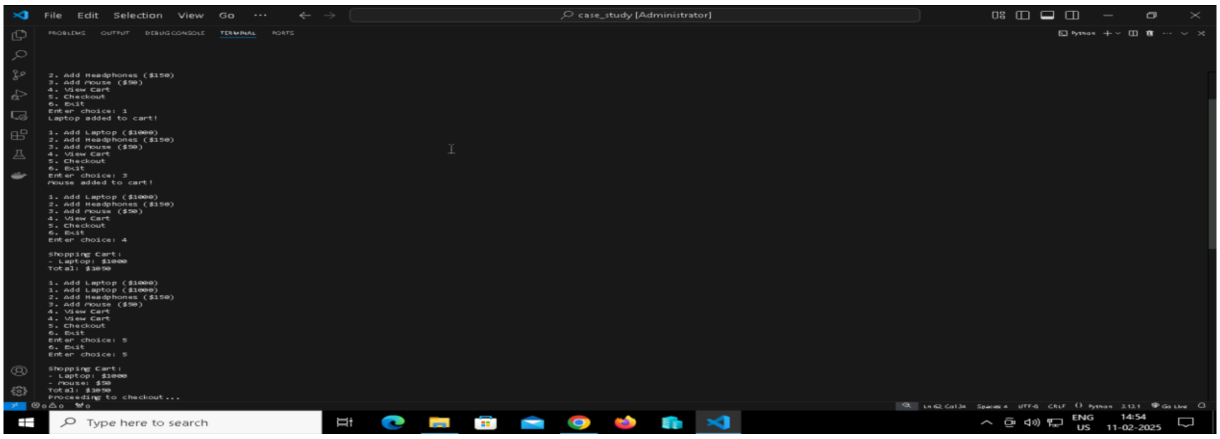


```python
class Product:
    def __init__(self, name, price):
        self.name = name  # Product name (e.g., Laptop)
        self.price = price  # Product price


class ShoppingCart:
    def __init__(self):
        self.cart = []  # List to hold products in the cart

    def add_product(self, product):
        self.cart.append(product)  # Add the product to the cart
        print(f"{product.name} added to cart!")

    def view_cart(self):
        if not self.cart:
            print("Cart is empty!")  # If cart is empty, notify the user
        else:
            print("\nShopping Cart:")
            total = 0  # To calculate the total price
            for p in self.cart:
                print(f"- {p.name}: ${p.price}")  # List each product
                total += p.price  # Add the price of the product to the total

            print(f"Total: ${total}")  # Display the total price of all items in the cart

    def checkout(self):
        if not self.cart:
            print("Cart is empty!")  # If the cart is empty, notify the user
        else:
            self.view_cart()  # Show the cart before proceeding to checkout
            print("Proceeding to checkout...")  # A message indicating checkout is happening

```



```python
def main():
        print("Script is running...")  # Debug print statement to confirm the script is running
        cart = ShoppingCart()  # Initialize a new shopping cart
        # Define some products available for purchase
        products = {
            "1": Product("Laptop", 1000),
            "2": Product("Headphones", 150),
            "3": Product("Mouse", 50),
        }

        while True:
            # Menu options for the user
            print("\n1. Add Laptop ($1000)\n2. Add Headphones ($150)\n3. Add Mouse ($50)\n4. View Cart\n5. Checkout\n6. Exit")
            choice = input("Enter choice: ")

            if choice in products:
                # If the user selects a product, add it to the cart
                cart.add_product(products[choice])
            elif choice == "4":
                # View the current cart
                cart.view_cart()
            elif choice == "5":
                # Proceed to checkout
                cart.checkout()
                break  # Exit after checkout
            elif choice == "6":
                # Exit the system
                print("Thank you for shopping!")
                break
            else:
                print("Invalid choice! Please try again.")  # Handle invalid input

    except Exception as e:
        print(f"An error occurred: {e}")


# Start the order management system
if __name__ == "__main__":
    main()
```

