**Python Tasks**

* **Task: Contact Book**
* **Description**: Create a program that manages a simple contact book. Users can add a contact (name and phone number), view all contacts, and search for a contact by name.
* **Requirements**:
* Store contacts in a list of dictionaries (e.g., {"name": "Alice", "phone": "1234567890"}).
* Write functions for:
* Adding a contact (prompt for name and phone).
* Displaying all contacts.
* Searching for a contact by name (case-insensitive).
* Validate phone numbers (e.g., must be numeric, 10 digits).
* Use a while loop for a menu-driven interface (options: add, view, search, exit).
* Tested Skills: Lists, dictionaries, functions, user input, string manipulation, loops.

**Code:**

# Contact book list

contacts = []

# Validate phone number

def is\_valid\_phone(phone):

return phone.isdigit() and len(phone) == 10

# Add a contact

def add\_contact():

name = input("Enter contact name: ").strip()

phone = input("Enter 10-digit phone number: ").strip()

if is\_valid\_phone(phone):

contacts.append({"name": name, "phone": phone})

print(f"Contact '{name}' added successfully.\n")

else:

print("Invalid phone number. It must be numeric and 10 digits long.\n")

# Display all contacts

def view\_contacts():

if not contacts:

print("Contact book is empty.\n")

else:

print("\nAll Contacts:")

for contact in contacts:

print(f"Name: {contact['name']}, Phone: {contact['phone']}")

print()

# Search for a contact by name

def search\_contact():

search\_name = input("Enter name to search: ").strip().lower()

found = False

for contact in contacts:

if contact["name"].lower() == search\_name:

print(f"Found Contact: Name: {contact['name']}, Phone: {contact['phone']}\n")

found = True

break

if not found:

print("Contact not found.\n")

# Menu loop

def contact\_book\_menu():

while True:

print("Contact Book Menu:")

print("1. Add Contact")

print("2. View All Contacts")

print("3. Search Contact")

print("4. Exit")

choice = input("Choose an option (1-4): ").strip()

if choice == "1":

add\_contact()

elif choice == "2":

view\_contacts()

elif choice == "3":

search\_contact()

elif choice == "4":

print("Exiting Contact Book. Goodbye!")

break

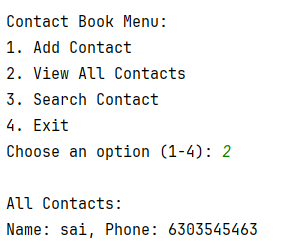
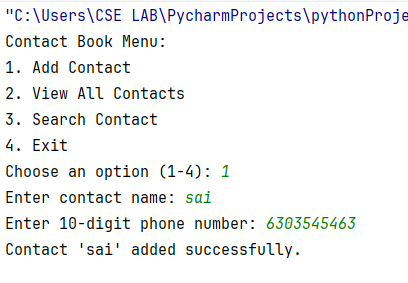
else:

print("Invalid choice. Please enter a number from 1 to 4.\n")

# Run the program

contact\_book\_menu()

**Output:**

****

* **Task: Expense Tracker**
* **Description**: Build a program to track daily expenses. Users can add expenses (description and amount) and view the total spent.
* **Requirements**:
* Store expenses in a list of tuples (e.g., ("Coffee", 5.50)).
* Write functions for:
  + Adding an expense (prompt for description and amount).
  + Calculating and displaying the total of all expenses.
  + Displaying all expenses.
* Validate that the amount is a positive number using try-except.
* Use a menu-driven interface (options: add, view, total, exit).
* Tested Skills: Lists, tuples, functions, error handling, user input, basic calculations.

**Code:**

expenses = []

def add\_expense():

description = input("Enter expense description: ").strip()

try:

amount = float(input("Enter expense amount: "))

if amount > 0:

expenses.append((description, amount))

print(f"Expense '{description}' added successfully.\n")

else:

print("Amount must be a positive number.\n")

except ValueError:

print("Invalid input. Please enter a valid number for the amount.\n")

def view\_expenses():

if not expenses:

print("No expenses recorded.\n")

else:

print("\nAll Expenses:")

for desc, amt in expenses:

print(f"{desc}: ${amt:.2f}")

print()

def total\_expenses():

total = sum(amount for \_, amount in expenses)

print(f"\nTotal Spent: ${total:.2f}\n")

def expense\_tracker\_menu():

while True:

print("Expense Tracker Menu:")

print("1. Add Expense")

print("2. View All Expenses")

print("3. View Total Spent")

print("4. Exit")

choice = input("Choose an option (1-4): ").strip()

if choice == "1":

add\_expense()

elif choice == "2":

view\_expenses()

elif choice == "3":

total\_expenses()

elif choice == "4":

print("Exiting Expense Tracker. Goodbye!")

