my-aaignment-4-joy-joy9

June 3, 2025

[1]:
```
def emotion_to_emoji(text):
    emotions = {'happy': ' ', 'sad': ' ', 'love': ' ', 'angry': ' ', 'excited':␣
 ↪' ', 'heart': ' ', 'joy': ' ', 'worry': ' ', 'smile': ' ', 'cry': ' '}
    words = text.split()
    for i in range(len(words)):
        clean_word = words[i].lower().strip('.,!?;:')
        if clean_word in emotions:
            words[i] = words[i].replace(clean_word, emotions[clean_word])
    return ' '.join(words)

# Test the function
sample_text = "I feel happy and excited about this wonderful day, but sometimes␣
 ↪I worry and feel sad about the future"
result = emotion_to_emoji(sample_text)
print(result)
```

```
I feel   and   about this wonderful day, but sometimes I   and feel   about the
future
```

0.1 quesion 2

[38]:
```
import pandas as pd

# Load and merge all data
url = "https://raw.githubusercontent.com/Majid-Sohrabi/2025_Python_Cognitive/
 ↪main/Datasets/"
dfs = {n: pd.read_csv(f"{url}{n}.txt", sep=",", encoding="latin1") for n in␣
 ↪['users','ratings','books']}
for df in dfs.values(): df.columns = df.columns.str.strip()
data = dfs['ratings'].merge(dfs['books'], on='ISBN').merge(dfs['users'],␣
 ↪on='user_id')
data.to_csv("JOY_BOOKS_DATA.txt", index=False)

# Analysis
d = data[data['rating'] > 0]
rc = d['title'].value_counts()
ar = d.groupby('title')['rating'].mean()
```

```
print("Most reviewed book:", rc.idxmax())
print("Least reviewed book:", rc.idxmin())
print("Highest rated book (min 100 reviews):", ar[rc >= 100].idxmax())

print("\nHighest rated book per country:")
for _, r in d.groupby(['country','title'])['rating'].mean().reset_index().
  ↪sort_values(['country','rating'], ascending=[True,False]).
  ↪drop_duplicates('country').iterrows():
    print(f"{r['country']}: {r['title']} (Rating: {r['rating']:.2f})")
```

```
Most reviewed book: Wild Animus
Least reviewed book: Past Due (Lashner, William)
Highest rated book (min 100 reviews): Harry Potter and the Goblet of Fire (Book
4)

Highest rated book per country:
argentina: Cyrano De Bergerac (Bantam Classics) (Rating: 10.00)
australia: 1984 (Rating: 10.00)
austria: 1984 (Rating: 10.00)
belgium: Almost Adam (Rating: 10.00)
brazil: 1984 (Rating: 10.00)
canada: 9-11 (Rating: 10.00)
chile: Dharma Bums (Rating: 10.00)
china: A Wind in the Door (Rating: 10.00)
croatia: Chocolat (Rating: 9.00)
denmark: Animal Farm (Rating: 10.00)
finland: A Time to Kill (Rating: 10.00)
france: 52 Deck Series: 52 Ways to Celebrate Friendship (Rating: 10.00)
germany: 52 Deck Series: 52 Ways to Celebrate Friendship (Rating: 10.00)
greece: Bel Canto: A Novel (Rating: 10.00)
india: From the Mixed-Up Files of Mrs. Basil E. Frankweiler (Rating: 10.00)
indonesia: Notes From a Small Planet (Rating: 9.00)
iran: A Perfect Stranger (Rating: 10.00)
ireland: A Little Princess (Rating: 10.00)
israel: American Psycho (Vintage Contemporaries) (Rating: 9.00)
italy: Beyond The Far Side (Rating: 10.00)
japan: A Brief History of Time : From the Big Bang to Black Holes (Rating:
10.00)
malaysia: 1st to Die: A Novel (Rating: 10.00)
mexico: El Codigo Da Vinci / The Da Vinci Code (Rating: 10.00)
netherlands: A History of the World in 10 1/2 Chapters (Vintage International)
(Rating: 10.00)
new zealand: A Heartbreaking Work of Staggering Genius (Rating: 10.00)
nigeria: Seven Habits Of Highly Effective People (Rating: 10.00)
norway: A Lesson Before Dying (Vintage Contemporaries (Paperback)) (Rating:
10.00)
```

```
pakistan: The Secret Life of Bees (Rating: 7.00)
philippines: A Fire Upon The Deep (Zones of Thought) (Rating: 10.00)
poland: A Woman of No Importance (Penguin Popular Classics) (Rating: 10.00)
portugal: 1984 (Everyman's Library) (Rating: 10.00)
romania: Angels &amp;amp; Demons (Rating: 10.00)
singapore: 52 Deck Series: 52 Ways to Celebrate Friendship (Rating: 10.00)
slovakia: Selected Tales (Penguin Popular Classics) (Rating: 10.00)
south africa: A Walk to Remember (Rating: 10.00)
south korea: A Bend in the Road (Rating: 10.00)
spain: 52 Deck Series: 52 Ways to Celebrate Friendship (Rating: 10.00)
sweden: Dracula (Wordsworth Classics) (Rating: 10.00)
switzerland: Angela's Ashes: A Memoir (Rating: 10.00)
turkey: The Catcher in the Rye (Rating: 10.00)
united kingdom: 2001 Cross-Stitch Designs: The Essential Reference Book (Rating:
10.00)
usa: 2001 Cross-Stitch Designs: The Essential Reference Book (Rating: 10.00)
```