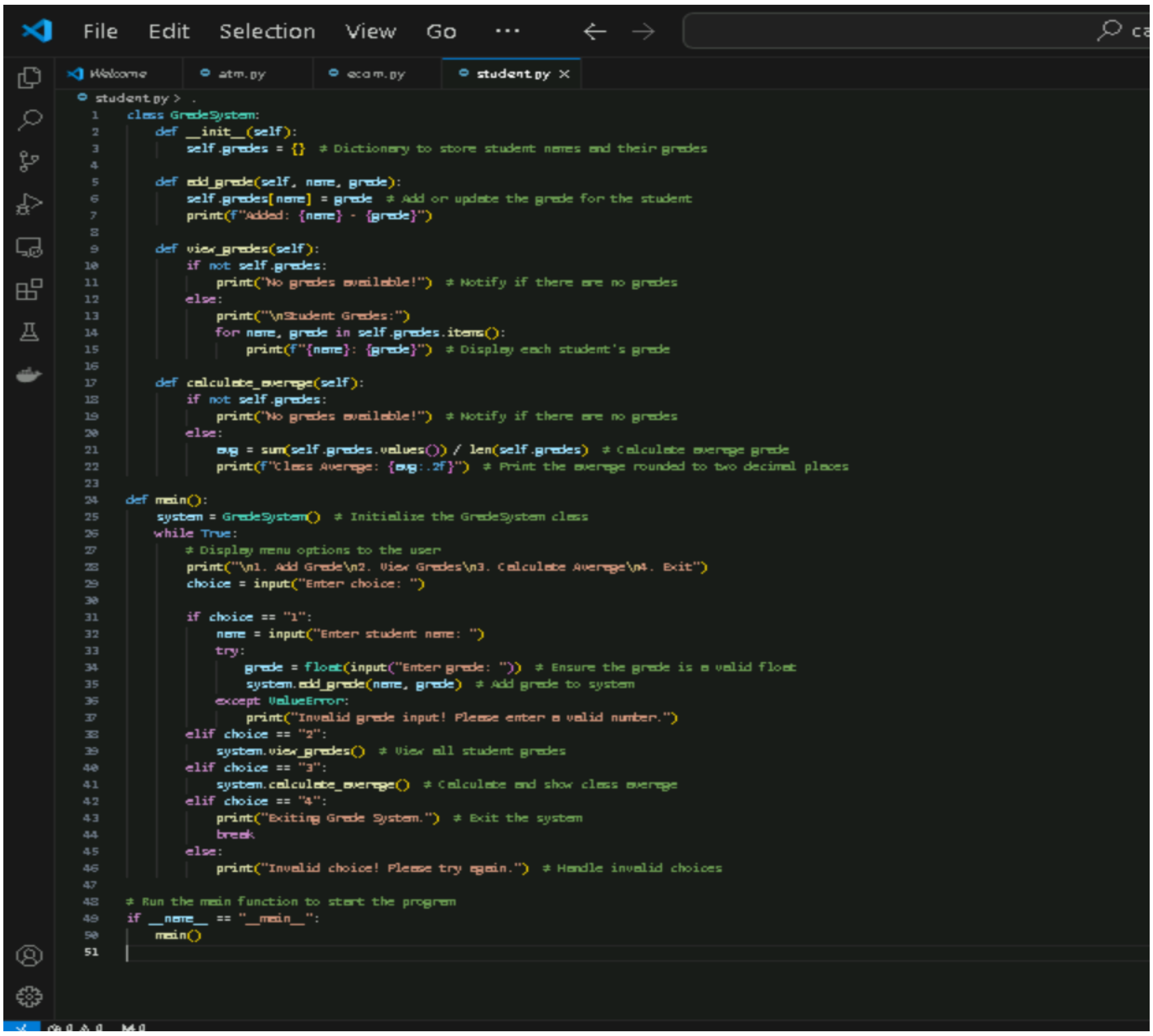
Output:



```
2. Add Headphones ($150)
3. Add Mouse ($50)
4. View Cart
5. Checkout
6. Exit
Enter choice: 1
Laptop added to cart!

1. Add Laptop ($1000)
2. Add Headphones ($150)
3. Add Mouse ($50)
4. View Cart
5. Checkout
6. Exit
Enter choice: 3
Mouse added to cart!

1. Add Laptop ($1000)
2. Add Headphones ($150)
3. Add Mouse ($50)
4. View Cart
5. Checkout
6. Exit
Enter choice: 4

Shopping Cart:
- Laptop: $1000
Total: $1050

1. Add Laptop ($1000)
1. Add Laptop ($1000)
2. Add Headphones ($150)
3. Add Mouse ($50)
4. View Cart
4. View Cart
5. Checkout
6. Exit
Enter choice: 5
6. Exit
Enter choice: 5

Shopping Cart:
- Laptop: $1000
- Mouse: $50
Total: $1050
Proceeding to checkout...
```