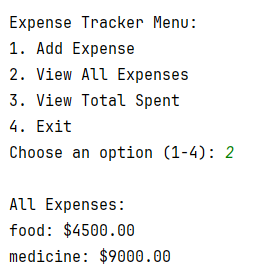
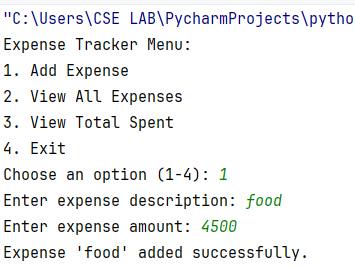
break

else:

print("Invalid choice. Please enter a number from 1 to 4.\n")

expense\_tracker\_menu()

**Output:**

****

* **Task: File-Based To-Do List**
  + **Description**: Write a program that lets users add and view to-do items, saving them to a file so they persist between runs.
  + **Requirements**:
* Store to-do items as a list of strings.
* Write functions for:
  + - Adding a to-do item.
    - Reading and displaying all to-do items from a file (todos.txt).
    - Saving the to-do list to the file after each addition.
* Handle file errors (e.g., file not found) using try-except.
* Use a simple command-line interface (options: add, view, exit).
* **Tested Skills:** File I/O, lists, functions, error handling, user input.

**Code:**

def load\_todos():

try:

with open("todos.txt", "r") as file:

return [line.strip() for line in file.readlines()]

except FileNotFoundError:

return []

def save\_todo(item):

try:

with open("todos.txt", "a") as file:

file.write(item + "\n")

except Exception as e:

print(f"Error saving to file: {e}")

def add\_todo():

item = input("Enter a to-do item: ").strip()

if item:

save\_todo(item)

print("To-do item added.\n")

else:

print("To-do item cannot be empty.\n")

def view\_todos():

todos = load\_todos()

if not todos:

print("No to-do items found.\n")

else:

print("\nYour To-Do List:")

for idx, item in enumerate(todos, 1):

print(f"{idx}. {item}")

print()

def todo\_menu():

while True:

print("To-Do List Menu:")

print("1. Add To-Do")

print("2. View To-Dos")

print("3. Exit")

choice = input("Choose an option (1-3): ").strip()

if choice == "1":

add\_todo()

elif choice == "2":

view\_todos()

elif choice == "3":

print("Exiting To-Do List. Goodbye!")

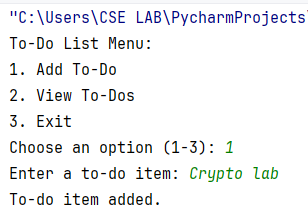
break

else:

print("Invalid choice. Please enter a number from 1 to 3.\n")

todo\_menu()

**Output:**

****

* **Task: Simple Calculator Class**
  + **Description**: Create a Python class for a basic calculator that performs addition, subtraction, multiplication, and division.
  + **Requirements:**
    - Define a Calculator class with methods for each operation.
    - Each method takes two numbers as arguments and returns the result.
    - Include input validation to prevent division by zero using try-except.
    - Write a main program that:
      * Creates a Calculator object.
      * Prompts the user to choose an operation and input two numbers.
      * Displays the result and loops until the user exits.
  + **Tested Skills:** Classes, methods, error handling, user input, loops.

**Code:**

class Calculator:

def add(self, a, b):

return a + b

def subtract(self, a, b):

return a - b

def multiply(self, a, b):

return a \* b

def divide(self, a, b):

try:

return a / b

except ZeroDivisionError:

return "Error: Cannot divide by zero"

def get\_number(prompt):

while True:

try:

return float(input(prompt))

except ValueError:

print("Invalid input. Please enter a valid number.")

def main():

calc = Calculator()

while True:

print("Simple Calculator Menu:")

print("1. Add")

print("2. Subtract")

print("3. Multiply")

print("4. Divide")

print("5. Exit")

choice = input("Choose an option (1-5): ").strip()

if choice == "5":

print("Exiting calculator. Goodbye!")

break

if choice in {"1", "2", "3", "4"}:

num1 = get\_number("Enter the first number: ")

num2 = get\_number("Enter the second number: ")

if choice == "1":

result = calc.add(num1, num2)

elif choice == "2":

result = calc.subtract(num1, num2)

elif choice == "3":

result = calc.multiply(num1, num2)

elif choice == "4":

result = calc.divide(num1, num2)

