TASK NO. 1

Python Code: hangman_game.py

```
import random


# List of words to choose from
word_list = ['python', 'hangman', 'programming', 'developer', 'automation', 'keyboard', 'function']


# Choose a random word
secret_word = random.choice(word_list)
guessed_letters = []
max_attempts = 6
attempts_left = max_attempts


def display_word():
    return " ".join([letter if letter in guessed_letters else "_" for letter in secret_word])


print("🎮 Welcome to Hangman!")
print(f"You have {max_attempts} incorrect guesses allowed.")
print(display_word())


while attempts_left > 0:
    guess = input("\nGuess a letter: ").lower()


    if not guess.isalpha() or len(guess) != 1:
        print("Please enter a single alphabetical character.")
```

```
        continue

    if guess in guessed_letters:
        print("You've already guessed that letter.")
        continue

    guessed_letters.append(guess)

    if guess in secret_word:
        print("✅Correct guess!")
    else:
        attempts_left -= 1
        print(f"❌Wrong guess. Attempts left: {attempts_left}")

    print(display_word())

    # Win condition
    if all(letter in guessed_letters for letter in secret_word):
        print("\n Congratulations! You guessed the word:", secret_word)
        break
else:
    print("\n Game Over! The word was:", secret_word)
```

Sample Gameplay:

 Welcome to Hangman!

You have 6 incorrect guesses allowed.

_ _ _ _ _ _ _

Guess a letter: p

✅Correct guess!

p _ _ _ _ _ _

Guess a letter: x

❌Wrong guess. Attempts left: 5

p _ _ _ _ _ _

...

TASK NO. 2

Personal Portfolio Website — Basic Version

/portfolio

├── index.html

├── style.css

├── script.js

└── images/ (photo, project images, etc.)

(HTML CODE)

<!DOCTYPE html>

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My Portfolio</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>Your Name</h1>
    <p>Web Developer | Python Enthusiast | Designer</p>
    <nav>
      <a href="#about">About</a>
      <a href="#projects">Projects</a>
      <a href="#resume">Resume</a>
      <a href="#contact">Contact</a>
    </nav>
  </header>

  <section id="about">
    <h2>About Me</h2>
    <p>Brief introduction about yourself.</p>
  </section>

  <section id="projects">
    <h2>Projects</h2>
```

```
    <div class="project">
      <h3>Stock Portfolio Tracker</h3>
      <p>A tool to track and manage stock investments.</p>
    </div>
</section>

<section id="resume">
    <h2>Resume</h2>
    <a href="your_resume.pdf" download>Download My Resume</a>
</section>

<section id="contact">
    <h2>Contact</h2>
    <form id="contact-form">
      <input type="text" placeholder="Your Name" required><br>
      <input type="email" placeholder="Your Email" required><br>
      <textarea placeholder="Your Message" required></textarea><br>
      <button type="submit">Send</button>
    </form>
</section>

<footer>
    <p>© 2025 Your Name</p>
</footer>
```

```
    <script src="script.js"></script>
</body>
</html>
```

(CSS CODE)

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}
header {
    background-color: #333;
    color: white;
    padding: 20px;
    text-align: center;
}
nav a {
    margin: 0 10px;
    color: white;
    text-decoration: none;
}
section {
    padding: 40px;
}
.project {
```

```
    background-color: #f0f0f0;
    padding: 20px;
    margin-bottom: 20px;
}
footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
}
```

(SCRIPT JS CODE)

```
// Simple form interaction
document.getElementById('contact-form').addEventListener('submit', function(event) {
    event.preventDefault();
    alert('Thanks for contacting me! I will get back to you soon.');
});
```

Stock Portfolio Tracker Web App — Python (Flask) + HTML/CSS

```
/stock_tracker
    ├── app.py
    ├── templates/
```

```
|   └── index.html
└── static/
    └── style.css
```

Install flask with the help of terminl

```
pip install flask
```

flask code

```
from flask import Flask, render_template, request, redirect

app = Flask(__name__)

# Dummy portfolio data
portfolio = []

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        symbol = request.form['symbol'].upper()
        shares = int(request.form['shares'])
        purchase_price = float(request.form['purchase_price'])
        portfolio.append({
```

```
            'symbol': symbol,
            'shares': shares,
            'purchase_price': purchase_price
        })
        return redirect('/')
    return render_template('index.html', portfolio=portfolio)

@app.route('/delete/<symbol>')
def delete_stock(symbol):
    global portfolio
    portfolio = [stock for stock in portfolio if stock['symbol'] != symbol.upper()]
    return redirect('/')

if __name__ == '__main__':
    app.run(debug=True)
```

(HTML CODE)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Stock Portfolio Tracker</title>
    <link rel="stylesheet" href="/static/style.css">
```

```
</head>
<body>
  <h1>Stock Portfolio Tracker</h1>

  <form method="POST">
    <input type="text" name="symbol" placeholder="Stock Symbol" required>
    <input type="number" name="shares" placeholder="Shares" required>
    <input type="text" name="purchase_price" placeholder="Purchase Price" required>
    <button type="submit">Add Stock</button>
  </form>

  <h2>Your Portfolio</h2>
  {% if portfolio %}
    <table>
      <tr>
        <th>Symbol</th>
        <th>Shares</th>
        <th>Purchase Price</th>
        <th>Action</th>
      </tr>
      {% for stock in portfolio %}
      <tr>
        <td>{{ stock.symbol }}</td>
        <td>{{ stock.shares }}</td>
        <td>${{ stock.purchase_price }}</td>
```

