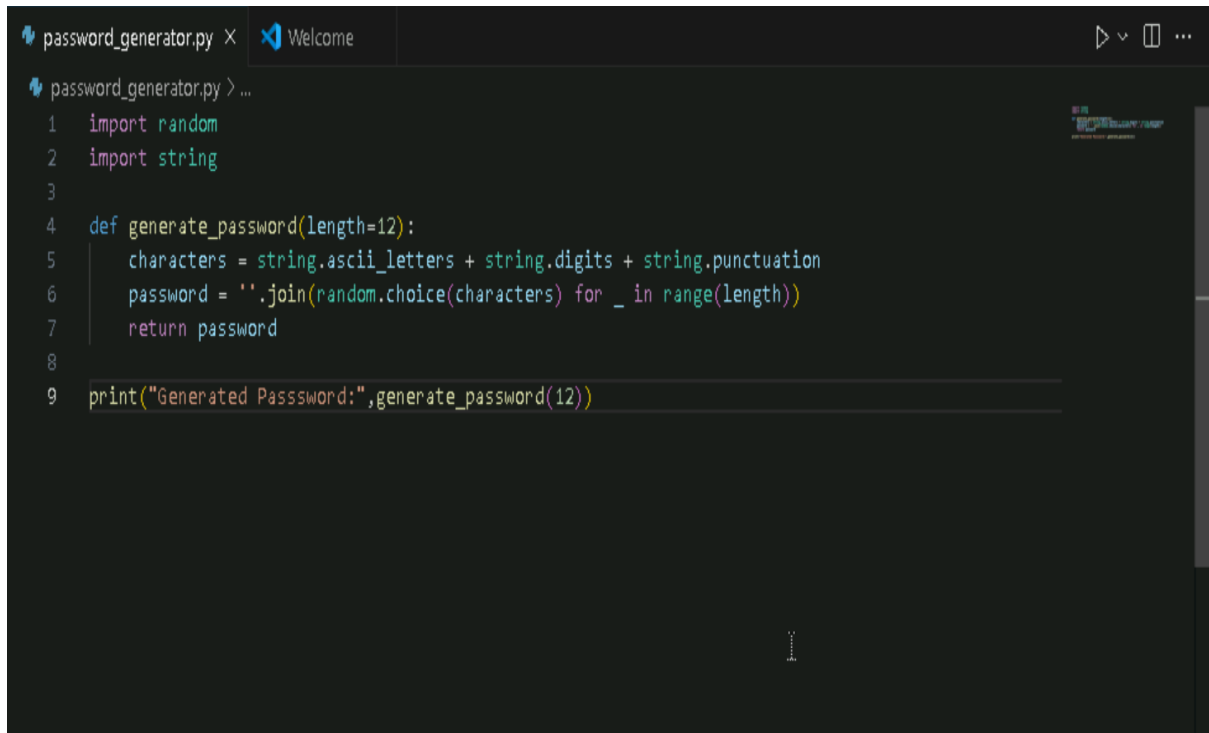


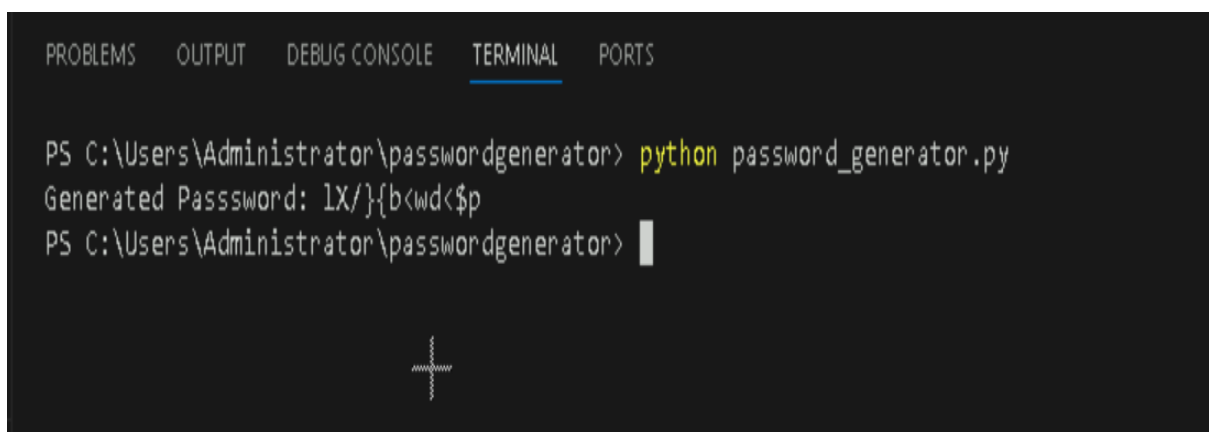
# 1. Password Generator

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.

A screenshot of a code editor with a dark theme. The top bar shows 'password\_generator.py' and a 'Welcome' tab. The editor contains the following Python code:

```
1 import random
2 import string
3
4 def generate_password(length=12):
5     characters = string.ascii_letters + string.digits + string.punctuation
