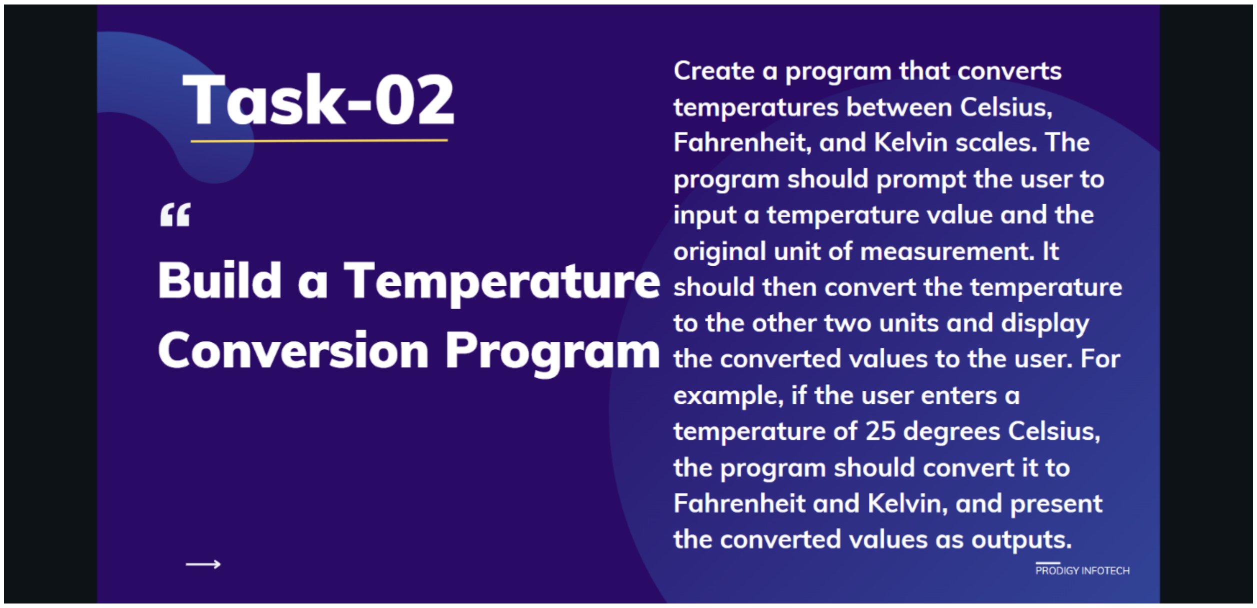
Prodigy InfoTech

Software Development

Track Code: SD

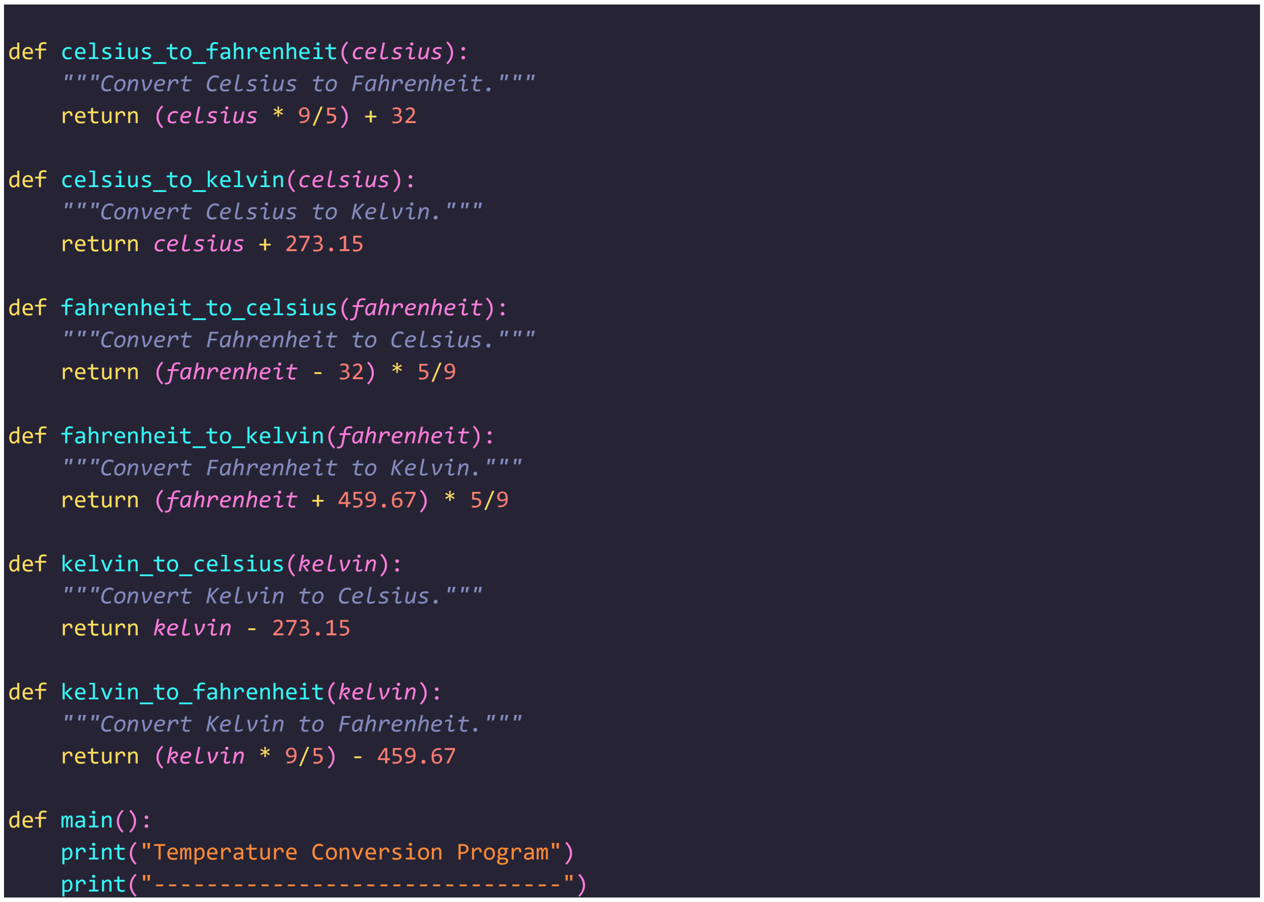

Task-02

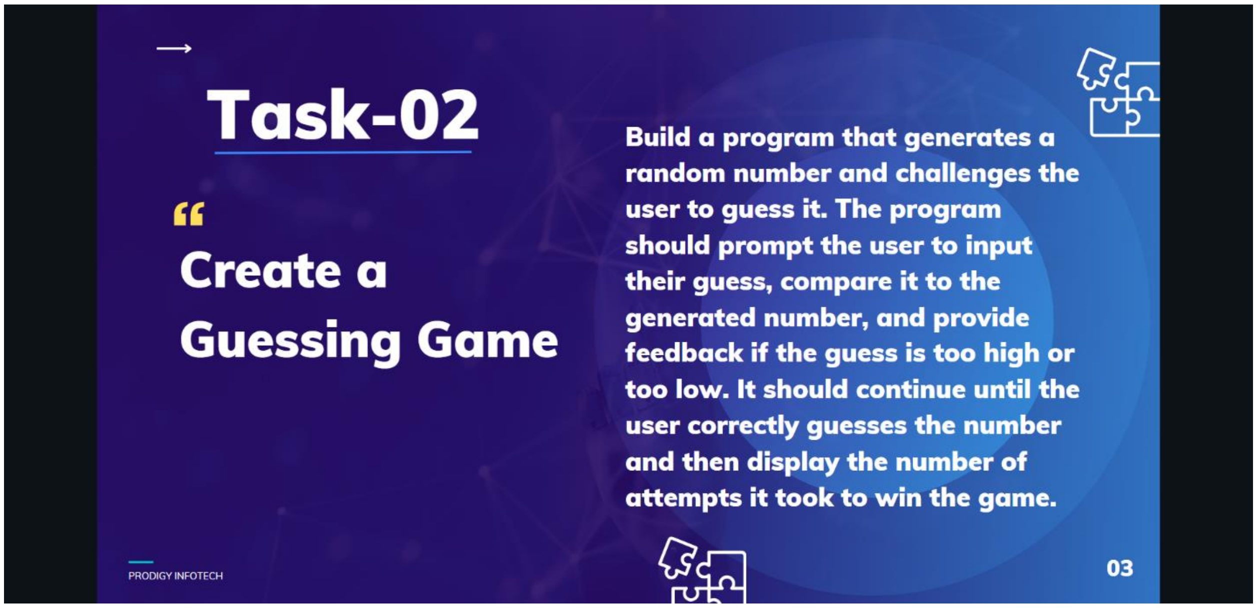
“

Build a Temperature Conversion Program

Create a program that converts temperatures between Celsius, Fahrenheit, and Kelvin scales. The program should prompt the user to input a temperature value and the original unit of measurement. It should then convert the temperature to the other two units and display the converted values to the user. For example, if the user enters a temperature of 25 degrees Celsius, the program should convert it to Fahrenheit and Kelvin, and present the converted values as outputs.

Source Code


```python
def celsius_to_fahrenheit(celsius):
    """Convert Celsius to Fahrenheit."""
    return (celsius * 9/5) + 32

def celsius_to_kelvin(celsius):
    """Convert Celsius to Kelvin."""
    return celsius + 273.15

def fahrenheit_to_celsius(fahrenheit):
    """Convert Fahrenheit to Celsius."""
    return (fahrenheit - 32) * 5/9

def fahrenheit_to_kelvin(fahrenheit):
    """Convert Fahrenheit to Kelvin."""