0.2 QUESTION3

[7]:
```python
class Student:
    def __init__(self, sid, age, marks):
        self.sid, self.age, self.marks = sid, age, marks
        self.valid = age > 17 and 0 <= marks <= 100
        self.qualified = self.valid and marks >= 35
        self.scholarship = 100 if marks >= 96 else 75 if marks >= 86 else 50 if␣
 ↪marks >= 76 else 20 if marks >= 60 else 10 if marks >= 50 else 0 if self.
 ↪qualified else 0
        print(f"Student {sid}: Age={age}, Marks={marks}, Valid={self.valid},␣
 ↪Qualified={self.qualified}, Scholarship={self.scholarship}%")

# Test with 5 students and print results
students = [("S001", 20, 95), ("S002", 16, 80), ("S003", 22, 30), ("S004", 19,␣
 ↪67), ("S005", 21, 88)]
for data in students:
    Student(*data)
```

```
Student S001: Age=20, Marks=95, Valid=True, Qualified=True, Scholarship=75%
Student S002: Age=16, Marks=80, Valid=False, Qualified=False, Scholarship=50%
Student S003: Age=22, Marks=30, Valid=True, Qualified=False, Scholarship=0%
Student S004: Age=19, Marks=67, Valid=True, Qualified=True, Scholarship=20%
Student S005: Age=21, Marks=88, Valid=True, Qualified=True, Scholarship=75%
```

0.3 QUESTION 4

[13]:
```
class TollBooth:
    def __init__(self):
        self.vehicles, self.cash = 0, 0
    def payVehicle(self, toll):
        self.vehicles += 1; self.cash += toll; print(f"Paid: {toll} rubles")
    def noPayVehicle(self):
        self.vehicles += 1; print("Free pass")
    def display(self):
        print(f"Vehicles: {self.vehicles}, Cash: {self.cash} rubles")

booth = TollBooth()
tolls = [0, 500, 1200, 1500]
while True:
    choice = input("Vehicle: 1-Bike(Free) 2-Car(500) 3-Truck(1200)␣
 ↪4-Heavy(1500) 5-Stats 6-Exit: ")
    if choice in "1234": (booth.noPayVehicle() if tolls[int(choice)-1]==0 else␣
 ↪booth.payVehicle(tolls[int(choice)-1]))
    elif choice=="5": booth.display()
    elif choice=="6": booth.display(); break
```

```
Vehicle: 1-Bike(Free) 2-Car(500) 3-Truck(1200) 4-Heavy(1500) 5-Stats 6-Exit:  4
Paid: 1500 rubles
Vehicle: 1-Bike(Free) 2-Car(500) 3-Truck(1200) 4-Heavy(1500) 5-Stats 6-Exit:  3
Paid: 1200 rubles
Vehicle: 1-Bike(Free) 2-Car(500) 3-Truck(1200) 4-Heavy(1500) 5-Stats 6-Exit:  5
Vehicles: 2, Cash: 2700 rubles
Vehicle: 1-Bike(Free) 2-Car(500) 3-Truck(1200) 4-Heavy(1500) 5-Stats 6-Exit:  2
Paid: 500 rubles
Vehicle: 1-Bike(Free) 2-Car(500) 3-Truck(1200) 4-Heavy(1500) 5-Stats 6-Exit:  1
Free pass
Vehicle: 1-Bike(Free) 2-Car(500) 3-Truck(1200) 4-Heavy(1500) 5-Stats 6-Exit:  6
Vehicles: 4, Cash: 3200 rubles
```