6     password = ''.join(random.choice(characters) for _ in range(length))
7     return password
8
9 print("Generated Password:", generate_password(12))
```

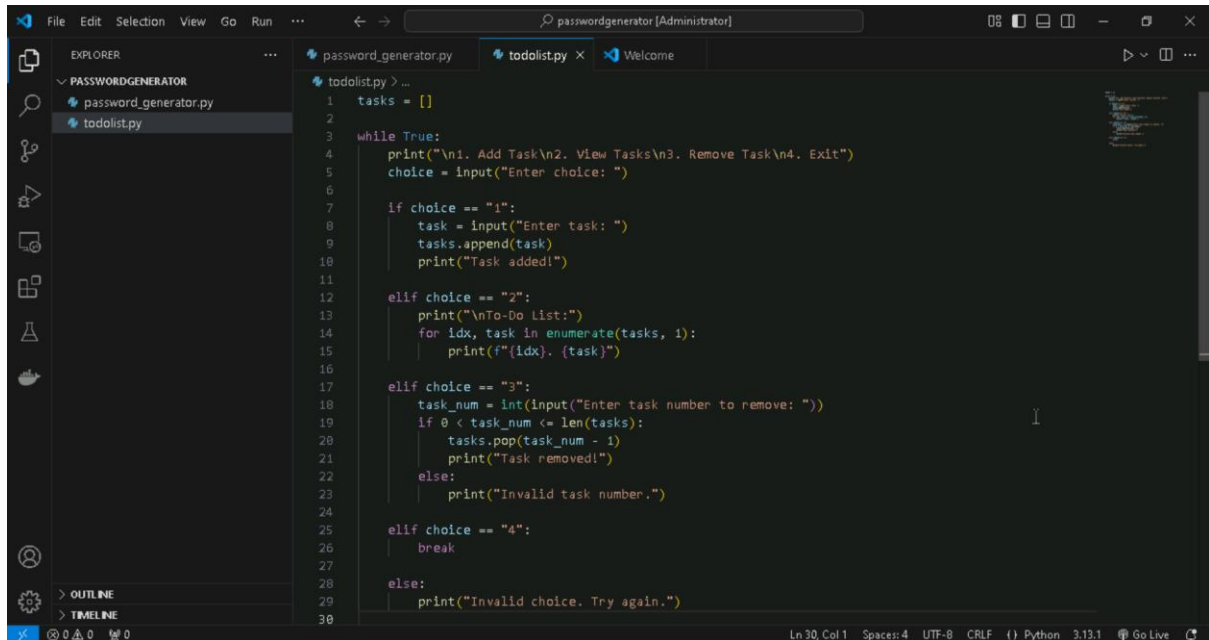
Output:

A screenshot of a terminal window with a dark theme. The terminal shows the command 'python password\_generator.py' being executed, followed by the output 'Generated Password: lX/}{b<wd<\$p'. The prompt 'PS C:\Users\Administrator\passwordgenerator>' is visible at the end of the line.