    return (fahrenheit + 459.67) * 5/9

def kelvin_to_celsius(kelvin):
    """Convert Kelvin to Celsius."""
    return kelvin - 273.15

def kelvin_to_fahrenheit(kelvin):
    """Convert Kelvin to Fahrenheit."""
    return (kelvin * 9/5) - 459.67

def main():
    print("Temperature Conversion Program")
    print("-------------------------------")
    try:
        temperature = float(input("Enter The Temperature Value: "))
        original_unit = input("Enter The Original Unit [Celsius, Fahrenheit, or Kelvin]: 
").lower()

        if original_unit == "celsius":
            fahrenheit = celsius_to_fahrenheit(temperature)
            kelvin = celsius_to_kelvin(temperature)
            print(f"\nConverted Temperatures:")
            print(f"Celsius: {temperature:.2f} °C")
            print(f"Fahrenheit: {fahrenheit:.2f} °F")
            print(f"Kelvin: {kelvin:.2f} K")
        elif original_unit == "fahrenheit":
            celsius = fahrenheit_to_celsius(temperature)
            kelvin = fahrenheit_to_kelvin(temperature)
            print(f"\nConverted Temperatures:")
            print(f"Celsius: {celsius:.2f} °C")
            print(f"Fahrenheit: {temperature:.2f} °F")