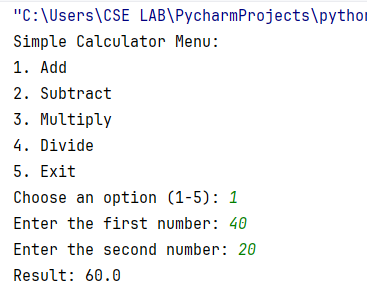
print(f"Result: {result}\n")

else:

print("Invalid choice. Please select from 1 to 5.\n")

main()

**Output:**



* + **Task: Unique Word Lister**
  + **Description**: Write a program that takes a sentence from the user, splits it into words, and displays only the unique words in alphabetical order.
  + **Requirements**:
    - Prompt the user for a sentence.
    - Split the sentence into words, ignoring case and punctuation.
    - Use a set to store unique words.
    - Sort the unique words alphabetically and display them.
    - Write a function to process the sentence and return the unique words.
    - Handle empty input gracefully.

**Code:**

import string

def get\_unique\_words(sentence):

if not sentence.strip():

return []

cleaned = sentence.translate(str.maketrans('', '', string.punctuation))

words = cleaned.lower().split()

unique\_words = sorted(set(words))

return unique\_words

def main():

sentence = input("Enter a sentence: ").strip()

unique\_words = get\_unique\_words(sentence)

if not unique\_words:

print("No words found.\n")

else:

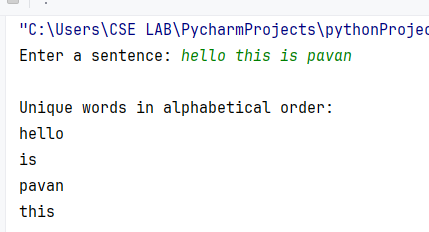
print("\nUnique words in alphabetical order:")

for word in unique\_words:

print(word)

main()

**Output:**



**HTML**

**Lists, Links, and Images**

1. **Write a HTML program to explain the working of lists.**
   * Include: ordered list, unordered list, nested lists, ordered list inside unordered list, and definition lists.

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<h1>Working of lists in HTML</h1>

<h2>Ordered list</h2>

<ol type = "A">

<li>Health Care</li>

<li>Rent</li>

<li>Bills</li>

<li>Insurance</li>

</ol>

<ol type="1">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

<li>Breakfast</li>

</ol>

<h2>Unordered list</h2>

<ul style="list-style-type:square;">

<li>SAI DURGA</li>

<li>SAIBOMMA</li>

<li>VISHNU SVJSSSS</li>

</ul>

<h2>Nested lists</h2>

<ul>

<li>Coffee</li>

<li>Biryani

<ul>

<li>Veg Biryani</li>

<li>Chick Biryani</li>

</ul>

</li>

<li>Milk</li>

</ul>

<h2>Definition List</h2>

<dl>

<dt>HTML</dt>

<dd>HyperText Markup Language, used to create web pages.</dd>

<dt>CSS</dt>

<dd>Cascading Style Sheets, used to style web pages.</dd>

<dt>JavaScript</dt>

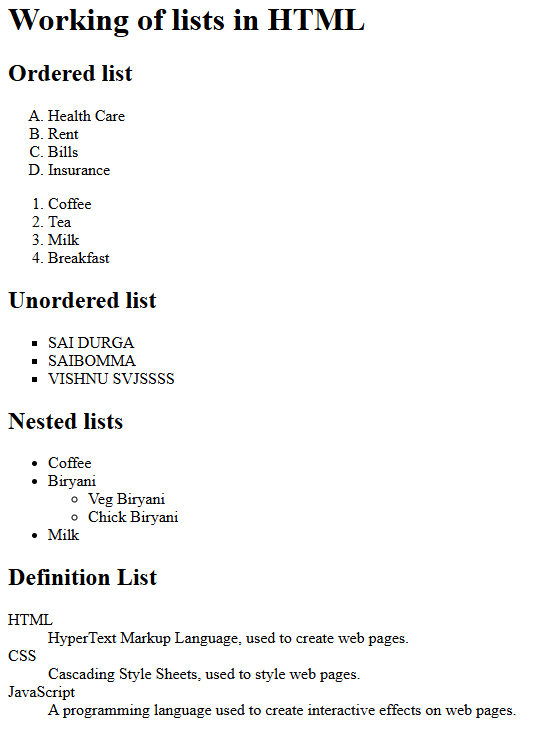
<dd>A programming language used to create interactive effects on web pages.</dd>