```
            <td><a href="/delete/{{ stock.symbol }}">Delete</a></td>
          </tr>
          {% endfor %}
        </table>
    {% else %}
        <p>No stocks in your portfolio yet.</p>
    {% endif %}
</body>
</html>
```

(CSS CODE)

```
body {
    font-family: Arial, sans-serif;
    margin: 30px;
}
form input, form button {
    margin: 5px;
    padding: 8px;
}
table {
    width: 80%;
    margin-top: 20px;
    border-collapse: collapse;
}
th, td {
```

```
    border: 1px solid #ccc;
    padding: 10px;
    text-align: center;
}
a {
    color: red;
}
```

TASK NO.3

Simple Rule-Based Chatbot (pure Python)

```
# basic_chatbot.py

def chatbot_response(user_input):
    user_input = user_input.lower()

    if "hello" in user_input or "hi" in user_input:
```

```
        return "Hello! How can I help you today?"
    elif "how are you" in user_input:
        return "I'm just a bot, but I'm doing great! Thanks for asking."
    elif "your name" in user_input:
        return "I'm your friendly chatbot!"
    elif "bye" in user_input:
        return "Goodbye! Have a nice day!"
    else:
        return "I'm sorry, I don't understand that. Can you rephrase?"

def main():
    print("Chatbot: Hello! I'm a simple chatbot. Type 'bye' to exit.")
    while True:
        user_input = input("You: ")
        if user_input.lower() == 'bye':
            print("Chatbot: Goodbye!")
            break
        response = chatbot_response(user_input)
        print(f"Chatbot: {response}")

if __name__ == "__main__":
    main()
```

We can use (NLTK) for the extra feature add on in he chat bot

It will add some extra feature in our chat bot .

TASK NO.4

Personal Portfolio Website

/portfolio

```
├── index.html
├── style.css
├── script.js
```

(HTML CODE)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Your Name - Portfolio</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>Your Name</h1>
    <p>Web Developer | Python Enthusiast | Automation Expert</p>
    <nav>
      <a href="#about">About</a>
      <a href="#projects">Projects</a>
      <a href="#resume">Resume</a>
      <a href="#contact">Contact</a>
    </nav>
  </header>

  <section id="about">
    <h2>About Me</h2>
    <p>Hello! I'm passionate about building web applications and automating workflows using Python.</p>
  </section>
```

```
<section id="projects">
  <h2>Projects</h2>
  <div class="project">
    <h3>Task Automation with Python</h3>
    <p>A Python script that automatically organizes files into folders based on type.</p>
  </div>
  <div class="project">
    <h3>Basic Chatbot</h3>
    <p>A text-based chatbot that can respond to simple user questions.</p>
  </div>
</section>

<section id="resume">
  <h2>Resume</h2>
  <a href="resume.pdf" download>Download My Resume</a>
</section>

<section id="contact">
  <h2>Contact Me</h2>
  <form id="contact-form">
    <input type="text" placeholder="Your Name" required><br>
    <input type="email" placeholder="Your Email" required><br>
    <textarea placeholder="Your Message" required></textarea><br>
    <button type="submit">Send Message</button>
  </form>
```

```
    </section>

    <footer>
        <p>© 2025 Your Name</p>
    </footer>

    <script src="script.js"></script>
</body>
</html>
```

(CSS CODE)

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    scroll-behavior: smooth;
}
header {
    background-color: #4CAF50;
    color: white;
    padding: 20px;
    text-align: center;
}
```

```
nav a {
    color: white;
    margin: 0 15px;
    text-decoration: none;
    font-weight: bold;
}
section {
    padding: 40px;
}
.project {
    background-color: #f2f2f2;
    padding: 20px;
    margin-bottom: 20px;
}
footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
}
```

(SCRIPT CODE)

NAME OF FILE-script.js

```
// Contact form simple alert
```

```
document.getElementById('contact-form').addEventListener('submit', function(e) {
    e.preventDefault();
    alert('Thank you for reaching out! I will get back to you soon.');
});
```

Task Automation with Python Scripts

Example: **File Organizer**

NAME OF FILE -file_organizer.py

```
import os
import shutil

# Set the directory you want to organize
source_folder = "C:/Users/YourUsername/Downloads"  # Change this!

# Define target folders for different file types
file_types = {
    "Images": ['.jpg', '.jpeg', '.png', '.gif'],
    "Documents": ['.pdf', '.docx', '.txt'],
    "Videos": ['.mp4', '.mkv', '.mov'],
    "Music": ['.mp3', '.wav'],
    "Archives": ['.zip', '.rar', '.tar']
}
```

```
def organize_files(folder):
    for filename in os.listdir(folder):
        file_path = os.path.join(folder, filename)

        if os.path.isfile(file_path):
            file_ext = os.path.splitext(filename)[1].lower()

            moved = False
            for folder_name, extensions in file_types.items():
                if file_ext in extensions:
                    target_folder = os.path.join(folder, folder_name)
                    if not os.path.exists(target_folder):
                        os.makedirs(target_folder)
                    shutil.move(file_path, os.path.join(target_folder, filename))
                    print(f"Moved: {filename} --> {folder_name}")
                    moved = True
                    break

            if not moved:
                print(f"Skipped: {filename} (no matching category)")

if __name__ == "__main__":
    organize_files(source_folder)
    print("File organization complete!")
```