            print(f"Kelvin: {kelvin:.2f} K")
        elif original_unit == "kelvin":
            celsius = kelvin_to_celsius(temperature)
            fahrenheit = kelvin_to_fahrenheit(temperature)
            print(f"\nConverted Temperatures:")
            print(f"Celsius: {celsius:.2f} °C")
            print(f"Fahrenheit: {fahrenheit:.2f} °F")
            print(f"Kelvin: {temperature:.2f} K")
        else:
            print("Invalid Input. Please Enter Celsius, Fahrenheit, or Kelvin.")

    except ValueError:
        print("Invalid Input. Please Enter a Valid Numeric Tmperature Value.")

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



→

Task-02

“

Create a Guessing Game

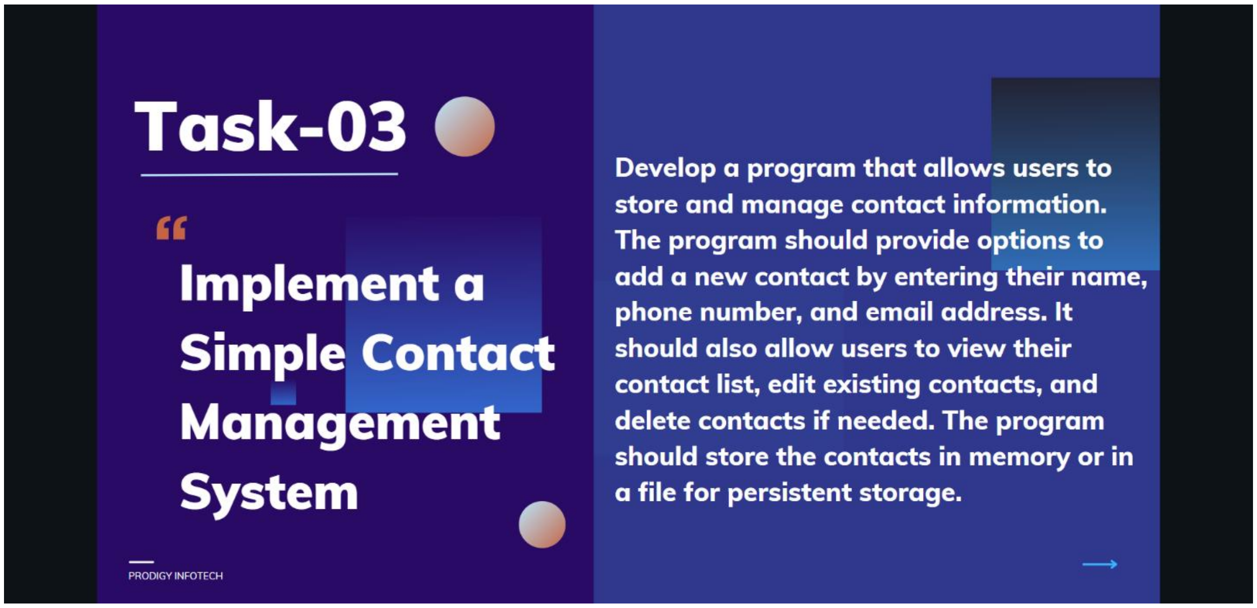
Build a program that generates a random number and challenges the user to guess it. The program should prompt the user to input their guess, compare it to the generated number, and provide feedback if the guess is too high or too low. It should continue until the user correctly guesses the number and then display the number of attempts it took to win the game.

PRODIGY INFOTECH

03

Source Code