</dl>

</body>

</html>

**Output:**

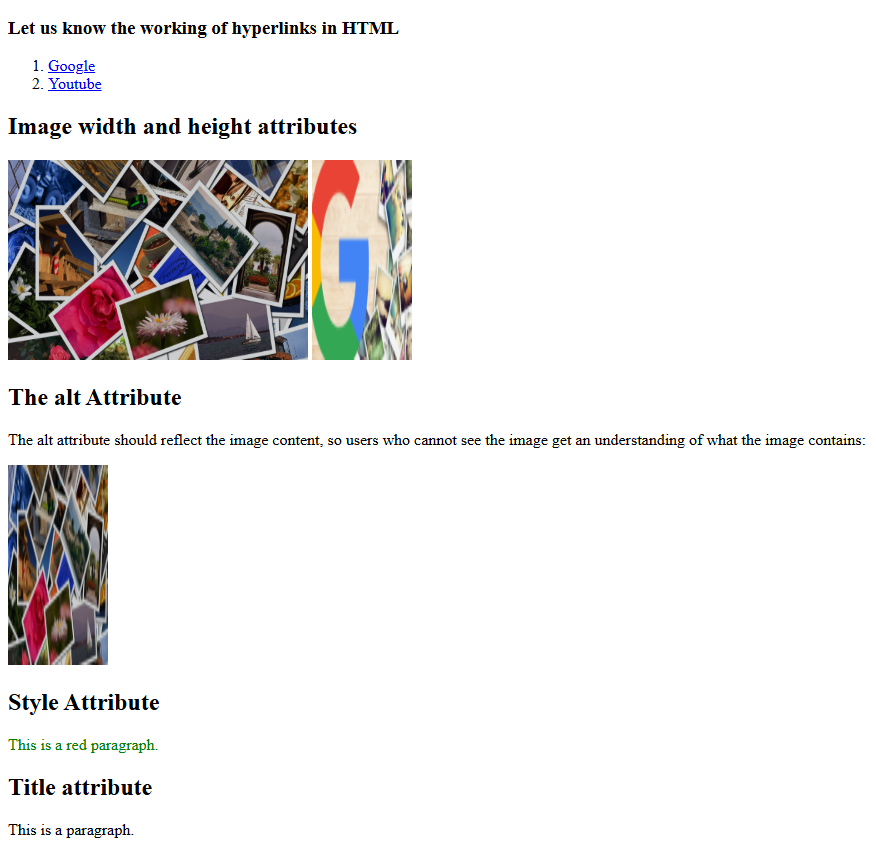


1. **Write a HTML program to explain the working of hyperlinks using the <a> tag and href, target attributes.**

**Code:**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
  
<h3>Let us know the working of hyperlinks in HTML</h3>  
<ol>  
 <li>  
 <a href="https://www.google.com">Google</a>  
 </li>  
 <li>  
 <a href="https://www.youtube.com">Youtube</a>  
 </li>  
</ol>  
  
  
<h2>Image width and height attributes</h2>  
<img src="img\_2.jpg" width="300" height="200">  
<img src="https://searchengineland.com/wp-content/seloads/2016/03/google-photos-images5-ss-1920.jpg" width = "100" height = "200">  
  
<h2>The alt Attribute</h2>  
<p>The alt attribute should reflect the image content, so users who cannot see the image get an understanding of what the image contains:</p>  
<img src="img\_2.jpg" width = "100" height = "200" alt="SHIP OR BOAT">  
  
<h2>Style Attribute</h2>  
<p style="color:green;">This is a red paragraph.</p>  
  
<h2>Title attribute</h2>  
<p title="I'm a student">This is a paragraph.</p>  
  
  
</body>  
</html>

**Output:**

****

1. **Create a HTML document that includes your image and your friend’s image with specific height and width.**
   * On clicking the images, it should navigate to their respective profiles.

**Code**:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Profile Images</title>  
</head>  
<body>  
 <h1>Our Profiles</h1>  
  
 <a href="https://www.microsoft.com/en-us/microsoft-365/outlook/email-and-calendar-software-microsoft-outlook?deeplink=%2fmail%2f0%2f%3fnlp%3d0&sdf=0" target="\_blank">  
 <img src="img\_2.jpg" alt="saibomma" width="200" height="200">  
 </a>  
  