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

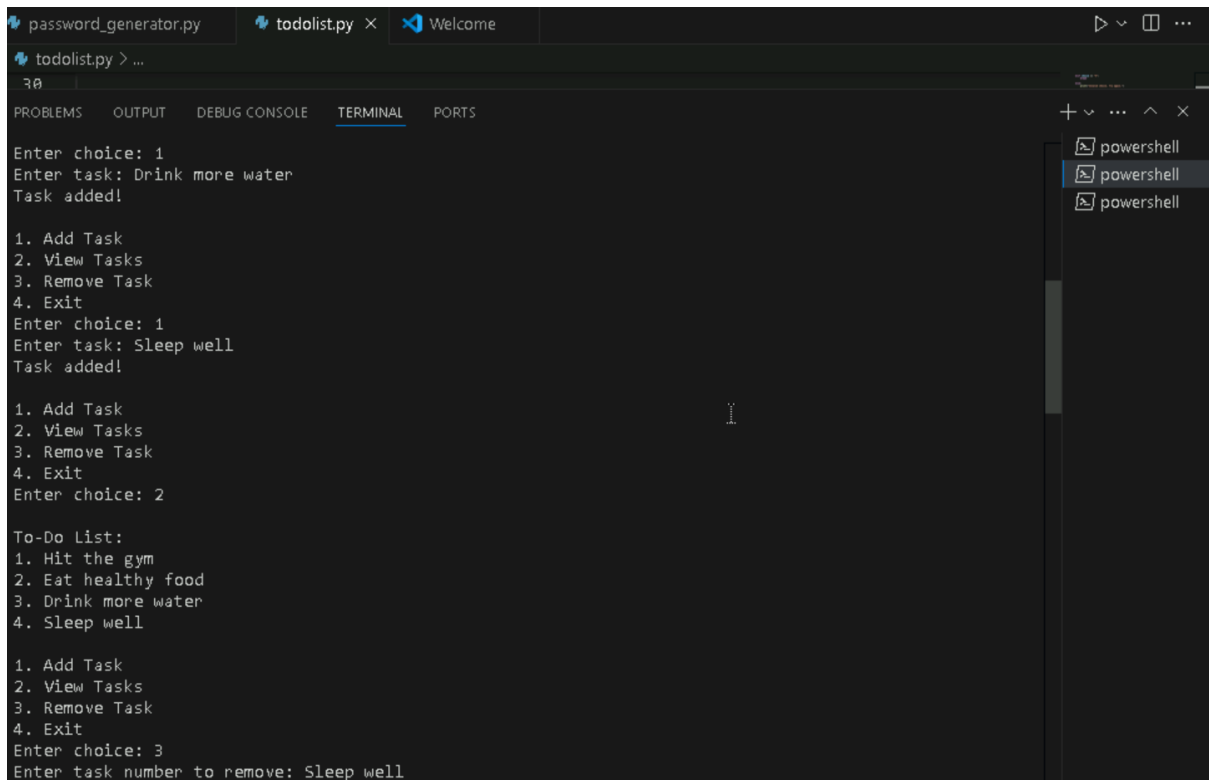
## 2. To-Do List (CLI)

1. Create a list to store tasks.
2. Provide options: Add, View, Remove, Exit.
3. Loop until the user exits.



```
1 tasks = []
2
3 while True:
4     print("\n1. Add Task\n2. View Tasks\n3. Remove Task\n4. Exit")
5     choice = input("Enter choice: ")
6
7     if choice == "1":
8         task = input("Enter task: ")
9         tasks.append(task)
10        print("Task added!")
11
12    elif choice == "2":
13        print("\nTo-Do List:")
14        for idx, task in enumerate(tasks, 1):
15            print(f"{idx}. {task}")
16
17    elif choice == "3":
18        task_num = int(input("Enter task number to remove: "))
19        if 0 < task_num <= len(tasks):
20            tasks.pop(task_num - 1)
21            print("Task removed!")
22        else:
23            print("Invalid task number.")
24
25    elif choice == "4":
26        break
27
28    else:
29        print("Invalid choice. Try again.")
30
```