```python
import random

def main():
    print("Welcome To The Number Guessing Game!")
    print("I Have Chosen A Random Number Between 1 And 100.")
    print("Try To Guess It Within 10 Attempts.\n")

    the_number = random.randint(1, 100)
    attempts = 0

    while True:
        guess = int(input("Enter Your Guess: "))
        attempts += 1

        if guess < the_number:
            print("Too Low! Try Again.")
        elif guess > the_number:
            print("Too High! Try Again.")
        else:
            print(f"Congratulations...! You Guessed It Right In {attempts} Attempts.")
            break

        if attempts >= 10:
            print(f"Sorry, You've Reached The Maximum Number Of Attempts. The Correct Number 
Was {the_number}.")
            break

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



Task-03

"Implement a Simple Contact Management System

Develop a program that allows users to store and manage contact information. The program should provide options to add a new contact by entering their name, phone number, and email address. It should also allow users to view their contact list, edit existing contacts, and delete contacts if needed. The program should store the contacts in memory or in a file for persistent storage.

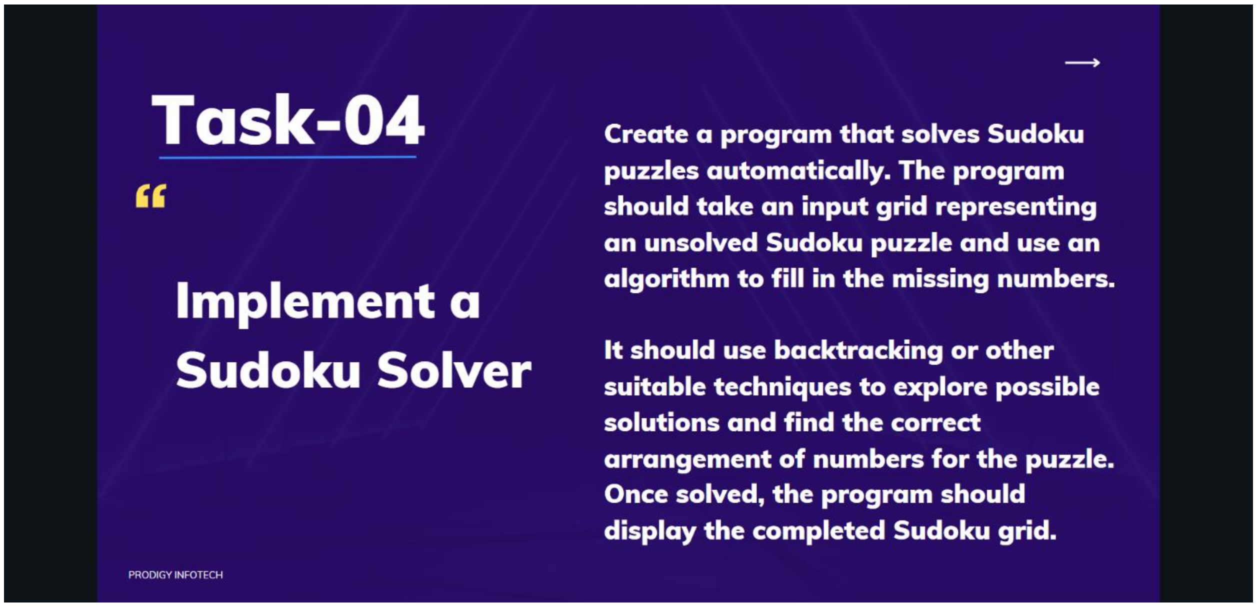
PRODIGY INFOTECH

Source Code

```
import datetime

class Contact:
    def __init__(self, name, contact_number, email=None):
        self.name = name
        self.contact_number = contact_number
        self.email = email
        self.date_created = datetime.datetime.now().isoformat()

    def __repr__(self):
        return f"({self.name}, {self.contact_number}, {self.email}, {self.date_created})"