0.4 QUESTION 5

[18]:
```
presidents_full = ["Michael Jackson", "Nelson Mandela", "Albert Einstein",␣
 ↪"Mahatma Gandhi", "Yuri Gagarin",
       "Marilyn Monroe", "Anne Frank", "Thomas Alva Edison", "Narendra Modi",␣
 ↪"Neil Armstrong",
```

```
        "J.K.Rowling", "Leo Tolstoy", "Henry Ford", "Pablo Picasso", "Steve␣
 ↪Jobs",
        "Abraham Lincoln", "Cristiano Ronaldo", "Magnus Carlsen", "Alexander␣
 ↪Pushkin", "Donald Trump",
        "Charlie Chaplin", "Leonardo Da Vinci", "Leonardo DiCaprio", "Jackie␣
 ↪Chan", "Stephen Hawking",
        "Osho Rajneesh", "John Cena", "Bernard Arnault", "Mark Twain", "Usain␣
 ↪Bolt",
        "Bruce Lee", "Jalāl al-Dīn Muḥammad Rumi", "Alex Ovechkin", "Elon␣
 ↪Musk", "John F. Kennedy",
        "James Stephen", "Richard Branson", "Michael Phelps", "Jeff Bezos",␣
 ↪"Ken Jeong", "Swami Vivekananda",
        "Bill Clinton", "Daniil Medvedev", "Mother Teresa"]

# Extract last names using original approach
surname_list = []
for full_name in presidents_full:
    name_parts = full_name.split(' ')
    final_name = name_parts[len(name_parts) - 1]
    surname_list.append(final_name)

print("Complete list of surnames:")
print(surname_list)
```

```
Complete list of surnames:
['Jackson', 'Mandela', 'Einstein', 'Gandhi', 'Gagarin', 'Monroe', 'Frank',
'Edison', 'Modi', 'Armstrong', 'J.K.Rowling', 'Tolstoy', 'Ford', 'Picasso',
'Jobs', 'Lincoln', 'Ronaldo', 'Carlsen', 'Pushkin', 'Trump', 'Chaplin', 'Vinci',
'DiCaprio', 'Chan', 'Hawking', 'Rajneesh', 'Cena', 'Arnault', 'Twain', 'Bolt',
'Lee', 'Rumi', 'Ovechkin', 'Musk', 'Kennedy', 'Stephen', 'Branson', 'Phelps',
'Bezos', 'Jeong', 'Vivekananda', 'Clinton', 'Medvedev', 'Teresa']
```

0.5 QUESTION 5B

[27]:
```
# Challenge 2: Fill in the blanks solutions
print("\nChallenge 2 Solutions:")

# 2a: Total length of strings
total = 0
for word in ["red", "green", "blue"]:
    total = total + len(word)
print("Total length:", total)

# 2b: List of word lengths
lengths = []
for word in ["red", "green", "blue"]:
    lengths.append(len(word))
```

```
print("Word lengths:", lengths)

# 2c: Concatenate all words
words = ["red", "green", "blue"]
result = ""
for word in words:
    result = result + word
print("Concatenated:", result)
```

```
Challenge 2 Solutions:
Total length: 12
Word lengths: [3, 5, 4]
Concatenated: redgreenblue
```