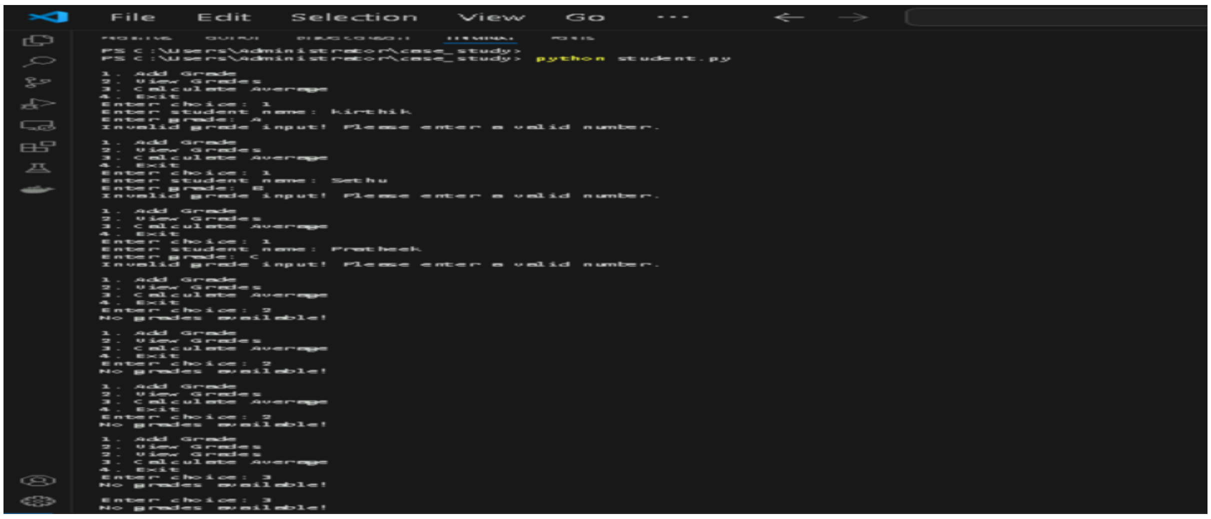
3. Case Study: Student Grade Management System

Problem Statement Develop a system to manage student grades: • Add student grades • View student grades • Calculate the average grade



```python
class GradeSystem:
    def __init__(self):
        self.grades = {}  # Dictionary to store student names and their grades

    def add_grade(self, name, grade):
        self.grades[name] = grade  # Add or update the grade for the student
        print(f"Added: {name} - {grade}")

    def view_grades(self):
        if not self.grades:
            print("No grades available!")  # Notify if there are no grades
        else:
            print("\nStudent Grades:")
            for name, grade in self.grades.items():
                print(f"{name}: {grade}")  # Display each student's grade

    def calculate_average(self):
        if not self.grades:
            print("No grades available!")  # Notify if there are no grades
        else:
            avg = sum(self.grades.values()) / len(self.grades)  # Calculate average grade
            print(f"Class Average: {avg:.2f}")  # Print the average rounded to two decimal places

def main():
    system = GradeSystem()  # Initialize the GradeSystem class
    while True:
        # Display menu options to the user
        print("\n1. Add Grade\n2. View Grades\n3. Calculate Average\n4. Exit")
        choice = input("Enter choice: ")

        if choice == "1":
            name = input("Enter student name: ")
            try:
                grade = float(input("Enter grade: "))  # Ensure the grade is a valid float
                system.add_grade(name, grade)  # Add grade to system
            except ValueError:
                print("Invalid grade input! Please enter a valid number.")
        elif choice == "2":
            system.view_grades()  # View all student grades
        elif choice == "3":
            system.calculate_average()  # Calculate and show class average
        elif choice == "4":
            print("Exiting Grade System.")  # Exit the system
            break
        else:
            print("Invalid choice! Please try again.")  # Handle invalid choices

# Run the main function to start the program
if __name__ == "__main__":
    main()
```

Output:

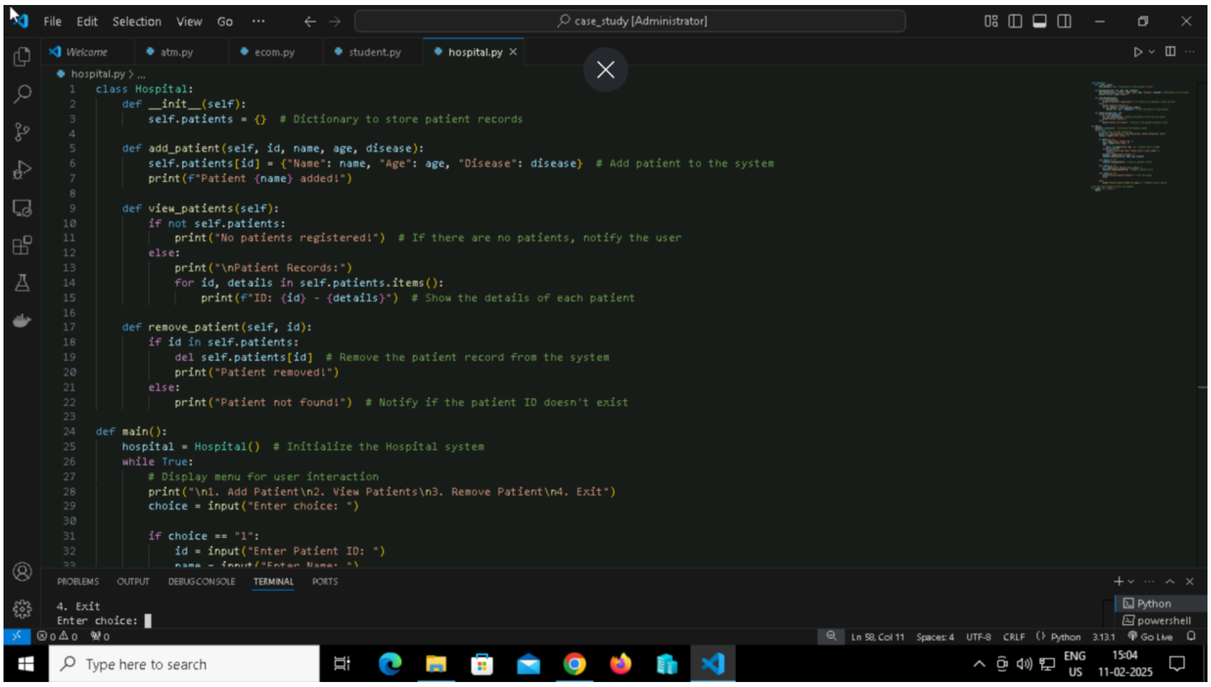


```
PS C:\Users\Administrator\case_study>
PS C:\Users\Administrator\case_study> python student.py

1. Add Grade
2. View Grades
3. Calculate Average
4. Exit
Enter choice: 1
Enter student name: kirthik
Enter grade: A
Invalid grade input! Please enter a valid number.

1. Add Grade
2. View Grades
3. Calculate Average
4. Exit
Enter choice: 1
Enter student name: Sethu
Enter grade: B
Invalid grade input! Please enter a valid number.

1. Add Grade
2. View Grades
3. Calculate Average
4. Exit
Enter choice: 1
Enter student name: Pratheek
Enter grade: C
Invalid grade input! Please enter a valid number.

1. Add Grade
2. View Grades
3. Calculate Average
4. Exit
Enter choice: 2
No grades available!

1. Add Grade
2. View Grades
3. Calculate Average
4. Exit
Enter choice: 2
No grades available!

1. Add Grade
2. View Grades
3. Calculate Average
4. Exit
Enter choice: 2
No grades available!

1. Add Grade
2. View Grades
2. View Grades
3. Calculate Average
4. Exit
Enter choice: 3
No grades available!

Enter choice: 3
No grades available!
```

4. Case Study: Hospital Patient Management

Problem Statement Create a hospital management system that: • Adds new patients • Displays patient details • Deletes patients.