```

```
from model import Contact
from tinydb import TinyDB, Query

db = TinyDB("contacts.json")
contacts_table = db.table("contacts")

def add_contact():
    name = input("Enter Contact Name: ")
    contact_number = input("Enter Contact Number: ")
    email = input("Enter Email Address (optional): ")

    contact = Contact(name, contact_number, email)
    contacts_table.insert(contact.__dict__)
    print(f"Contact '{name}' Added Successfully!")
```

```python
def view_contacts():
    all_contacts = contacts_table.all()
    for contact in all_contacts:
        print(contact)

def main():
    while True:
        print("\nContact Book Menu:")
        print("1. Add Contact")
        print("2. View Contacts")
        print("3. Exit")
        choice = input("Enter Your Choice: ")

        if choice == "1":
            add_contact()
        elif choice == "2":
            view_contacts()
        elif choice == "3":
            print("Exiting Contact Book... Goodbye!")
            break
        else:
            print("Invalid Choice... Please select 1, 2, or 3.")

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



Task-04

"

Implement a Sudoku Solver

Create a program that solves Sudoku puzzles automatically. The program should take an input grid representing an unsolved Sudoku puzzle and use an algorithm to fill in the missing numbers.

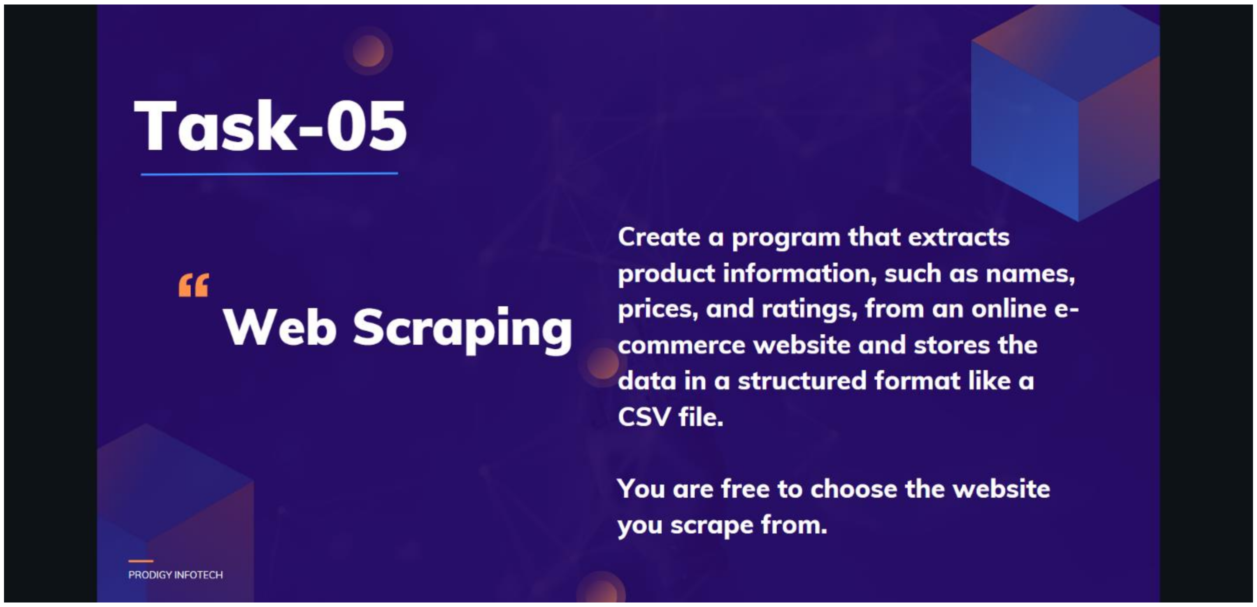
It should use backtracking or other suitable techniques to explore possible solutions and find the correct arrangement of numbers for the puzzle. Once solved, the program should display the completed Sudoku grid.

PRODIGY INFOTECH

Source Code