 <a href="https://www.google.com/imgres?q=images%20photos&imgurl=https%3A%2F%2Fimages.unsplash.com%2Fphoto-1678557856807-7ae6ff6893d1%3Ffm%3Djpg%26q%3D60%26w%3D3000%26ixlib%3Drb-4.1.0%26ixid%3DM3wxMjA3fDB8MHxzZWFyY2h8M3x8Ym95fGVufDB8fDB8fHwy&imgrefurl=https%3A%2F%2Funsplash.com%2Fphotos%2Fa-close-up-of-a-childs-face-smiling-MTtqnJgqfeU&docid=PG1zQTguag8cDM&tbnid=Y2FCCg6O7OTCAM&vet=12ahUKEwjel7bE54uOAxVkxjgGHaKzPNwQM3oFCIYBEAA..i&w=3000&h=4500&hcb=2&ved=2ahUKEwjel7bE54uOAxVkxjgGHaKzPNwQM3oFCIYBEAA" target="\_blank">  
 <img src="img\_3.jpg" alt="SVJSS" width="200" height="200">  
 </a>  
</body>  
</html>

**Output:**

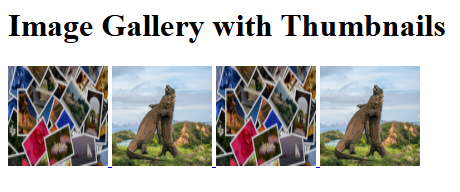


1. **Write a HTML program to demonstrate an image gallery using thumbnails (100×100 pixels).**
   * Each thumbnail should link to the full-size image.

**Code:**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Thumbnail Image Gallery</title>  
</head>  
<body>  
 <h1>Image Gallery with Thumbnails</h1>  
  
  
 <div>  
  
 <a href="img\_2.jpg" target="\_blank">  
 <img src="img\_2.jpg" alt="Image 1" width="100" height="100">  
 </a>  
  
  
 <a href="img\_3.jpg" target="\_blank">  
 <img src="img\_3.jpg" alt="Image 2" width="100" height="100">  
 </a>  
  
  
 <a href="images/fullsize3.jpg" target="\_blank">  
 <img src="img\_2.jpg" alt="Image 3" width="100" height="100">  
 </a>  
  
 <a href="images/fullsize4.jpg" target="\_blank">  
 <img src="img\_3.jpg" alt="Image 4" width="100" height="100">  
 </a>  
 </div>  
</body>  
</html>

**Output:**



**HTML Tables, Forms, and Frames**

1. **Write a HTML program to explain the working of tables.**
   * Use tags: <table>, <tr>, <th>, <td>
   * Include attributes: border, rowspan, colspan.

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>HTML Table Example</title>

</head>

<body>

<h1>HTML Table Example</h1>

<table border="1" cellpadding="5" cellspacing="0">

<tr>

<th rowspan="2">Name</th>

<th colspan="2">Contact Details</th>

<th rowspan="2">Age</th>

</tr>

<tr>

<th>Email</th>

<th>Phone</th>

</tr>

<tr>

<td>SVJSS</td>

<td>22137@OUTLOOK.com</td>

<td>923-456-7890</td>

<td>30</td>

</tr>

<tr>

<td>POTNURU</td>

<td>SAIBOMMAe@MOVIERULZ.com</td>

<td>887-654-3210</td>

<td>25</td>

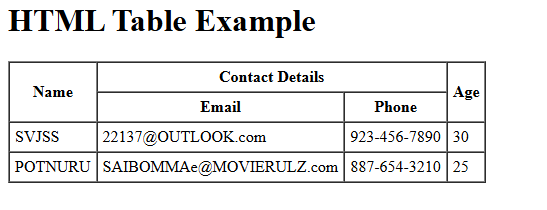
</tr>

</table>

</body>

</html>

**Output:**

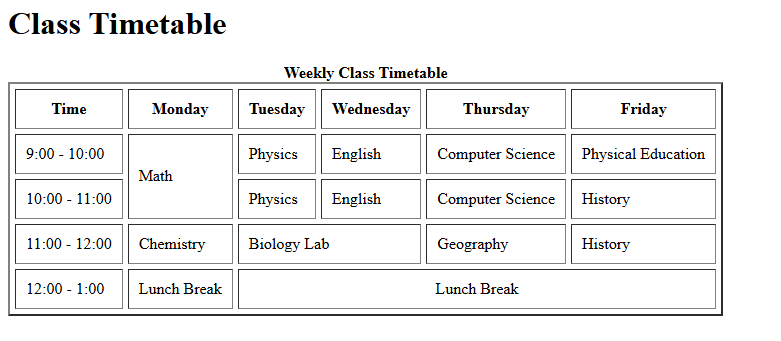
****

1. **Write a HTML program to create a class timetable using a table.**
   * Use: <caption>, cellspacing, cellpadding, border, rowspan, colspan.