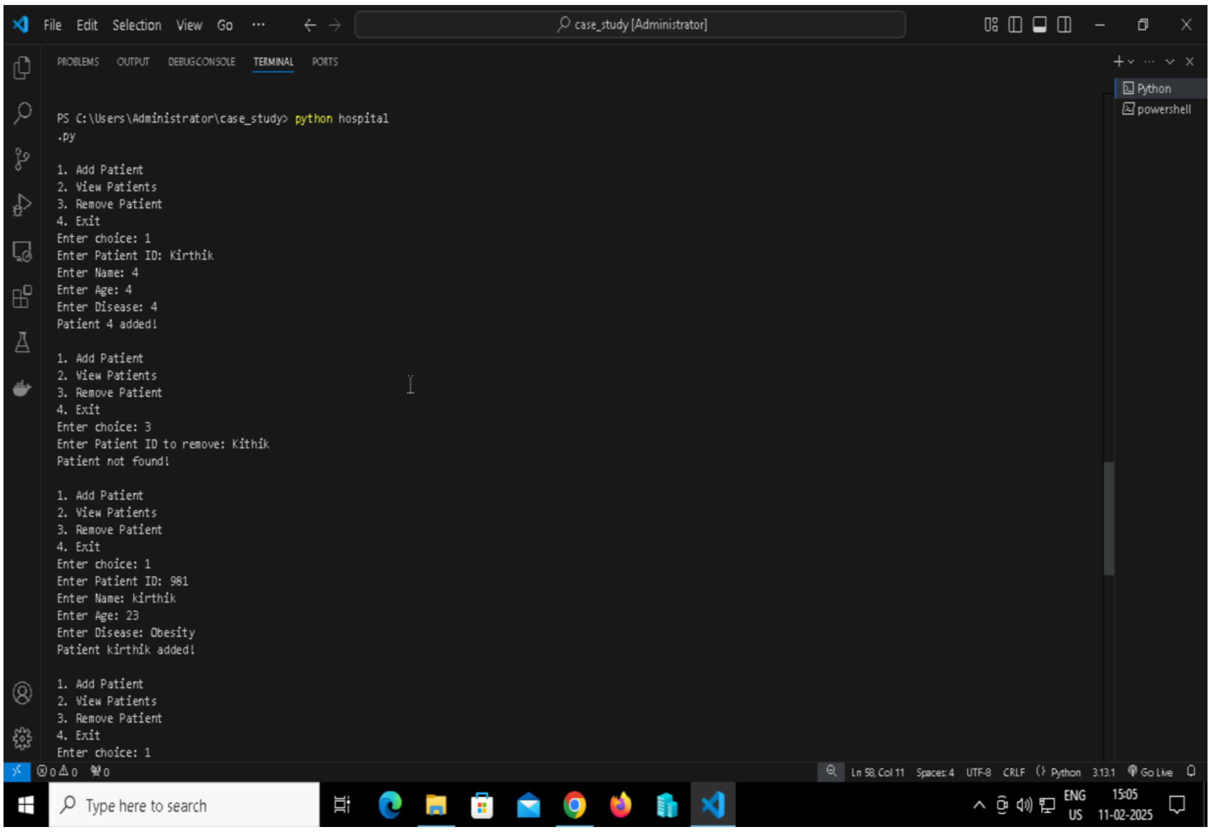
Code:



```python
class Hospital:
    def __init__(self):
        self.patients = {}  # Dictionary to store patient records

    def add_patient(self, id, name, age, disease):
        self.patients[id] = {"Name": name, "Age": age, "Disease": disease}  # Add patient to the system
        print(f"Patient {name} added!")

    def view_patients(self):
        if not self.patients:
            print("No patients registered!")  # If there are no patients, notify the user
        else:
            print("\nPatient Records:")
            for id, details in self.patients.items():
                print(f"ID: {id} - {details}")  # Show the details of each patient

    def remove_patient(self, id):
        if id in self.patients:
            del self.patients[id]  # Remove the patient record from the system
            print("Patient removed!")
        else:
            print("Patient not found!")  # Notify if the patient ID doesn't exist

def main():
    hospital = Hospital()  # Initialize the Hospital system
    while True:
        # Display menu for user interaction
        print("\n1. Add Patient\n2. View Patients\n3. Remove Patient\n4. Exit")
        choice = input("Enter choice: ")

        if choice == "1":
            id = input("Enter Patient ID: ")
            name = input("Enter Name: ")
```

```
4. Exit
Enter choice: 
```

Output:



```
PS C:\Users\Administrator\case_study> python hospital
.py

1. Add Patient
2. View Patients
3. Remove Patient
4. Exit
Enter choice: 1
Enter Patient ID: Kirthik
Enter Name: 4
Enter Age: 4
Enter Disease: 4
Patient 4 added!

1. Add Patient
2. View Patients
3. Remove Patient
4. Exit
Enter choice: 3
Enter Patient ID to remove: Kithik
Patient not found!

1. Add Patient
2. View Patients
3. Remove Patient
4. Exit
Enter choice: 1
Enter Patient ID: 981
Enter Name: kirthik
Enter Age: 23
Enter Disease: Obesity
Patient kirthik added!

1. Add Patient
2. View Patients
3. Remove Patient
4. Exit
Enter choice: 1
```