```
def is_valid(board, row, col, num):
    # Check row, column, and 3x3 subgrid for validity
    for i in range(9):
        if board[row][i] == num or board[i][col] == num:
            return False
    start_row, start_col = 3 * (row // 3), 3 * (col // 3)
    for i in range(start_row, start_row + 3):
        for j in range(start_col, start_col + 3):
            if board[i][j] == num:
                return False
    return True

def solve_sudoku(board):
    for row in range(9):
        for col in range(9):
            if board[row][col] == 0:
                for num in range(1, 10):
                    if is_valid(board, row, col, num):
                        board[row][col] = num
                        if solve_sudoku(board):
                            return True
                        board[row][col] = 0
                return False
    return True

def print_sudoku(board):
```

```python
    for row in board:
        print(" ".join(map(str, row)))

if __name__ == "__main__":
    # Example unsolved Sudoku grid (0 represents empty cells)
    sudoku_grid = [
        [5, 3, 0, 0, 7, 0, 0, 0, 0],
        [6, 0, 0, 1, 9, 5, 0, 0, 0],
        [0, 9, 8, 0, 0, 0, 0, 6, 0],
        [8, 0, 0, 0, 6, 0, 0, 0, 3],
        [4, 0, 0, 8, 0, 3, 0, 0, 1],
        [7, 0, 0, 0, 2, 0, 0, 0, 6],
        [0, 6, 0, 0, 0, 0, 2, 8, 0],
        [0, 0, 0, 4, 1, 9, 0, 0, 5],
        [0, 0, 0, 0, 8, 0, 0, 7, 9]
    ]

    if solve_sudoku(sudoku_grid):
        print("Solved Sudoku:")
        print_sudoku(sudoku_grid)
    else:
        print("No Solution Exists...")
```


Task-05

“ Web Scraping

Create a program that extracts product information, such as names, prices, and ratings, from an online e-commerce website and stores the data in a structured format like a CSV file.

You are free to choose the website you scrape from.

PRODIGY INFOTECH

Source Code

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

baseurl = "https://www.thewhiskyexchange.com"
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.82 Safari/537.36'
}

def scrape_product_links():
    url = f"{baseurl}/c/35/japanese-whisky"
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')
    product_list = soup.find_all("li", {"class": "product-grid__item"})
    product_links = [baseurl + item.find("a")["href"] for item in product_list]
    return product_links

def scrape_product_details(product_url):
    response = requests.get(product_url, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')
    product_name = soup.find("h1", {"class": "product-main__name"}).text.strip()
    product_price = soup.find("p", {"class": "product-action__price"}).text.strip()
    product_rating = soup.find("span", {"class": "review-overview__rating"}).text.strip()
```

```python
    return {"Name": product_name, "Price": product_price, "Rating": product_rating}

def main():
    product_links = scrape_product_links()
    product_data = []
    for link in product_links:
        product_data.append(scrape_product_details(link))

    df = pd.DataFrame(product_data)
    df.to_csv("whisky_products.csv", index=False)
    print("Data Saved To whisky_products.csv")

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