Code:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8" />  
 <title>Class Timetable Example</title>  
</head>  
<body>  
  
<h1>Class Timetable</h1>  
  
<table border="2" cellspacing="5" cellpadding="10">  
 <caption><strong>Weekly Class Timetable</strong></caption>  
  
 <tr>  
 <th>Time</th>  
 <th>Monday</th>  
 <th>Tuesday</th>  
 <th>Wednesday</th>  
 <th>Thursday</th>  
 <th>Friday</th>  
 </tr>  
 <tr>  
 <td>9:00 - 10:00</td>  
 <td rowspan="2">Math</td>  
 <td>Physics</td>  
 <td>English</td>  
 <td>Computer Science</td>  
 <td>Physical Education</td>  
 </tr>  
  
  
 <tr>  
 <td>10:00 - 11:00</td>  
  
 <td>Physics</td>  
 <td>English</td>  
 <td>Computer Science</td>  
 <td>History</td>  
 </tr>  
  
  
 <tr>  
 <td>11:00 - 12:00</td>  
 <td>Chemistry</td>  
 <td colspan="2">Biology Lab</td>  
 <td>Geography</td>  
 <td>History</td>  
 </tr>  
  
 <tr>  
 <td>12:00 - 1:00</td>  
 <td>Lunch Break</td>  
 <td colspan="4" style="text-align:center;">Lunch Break</td>  
 </tr>  
  
</table>  
  
</body>  
</html>

**Output:**



1. **Write a HTML program to design a Registration Form.**
   * Include: text field, password field, number field, date of birth, checkboxes, radio buttons, list box (<select> and <option>), <textarea>, submit and reset buttons.
   * Use a table layout for better presentation.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Registration Form Example</title>

</head>

<body>

<h1>Registration Form</h1>

<form action="/submit\_registration" method="post">

<table border="1" cellpadding="8" cellspacing="0">

<tr>

<td><label for="fullname">Full Name:</label></td>

<td><input type="text" id="fullname" name="fullname" required></td>

</tr>

<tr>

<td><label for="password">Password:</label></td>

<td><input type="password" id="password" name="password" required></td>

</tr>

<tr>

<td><label for="age">Age:</label></td>

<td><input type="number" id="age" name="age" min="1" max="120" required></td>

</tr>

<tr>

<td><label for="dob">Date of Birth:</label></td>

<td><input type="date" id="dob" name="dob" required></td>

</tr>

<tr>

<td>Gender:</td>

<td>

<input type="radio" id="male" name="gender" value="Male" required>

<label for="male">Male</label>

<input type="radio" id="female" name="gender" value="Female">

<label for="female">Female</label>

<input type="radio" id="other" name="gender" value="Other">

<label for="other">Other</label>

</td>

</tr>

<tr>

<td>Hobbies:</td>

<td>

<input type="checkbox" id="reading" name="hobbies" value="Reading">

<label for="reading">Reading</label>

<input type="checkbox" id="traveling" name="hobbies" value="Traveling">

<label for="traveling">Traveling</label>

<input type="checkbox" id="gaming" name="hobbies" value="Gaming">

<label for="gaming">Gaming</label>

<input type="checkbox" id="sports" name="hobbies" value="Sports">

<label for="sports">Sports</label>

</td>

</tr>

<tr>

<td><label for="country">Country:</label></td>

<td>

<select id="country" name="country" required>

<option value="">--Select Country--</option>

<option value="usa">USA</option>

<option value="uk">United Kingdom</option>

<option value="india">India</option>

<option value="australia">Australia</option>

<option value="canada">Canada</option>

</select>

</td>

</tr>

<tr>

<td><label for="about">About You:</label></td>

<td>

<textarea id="about" name="about" rows="4" cols="40" placeholder="Tell us about yourself..."></textarea>

</td>

</tr>

<tr>

<td colspan="2" style="text-align:center;">

<input type="submit" value="Submit">

<input type="reset" value="Reset">

</td>

</tr>

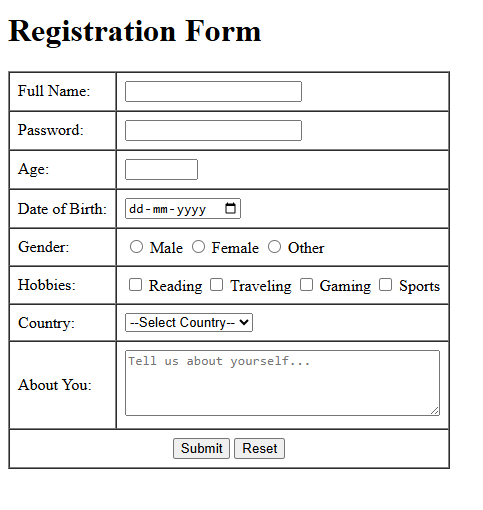
</table>

</form>

</body>

</html>

**Output:**



1. **Write a HTML program to demonstrate the use of frames.**
   * Divide the page into 3 parts:
     + Frame 1: image
     + Frame 2: paragraph
     + Frame 3: hyperlink
   * Use the noframes attribute for fallback content.