### Output:



```
password_generator.py  todolist.py x  Welcome
todolist.py > ...
30

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

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!

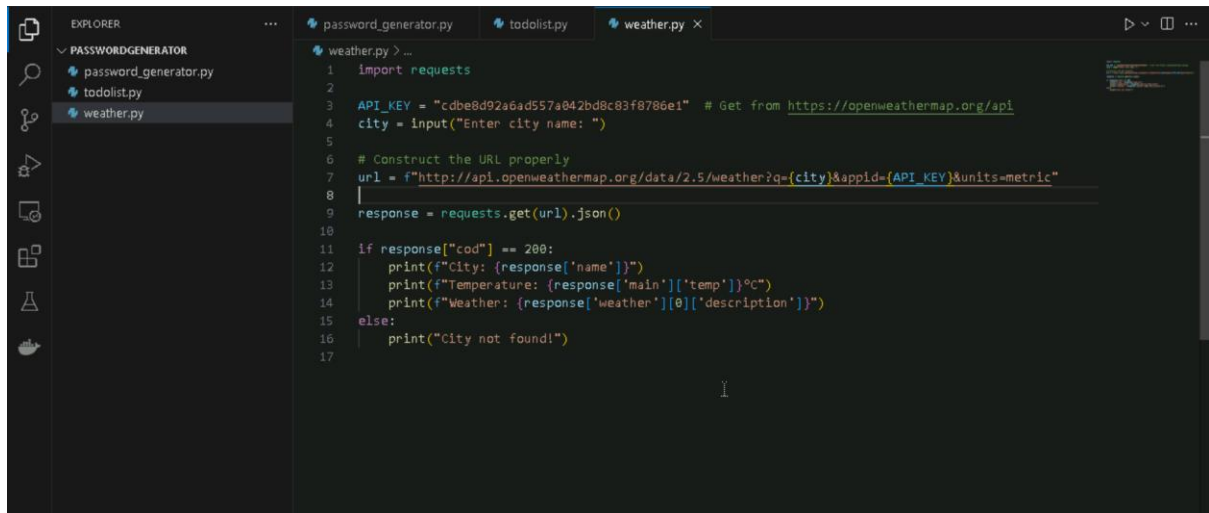
1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Enter choice: 2

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.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'PASSWORDGENERATOR' with three files: 'password\_generator.py', 'todolist.py', and 'weather.py'. The 'weather.py' file is selected and its code is displayed in the editor. The code is a Python script that uses the 'requests' library to fetch weather data from the OpenWeatherMap API. It prompts the user to enter a city name, constructs a URL with the city name and API key, and then prints the city name, temperature, and weather condition.