Code:

**HTML5**

1. **Write a HTML program using semantic tags:**
   * Use: <article>, <aside>, <figure>, <figcaption>, <footer>, <header>, <main>, <nav>, <section>, <div>, <span>.
2. **Write a HTML program to embed audio and video into a web page.**

**Module 1: Git Fundamentals (Local Version Control)**

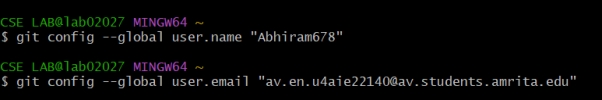
**Lab 1: Getting Started with Git**

**Objective**: To introduce basic Git commands for initializing a repository, tracking changes, and committing.

**Programs**:

**1. Setup and First Commit:**

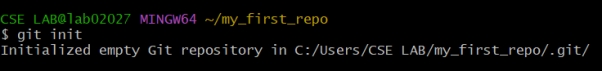
* Install Git on your system.
* Configure Git with your username and email.



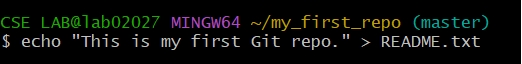
* Create a new directory for a project (e.g., my\_first\_repo).



* Initialize a Git repository within this directory (git init).



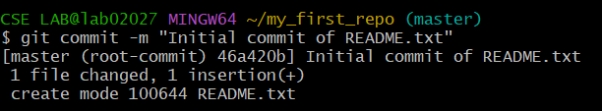
* Create a simple text file (e.g., README.txt) with some content.



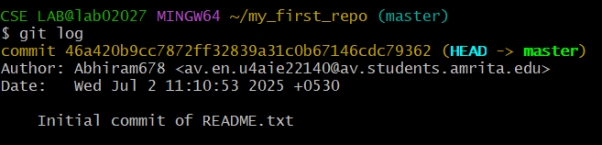
* Add the file to the staging area (git add README.txt).

C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps5.jpg

* Commit the changes with a meaningful message (git commit -m "Initial commit of README.txt").



* View the commit history (git log).

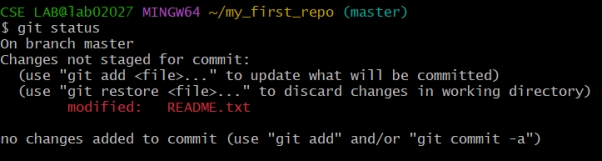


**2. Modifying and Tracking Changes:**

* Modify README.txt and add new content.

C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps8.jpg

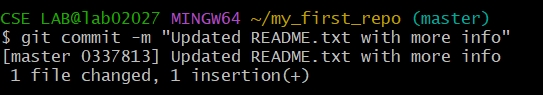
* Check the status of your repository (git status).



* Stage the modified file.

C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps10.jpg

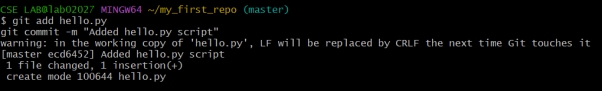
* Commit the changes.



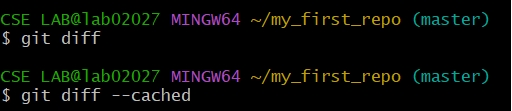
* Create another file (e.g., hello.py).

C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps12.jpg

* Add and commit hello.py.



* Practice git diff to see unstaged and staged changes.



**3. Undoing Changes (Basic):**

* Make a change to README.txt.

C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps15.jpg

* Use git restore README.txt to discard unstaged changes.

C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps16.jpg

* Make a change and stage it.

C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps17.jpg

C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps18.jpg

* Use git restore --staged README.txt to unstage a file.

C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps19.jpg

Lab 2: Branching and Merging

· Objective: To understand the concepts of branching, merging, and resolving conflicts in Git.