```
1 import requests
2
3 API_KEY = "cdbe8d92a6ad557a042bd8c83f8786e1" # Get from https://openweathermap.org/api
4 city = input("Enter city name: ")
5
6 # Construct the URL properly
7 url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}&units=metric"
8
9 response = requests.get(url).json()
10
11 if response["cod"] == 200:
12     print(f"City: {response['name']}")
13     print(f"Temperature: {response['main']['temp']}°C")
14     print(f"Weather: {response['weather'][0]['description']}")
15 else:
16     print("City not found!")
17
```

**Output:**



The screenshot shows a terminal window with the following 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.

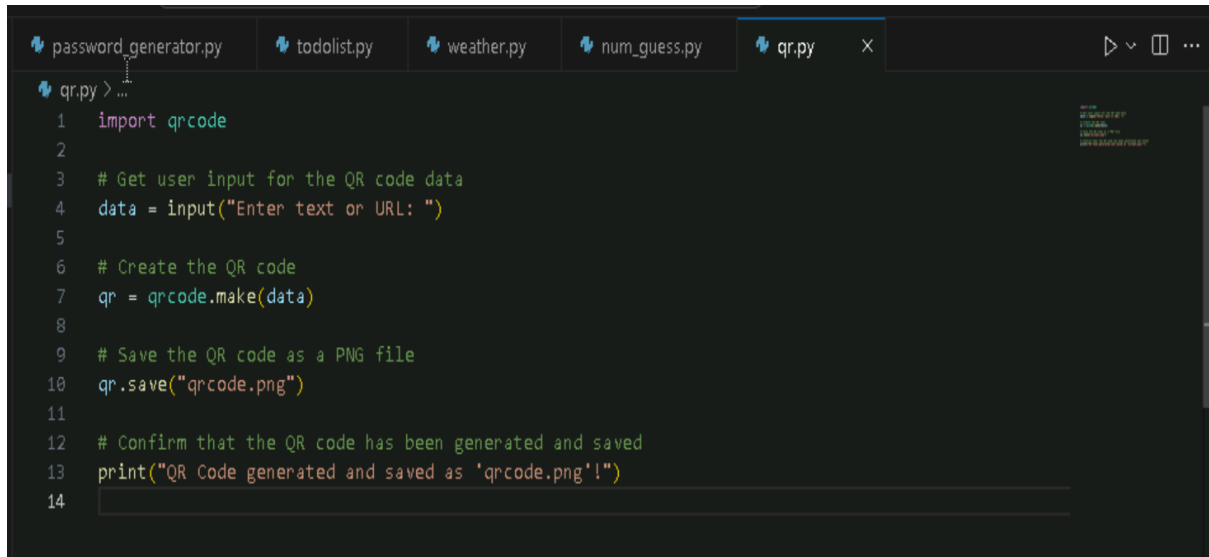
```
num_guess.py > ...
1  import random
2
3  # Generate a random number between 1 and 100
4  number = random.randint(1, 100)
5
6  while True:
7      guess = int(input("Guess the number (1-100): "))
8
9      if guess < number:
10         print("Too low! Try again.")
11     elif guess > number:
12         print("Too high! Try again.")
13     else:
14         print("Congratulations! You guessed it right.")
15         break
16
```

### Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
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>
```

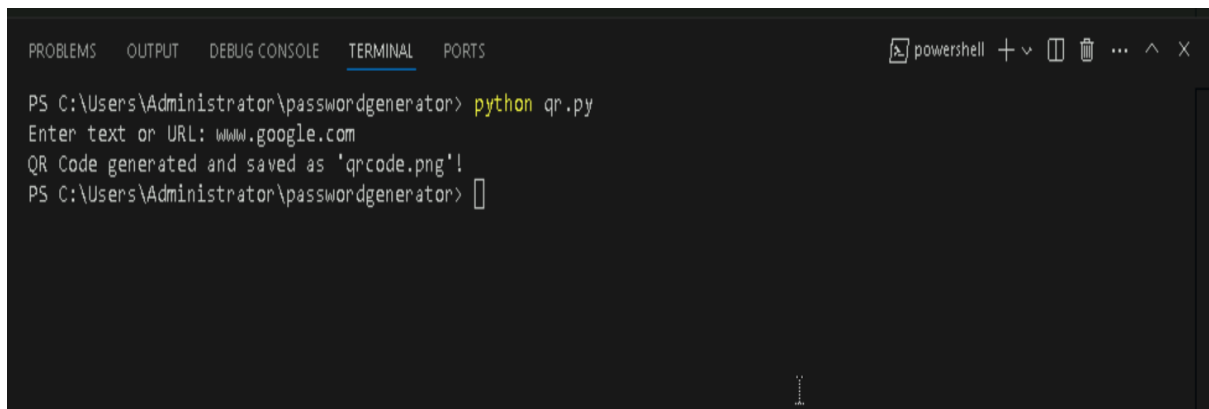
## 5. QR Code Generator

1. Install qrcode library (pip install qrcode).
2. Take user input (text/link) to convert.
3. Generate and save the QR code.



```
password_generator.py  todolist.py  weather.py  num_guess.py  qr.py  x
qr.py > ...
1  import qrcode
2
3  # Get user input for the QR code data
4  data = input("Enter text or URL: ")
5
6  # Create the QR code
7  qr = qrcode.make(data)
8
9  # Save the QR code as a PNG file
10 qr.save("qrcode.png")
11
12 # Confirm that the QR code has been generated and saved
13 print("QR Code generated and saved as 'qrcode.png'!")
14
```

Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
powershell + v [ ] [ ] ... ^ x

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> [ ]
```