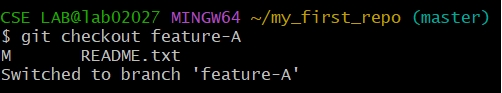
· Programs:

1. Create and Switch Branches:

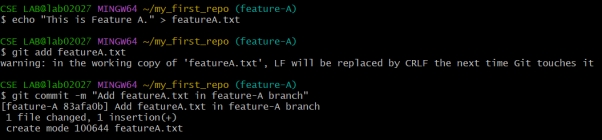
* Create a new branch named feature-A (git branch feature-A).

C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps20.jpg

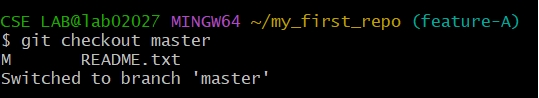
* Switch to feature-A (git checkout feature-A).



* Create a new file (e.g., featureA.txt) and commit it on feature-A.

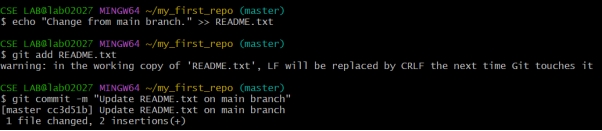


* Switch back to the main (or master) branch.

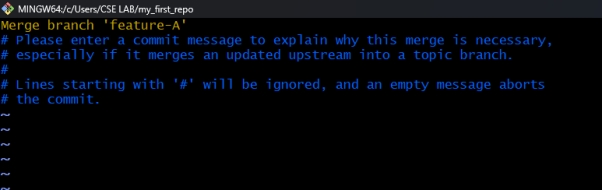


2. Merging Branches:

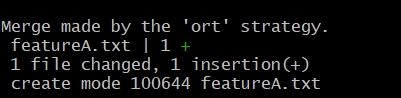
* On the main branch, make a change to README.txt and commit it.



* Merge feature-A into main (git merge feature-A).

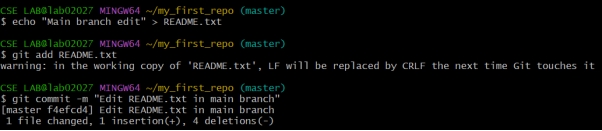


* Observe the merge process.

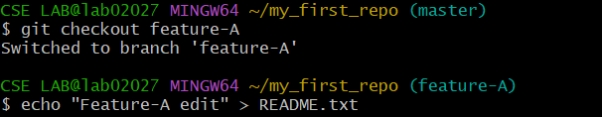


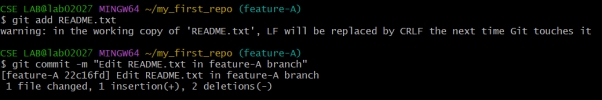
3. Conflict Resolution:

* On main, modify a line in README.txt.

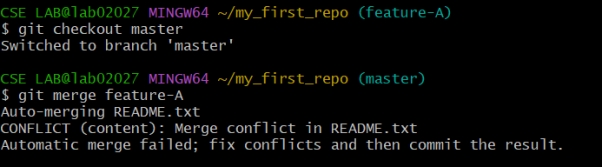


* On feature-A, modify the same line in README.txt differently.

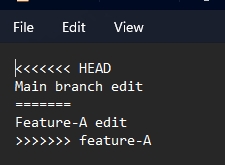


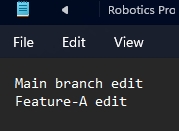


* Try to merge feature-A into main. Observe the merge conflict.



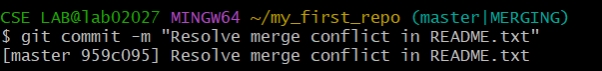
* Manually resolve the conflict in README.txt.

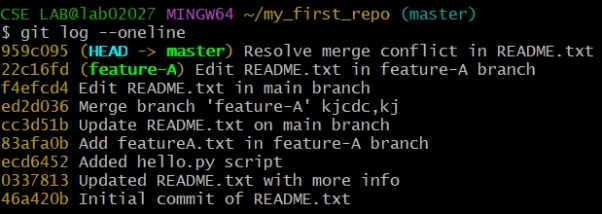




* Add the resolved file and commit the merge.

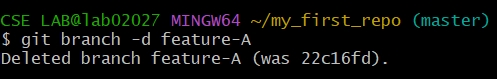
C:\Users\CSELAB~1\AppData\Local\Temp\ksohtml19824\wps33.jpg





4. Deleting Branches:

* Delete feature-A after it's merged (git branch -d feature-A).



* Attempt to delete an unmerged branch and observe the warning/error. Use -D to force delete.

